International Journal of
Soft Computing And Software Engineering (JSCSE)
e-ISSN: 2251-7545
Vol.3,No.11, 2013
Published online: Nov 25, 2013

DOI: 10.7321/jscse.v3.n11.4

Agent-based Service Oriented Architecture (SOA) for Cross-platform Communication

Najhan M.Ibrahim*[1], Mohd Fadzil Hassan [2], M.Hussin Abdullah[3]
Department of Management and Information Technology
Kolej Universiti Islam Sultan Azlan Shah, Bukit Chandan, 33000 Bandar DiRaja Kuala Kangsar, Perak Darul Ridzuan, Malaysia
Department of Computer and Information Sciences
Universiti Teknologi PETRONAS, Bandar Seri Iskandar, 31750 Tronoh, Perak, Malaysia
Najhan@kuisas.edu.my, mfadzil_hassan@petronas.com.my, mhussin@kuisas.edu.my

Abstract. **Service-oriented architecture (SOA) refers to a software paradigm to develop systems where it is a collection of services in essence, the services of which communicate with each other. The communication not only can engage simple data passing but also can connect two or more services coordinating some activities. Cross-platform communication among systems is used to describe as Information Technology in assisting and managing any work or transaction activity without any interruption. Web services are distributed software components that support cross-platform communication among applications over a network. Currently, there are many specifications and deployment styles of Web services to construct SOA applications as well as several implementation techniques and technologies. However, integration and communication among cross-platforms cannot be guaranteed due to differences in application deployment, versions of Web service standards and the specifications supported. Middleware has become the accepted system to connect cross-platforms and distributed applications, and the Message Oriented Middleware is the most difficult to ignore in integration and communication of distributed systems. The proposed solution is a novel framework to automate the integration and communication among different SOA-based applications based on an extension of the Message Oriented Middleware (MOM) with Agent technology. This framework integrates agent technology for flexibility and adaptively in the communication flow and MOM as a common attach technology for distributed systems. Furthermore, a Translation Model is proposed to facilitate the translation process among different Web service descriptions. The aim is to provide a generic and autonomic translation process to support the cross-platform communication framework. As a proof of the concept, the research work is projected on a case study of the real world SOA application for an agent-based trading system.**

Keywords: *Service Oriented Architecture (SOA), Message Oriented Middleware (MOM), Agent Technology, Web Services, Agent-based SOA, Cross-platform communication.*

* Najhan Muhamad Ibrahim
Department of Management and Information Technology
Kolej Universiti Islam Sultan Azlan Shah, Bukit Chandan, 33000 Bandar DiRaja Kuala Kangsar, Perak Darul Ridzuan, Malaysia,
Email: *hanfast@gmail.com* Tel:+6012-692-7200

1. Introduction

Service-oriented architecture (SOA) is a software paradigm for integrating loosely-coupled and distributed services into an interconnected workflow or organization process. SOA-based systems may integrate both heritage and new services created by an organization, either internally or via external trusted partners [1]. It allows an application to be connected and executed across multiple business domains at geographically distributed locations. Within an SOA system, services compose workflows and integrate different processes on run-time to enable flexible connections and collaborations among business partners [2].The loose coupling of services and reusable properties across a multi-domain makes SOA as the most flexible and interoperable software architecture. Moreover, the integrations of the legacy system become significant due to very fast growth of software and technology development. SOA can, furthermore, help an organization to overcome some of the integration issues with specific technological deployment such as by an XML file that is able to pass over different software applications [3, 4].

Web services are distributed components that support an interoperable machine-to-machine interaction over a network. It is a set of software that can be used in different ways to implement an application system. In this moment of time, there are several specifications of web services available, for example, XML (eXtensible Markup Language), SOAP (Simple Object Access Protocol), and WSDL (Web Services Description Language); it also includes several implementation technologies such as REST (Representational State Transfer). SOA is an architectural style, whereas Web services are an implementation approach [5]. In addition, SOAP is a protocol specification for exchanging structured information in the implementation of Web Services within a computer network. It relies on Extensible Markup Language (XML) for messaging format. Many applications have adopted SOAP to facilitate the communication between them. In [6], how to use SOAP to integrate different applications was explored. XML is a markup language for data structure exchanging and formatting. Web service properties such attributes, interface and others properties are described by WSDL. A WSDL document can be read by other client systems to learn about the service. On the other hand, REST refers to a web service approach to implement an SOA system, it is not a standard or API. REST is also used to build distributed applications such as Web applications. The existing standards including HTTP and XML can be used to implement REST applications. [7-12].

Currently, web service standards and specifications are used to implement SOA and connect it to different applications to ensure the cross-platform communication among them. Cross-platform communication can be defined as the ability of diverse systems and organizations to work (inter-operate) or communicate together efficiently. In the loosely coupled environment of SOA, separate resources do not need to know the details of each work, but they need to have enough common ground to integrate or exchange messages without any error or misunderstanding [13]. In addition, there has been an increasing interest in the cross-platform communication of different approaches to implement SOA and also different standards of technology in each approach such as XML schema, SOAP, WSDL, and WS-* (WS Splat) protocols. Various application systems or Middleware technologies are frequently used as the communication infrastructure for a distributed system to enable the cross-platform communication [14, 15] such as WS2JADE, P-GRID and REST based applications. Therefore, the communication and integration between different approaches of web services to develop the SOA application will lead to communication barriers. [16] Mentioned that different types of web service approaches, deployment styles and even standards will also raise communication issues. This is due to the different purposes and issues to solve in each deployment of web services to implement SOA. Most of the adopted applications are problem specific and do not investigate the general view of integration and communication among different domains of SOA.

2. Background

Middleware is a basic system to connect distributed applications because of its properties in cost and time savings (increases productivity) [3]. Even though SOAP is the most accepted communication mechanism for SOA applications, other communication techniques are still used such as REST, and

Messaging services [17]. However, integration cannot be assured because of various reasons such as differences in applications and versions of Web Service standards and specifications supported in the SOA deployment style [16]. Message Oriented Middleware (MOM) is becoming difficult to ignore in the integration and communication of the distributed cross-platform application and is strongly depended on in message oriented communication [18]. The authors have proposed an enhancement of MOM to make use of an intelligent bright for information exchange in multi-domain restriction [19]. MOM is also a communication middleware that provides program-to-program connection by message passing. It supports multiple protocols that consist of a facility to support reliable and scalable high-performance distributed platforms. Most MOM environments are implemented with queued message store-and-forward capability, which is Message Queuing Middleware (MQM) [19-21]. For these reasons, the use of MOM technology to solve cross-platform issues among SOA applications is suggested in this study.

. However, in this thesis the studies were explored and analyzed related to cross-platform integration and communication. As a result, some the following weaknesses of Message Oriented Middleware (MOM) were found [22-27].

- It is able to support only a small number of executions and has a limited ability to link with complex applications.
- It is unable to trace the execution of the application and cannot rollback tasks at any level. The MOM's based execution models are too simple. Therefore, it cannot record any level of execution in applications or maintain the states of those executing tasks.
- When users need to terminate tasks, MOM cannot correctly rollback all or some of the specified tasks.
- It has a limited ability to link complex applications.
- It loses the trace of execution and is unable to rollback the task.
- It is very low level in autonomous.

To overcome these weaknesses, some researchers have been conducted to extend Message Oriented Middleware (MOM). Most of current the MOM extensions address only one or two of the communication issues to solve their specific integration and communication problem. They support only point-to-point or one-to-one communication [27] like instant messaging. In this situation, only two participants are able to communicate at a time and both have to be in active mode. Therefore, it is strongly argued that research in this area is needed to consolidate these distinct means of communication for the cross-platform system to find a common ground toward the establishment of a generic framework for cross-platform communication to be applied within the SOA context. In addition, most of the researchers within this area are focusing more on the cross-platform specification, standardization and requirements [28, 29]. Based on [29], there are 27 interoperate requirements to achieve collaboration and integration. A literature survey has been and it was found that there are some overlapping attributes and some attributes that have still not been taken into consideration. Therefore, in the first survey matrix, the authors have proposed 16 inter-operative requirements for cross-platform integration and communication [30]. In this comparative study, the authors have considered 8 of the most significant communication requirements for cross-platforms. It would be difficult to implement all of the 16 attributes in this first pilot study. This research work will focus more on generic and autonomous level of communication among different SOA applications with multi agent technology. Based on the mentioned limitations of MOM, such as its limited ability to link complex applications, losing the trace of execution and its being low level autonomous, it is necessary to construct a generic, flexible and autonomic framework for communication to support the multi-domain of a SOA system.

2.1 Scope of Study

This research work deliberates on enabling communication among cross-platforms within an SOA system across platforms, operating systems and programming languages. Furthermore, it proposes integration of agent technology into Message Oriented Middleware. In particular, the authors are concerned with Message Oriented middle for message-based communication. The authors will provide a comparative study with related works. Moreover, an enhancement of the communication framework is proposed and the attributes of this framework through an experimental study are also illustrated. The search for resources and the making of a contract are not included in this work with an assumption that both applications have been established in the connection based on the contract of an SOA system. The complex content of the message request and response will also not be considered; this is because the study will merely focus on the communication process between both systems.

3. Agent-based Technology

Over the last decade, Agent technology has shown great potential for solving problems in the large scale distributed and cross-platform systems. The reason for the growing success of Multi-agent technology in this area is that the inherent distribution allows a natural decomposition of the system into multiple agents that interact with each other to achieve a desired global goal [31]. The Multi-agent technology can significantly enhance the design and analysis of problem domains under three following conditions [32]: 1). the problem domain is geographically distributed, 2). the sub-systems exist in a dynamic environment, and 3). the sub-systems need to more flexibly interactive with each other. The domain of traffic and transportation systems is well suited for an Agent-based approach because of its geographically distributed and dynamic changing nature [33]. This study’s literature research shows that the techniques and methods resulting from the field of Agent and Multi-agent systems have been applied to many aspects of distributed systems and cross-platform communication, including modelling and simulation of an Agent Platform for Reliable Asynchronous Distributed Programming [34]. Multi-agent Systems can be considered as overviews of a new Paradigm for Distributed Systems [35] and more exploration of this will be found in the related work section.

3.1 Agent Communication Language (ACL)

Agent Communication Language (ACL), proposed by the Foundation for Intelligent Physical Agents (FIPA), is a proposed standard language for Agent communications. Knowledge Query and Manipulation Language (KQML) meanwhile is another proposed standard as well. The most popular ACLs are FIPA-ACL by FIPA and KQML. Both rely on the speech act theory developed by Searle in 1960 and enhanced by Winograd and Flores in the 1970s. They defined a set of per-formatives (list of FIPA Communicative Act Specifications) and their meanings. The content of the per-formatives is not standardized, but varies from system to system. To make agents understand each other, they not only have to speak the same language, but must also have a common ontology. Ontology is a part of the agent's knowledge base that describes what kind of things an agent can deal with and how they are related to each other [36].

The main idea of an ACL is to represent an appropriate framework that allows different agents to interact and communicate with significant statements that pass on information about their environment or knowledge. An important part of the Agent approach is the concept that agents (like humans) can function more effectively in groups characterized by cooperation and the division of workers [37]. Agent programs are designed to independently collaborate with each other in order to satisfy their goal. The balance between collaboration and fulfilling its own goals is made by each agent individually and depending on the situation.

3.2 Multi Agent Systems (MAS)

Multi-agent systems (MAS) are the wide subject of research to study the systems developed by multiple heterogeneous intelligent software entities, called agents. The agents in MAS are able to participate, collaborate or simply leave. In recent years the interest in MAS has grown greatly, and today Multi-agent technology is being used in a large range of significant industrial application areas ranging from information management through industrial process control to

International Journal of
Soft Computing And Software Engineering (JSCSE)
e-ISSN: 2251-7545
Vol.3,No.11, 2013
Published online: Nov 25, 2013
DOI: 10.7321/jscse.v3.n11.4

electronic commerce. All these applications have one thing in common. Agents must be able to communicate with each other to decide what decision and action to take and how this action can be coordinated with other actions. The language used by the agent for this exchange is the Agent communication language (ACL). An ACL stems from the need to coordinate the action of an agent with that of the other agents. It can be used to share information and knowledge among agents in a distributed and cross-platform computing environment, and also to request the performance of a task [38-40].

3.3 Agent technology for a cross-platform system

Agent technology has an important potential to facilitate connectivity in a distributed and cross-platform system. The main reason for the great success of agent technology in this area is the flexibility of the connection and integration among different applications over a distributed environment [31]. The agent software is suitable for three environments: the first is the distributed system, the second is for a dynamic environment and the last is for the system that needs a flexible interaction. An environment of distributed and cross-platform communication systems is well suited for an agent-based system because of it is distributed in nature and dynamically flexible. Literature studies have shown that the enhancement and practicality has been proved from the field of agent technology as they have been deployed to solve many issues of distributed systems and cross-platform environments [34]. For example, modeling and simulation of the Agent Platform for Reliable Asynchronous Distributed Programming[34], Multi-agent Systems: Overview of a New Paradigm for Distributed Systems [35] and more exploration in the related work section.

3.4 Related Works

In literature study, there has been an increasing amount of extension of the Agent technology for a distributed and cross-platform system in the SOA context due to its great capability, facility, flexibility and support for the multi-protocols of the Multi-agent system (MAS). These works have been carrying on from different approaches. In the proposed framework for this present, Multi-agent are responsible for managing the communication and translating the message that are out-going and in-coming to enable the communication across platforms, operating systems and programming languages. Nonetheless, no extension has been applied for cross-platforms in the scope of this thesis; the state-of-the-art research on this area has been studied to decide which cross-platform specification is most suitable to implement the suggested communication framework. In this section, some of the cross-platform communication techniques with Agent technology suggested by other researchers in different prospective are presented.

Identifying the quality attributes related to cross-platform communication is the best step to develop the framework. One of the important works related to enhancement of Agent technology for a distributed system is an Agent platform for reliable asynchronous distributed communication. Bellissard L. et al. [34] introduced a distributed communication model based on autonomous agents' software. Agents behave as the attached software components and provide an atomic execution from node to node. They are also the dynamic objects which can run in parallel and have their own state in each communication. They act regarding to an event and reaction model. An event is a typed data structure or method used to exchange information with other agents.

In another similar work, A. Lin and P. Maheshwari [33] proposed to construct an Agent-based middleware (AbM) for Web Service dynamic integration on Peer-to-Peer networks to facilitate the integration of optimal quality of Web Services for application integration. AbM will dynamically and autonomously accomplish the goal on behalf of a user by employing the best quality of Web Services that are purely distributed. With AbM, system developers can save costly time by asking agents to collect as many Web Services as they need and understand their usages. In addition, Web Services provide a function that may be large and frequently changes, but in a composition pattern they are to achieve a business process goal. Web Service dynamic integration (WSDI) is the concept of replacing Web Services with other better quality Web Services in integration prototypes at run time to maximize the quality of the performance in a business process.

In a recent study by Xiang Li [41], it was mentioned that to integrate process operation systems effectively, an Agent-XML based information integration platform is proposed. The subsystems are encapsulated as agents based on XML and Agent technology. Based on the integration concept, the agents deployed in different domains of the system are integrated. Since the encapsulation of different sub-systems has been implemented by Agent software, the security and stability are guaranteed in this integration platform. This is also to ensure the good cross-platform communication between different agents which is handled by XML. Figure 1 elaborates on the information integration platform of the Agent-based XML. In addition, Xiang Li has identified four basic requirements for this integration platform: (1) the integration should support communication among different modules, (2) the integration should be dynamic, (3) the security and stability should be assured from former systems, and (4) the integration is to extend the function, and not neglect the function of the sub-system. Furthermore, Xiang Li proposed three basic functions as shown in Figure below; namely (1) data integration and system cooperation, (2) the data analysis and decision support and (3) User interaction.

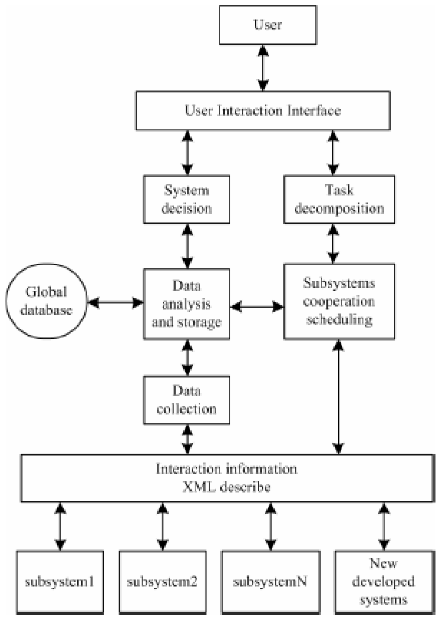

User
User Interaction Interface
System decision
Task decomposition
Global database
Data analysis and storage
Subsystems cooperation scheduling
Data collection
Interaction information XML describe
subsystem1
subsystem2
subsystemN
New developed systems

Figure 1. The Agent-XML-based information integration platform [41]

Cervera et al. [42] implemented a different framework for cross-platforms that supports the implementation of control tasks based on software agents and streaming technologies. This implementation consists of free, off-the-shelf, software components, resulting in a transparent system in which the configuration can be adapted to existing hardware in very different ways without any modification in the source code. On the other hand [43] and [44] proposed Multi-agent systems for translation among heterogeneous service description languages and Multi-agent frameworks for a Multi-agent-oriented office network. Both frameworks aim to enhance communication within their own specific domain.

In another similar work, H. Farooq Ahmad [35] proposed a communication framework for cross-platforms by exploring the basic Agent system's architecture with highlights on Agent communication languages (ACL). The two most accepted Agent communication languages, namely FIPA-ACL and KQML, have briefly been reviewed. This work has proposed a communication framework that provides a dynamic connection between FIPA agents and WSMO (Web Service Modeling Ontology) based semantic Web Services. FIPA is a standard organization for the development of Agent-based technology and defines a range of architectures for Multi-agent systems in order for FIPA agents to integrate with Web Services. Each element of FIPA-MAS must be SOA compliant. On the other hand, WSMO is a language for the semantic markup of grid services and provides a semantic layer over grid resources. In addition, FIPA and WSMO differ in terms of sentence structure, semantics and implementation, which prevent the communication between agents and grid services as shown in Figure 2 [36]. This integration aims to develop SOA compliant FIPA-ACL ontology by merely considering one specific communication issue but not considering a general communication requirement for cross-platforms that needs to develop a generic framework from quality attributes for communication across platforms, operating systems and programming languages.

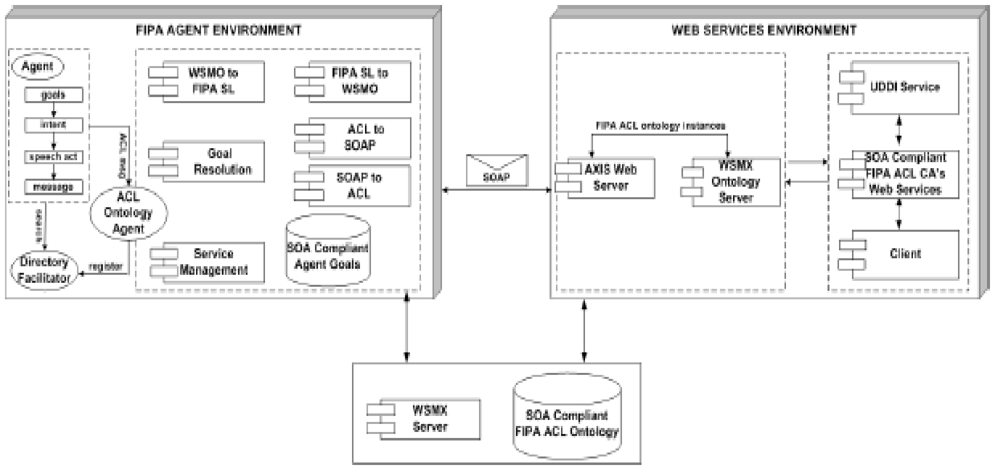


Figure 2. Architectural Realization of the SOA Compliant FIPA ACL [36]

Purvis M. et al. [31] described a framework for building distributed information systems from available resources based on software agents and distributed technologies. They enhanced and adopted an Agent-based architecture by message exchanging via the FIPA Agent communication language (ACL). The architecture was also implemented to grant and open an Agent-based environment for the integration of distributed sources of information over the network.

4. Agent-based Message Oriented Middleware

According to the literature study, most of the current discussions in cross-platform and distributed systems are on middleware technology as an interoperate application because of its properties in cost and time savings (reduce complexity and increase productivity). Even though web services over SOAP are the most accepted communication mechanism for SOA, other communication techniques are used, such as REST and messaging services [5]. Nevertheless, an organisation cannot truly take advantage of an application's benefits without a well-integrated and communicate within a cross-platform and distributed software infrastructure. Middleware enables this integration by sending approaching applications out to cross-platform environments and releasing the domain-specific value of each application [17, 45].

To overcome the weaknesses of multi-domain communication, some researchers have conducted studies to extend the MOM architecture. Most of these efforts have addressed one or two issues of the MOM limitations and have not considered the general requirement for cross-platform communication [1]. Therefore, it is strongly argued that a study in this area is needed to consolidate these distinct issues of cross-platform integration and communication to find a

International Journal of
Soft Computing And Software Engineering (JSCSE)
e-ISSN: 2251-7545
Vol.3,No.11, 2013
Published online: Nov 25, 2013
DOI: 10.7321/jscse.v3.n11.4

common ground towards the establishment of a generic framework for cross-platform communication to be applied within the SOA context. Development of a generic and flexible framework, architectures, tools and platforms to support different applications among cross-platforms and geographically distributed systems would be a good step towards achieving cross-platform communication in SOA systems. The suggested framework is a multi-agent system (MAS) where agents manage all the communication processes among applications. At the same time, there will be a translation agent involved in this process. The general view for the framework is shown in the figure below.

In Figure 3, the cross-platform framework includes four Multi-Agents System (MAS) in each side of application to be used in the communication process between different SOA applications. Included is the Agent Sender, Agent Receiver, Agent Manager and Agent Translator; these multi agent systems represent communication enabled for different SOA applications which use different types of messages in their system communication. At the left side of the diagram is a representation of the SOA-based application 1 and the right side has a representation as the SOA-based application N.

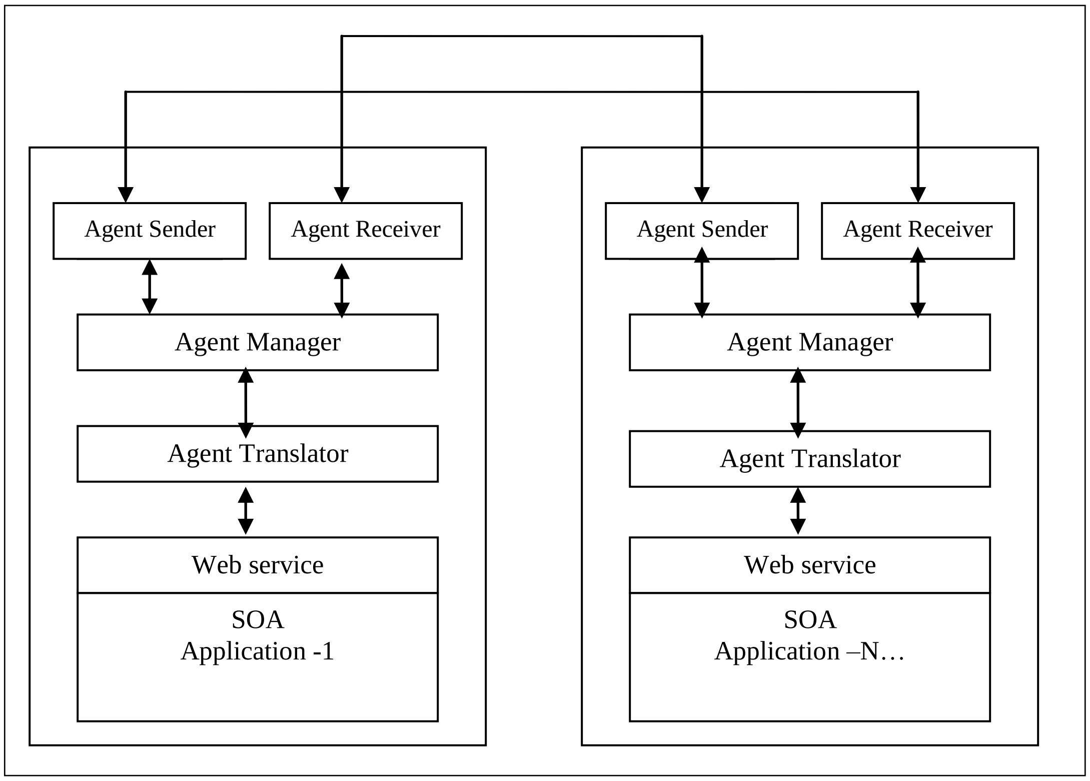


Figure 3. Suggested Frameworks (General view)

In addition, both sides have an agent sender and agent receiver to manage incoming and outgoing messages from each application system. These agents use a message queue to store the sequence of transactions in order to assist both applications even though the partner is in the inactive mode. When one of the systems sends or receives a message, it will put the message into the message queue and the agent will manage the message as to whether the message can be sent to the next process or has to wait for a previous message to complete the process. Those messages will be sent to the agent manager which will analyse each message in terms of failure recovery, guaranteed transmission and scalability. Afterward, the manager will pass the message to the agent translator; the agent has its own specification to translate and mapping the message from the Directory Facility (DF) of the JADE platform where the library files for each Web service message are able to be altered [46, 47]. There are two important tasks for the Agent translator; they are to translate the incoming and outgoing messages. The incoming message is an ACL message from the agent manager which needs to be translated into a WSDL message in order to redescribe it back into their own Web service message; such an outgoing message will be a WSDL message that has been described to the Web service message for its SOA application. The WSDL message needs to be translated into ACL in order to enable the agent to carry and manage the message to the partner’s SOA application. The following will describe the WorkFlow management system for the Agent-based MOM.

4.1 Agent-based MOM Workflow management system (AMWMS)

In general, users are able to describe the interactions among agents by using the agent communication language (ACL) which is the basic standard agent communication proposed by FIPA (Foundation for intelligent Physical Agent) [39]. The workflow system can coordinate and control the interactions among agents which are used to perform tasks, such as message passing and executing tasks. The proposed approach for workflow management in SOA environments is an Agent-based MOM WorkFlow Management System (AMWMS). AMWMS provides high-level and flexible interoperation to enable transparent communication among agents over cross-platforms and distributed systems over a wide-area network.

The basic idea for the AMWMS is to simplify and facilitate the complex environment of cross-platform communication; it consists of a collection of federated servers with a hosting AMWMS conceptual engine in each of the Agent-based MOMs. The partnership of the processing resources which host the AMWMS environment, make their own placement and communication decisions independent of each other. The AMWMS environment provides the necessary framework for the seamless communication and execution of the component parts of the users' requests across the cross-platform system to ensure that the request is fulfilled. The AMWMS architecture has been adopted from a service-oriented perspective with a high degree of automation that supports flexible collaborations and computations on a large complex application as shown in Figure 4. Workflow engines are distributed across an SOA-based application. In this work, the cMOM [48] (Composite-event Based Message-Oriented Middleware) was adopted as the AMWMS engine. The communication and message passing can manage themselves where an AMWMS engine can be interconnected with those SOA services and resources in the engine. AMWMS engines can be dynamically detected and connected into different SOA architectures, such as XML-based applications, SOAP-based applications and CORBA-based applications. Due to the dynamic nature and flexibility of agent-based technology, the AMWMS conceptual engine is suitable for cross-platform communication.

International Journal of
Soft Computing And Software Engineering (JSCSE)
e-ISSN: 2251-7545
Vol.3,No.11, 2013
Published online: Nov 25, 2013

DOI: 10.7321/jscse.v3.n11.4

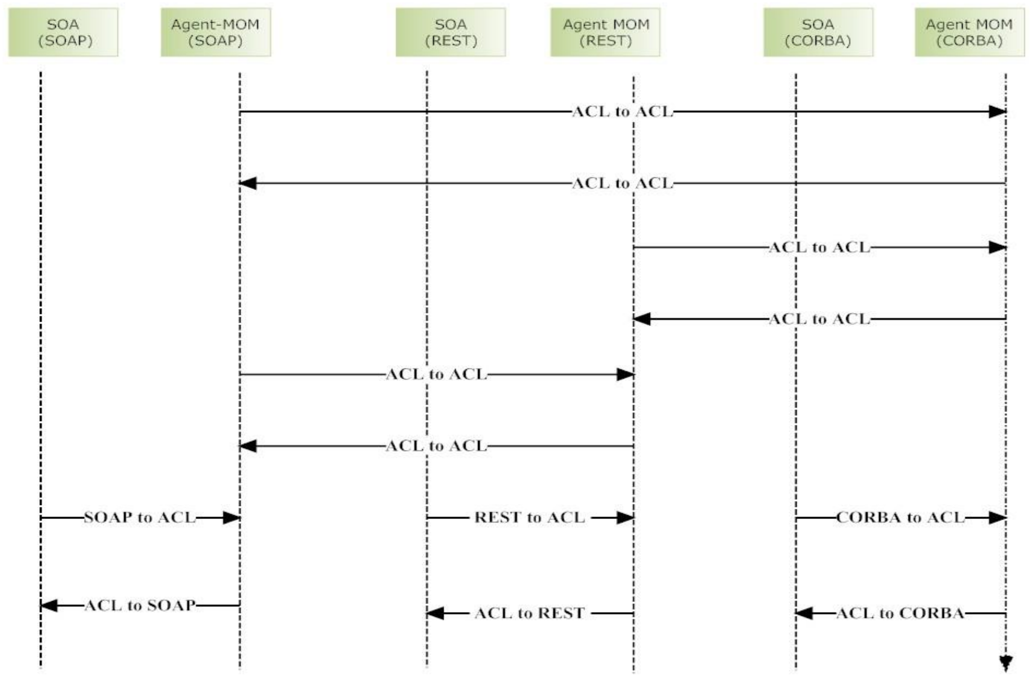


Figure 4. Agent-based MOM Workflow management system (AMWMS)

In addition, this communication workflow decreases the level of complexity in cross-platform communication. In an Agent-based MOM workflow, all components will have two way interactions and the platform will use the standard messaging proposed by FIPA which is ACL (Agent Communication Language), each agent will manage each message priority based on its rules and analyse the action that should be taken. For example, a SOAP-based need to send a message to a CORBA-based application where the manager agent of the Agent-based MOM that is attached with a SOAP-based application will analyse the message that needs to be sent and the agent sender will send the message to receiving agent of the agent-based MOM that is attached with the CORBA-based application. The translator agent in each side will manage the translation process from WSDL to ACL and vice versa [20].

4.2 Agent-based MOM Multi Agent System (ABMMAS)

In the previous section, the proposed framework was presented in an overall view. In this section, the components of the multi-agent systems in the Agent-based MOM framework will be explained in detail. ABMMAS is used as a plug and play system among the different applications in an SOA system to enable communication. Each application of the SOA system needs to be attached with ABMMAS; this includes all incoming/outgoing messages to/from partner application systems. In general, it will translate a WSDL message (web services) from an SOA application to an ACL message. Afterwards, it will analyse the message content of both the incoming and outgoing in the SOA application. Therefore, it can respond with the right action that has been requested. For instance, a SOAP-based application has requested to send a message to a REST-based application; ABMMAS will analyse the request and send the message to ABMMAS of the REST-based application, dynamically. ABMMAS consists of four agents which are the Sender, Receiver, Translator and Manager Agents. The following sub sections will present the roles and functionalities of each agent by AUML [49].

4.2.1 Sender

The sender agent is responsible for sending the ACL message to the agent receiver of the partner Agent-based MOM system. It is able to store incoming messages in case the agent receiver is in the inactive mode. It also collects the basic information directly from the agent manager from the original message of the agent translator that has been translated from the Web service format. The basic information in the Web service is identified as communication directives. The basic information could include sender, receiver, reply to, message content, language, encoding, basic ontology mapping and many others depending on how it is defined in the Web service. The collected information will be stored in a message queue which manages the messages for the next process.

Sender

<u>Description:</u>

This role is sending the outgoing ACL message from the agent manager to the agent receiver of the partner and saving it at the same time in the history.

<u>Protocols and Activities:</u>

GetACL, StoreMessageQueue, SendACL, ActiveDropDelay

<u>Permissions:</u>

Send	ACLMessage
Store	Message
Active	Message Drop / Delay

<u>Responsibilities:</u>

Liveness:

Sender = (GetACL. StoreMessageQueue, SendACL)

ActiveDropDelay = (GetDropDelay, StoreMessage, SendACL)

<u>Safety:</u>

MessageQueue $\neq$ NULL

ACLMessage TRUE

Figure 5. Role Schema for Sender Agent

As shown in Figure 5, this agent is allowed to receive the ACL messages, store them in the message queue, and send the message. The agent is initiated by receiving the ACL message to generate the destination of the receiver and then send the ACL message which is expressed with the following formula:

Sender = (GetACL. StoreMessageQueue. SendACLMessage)

The following is the simple code of JADE to send a message to another agent. The agent needs to understand the message as to where it needs to send the message and what the message content is. Afterwards, it will fill the fields of an ACLMessage object and then call the send () method of the agent class. The code below sends the inform message to an agent, whose nickname is application1, that the product price is 100USD [39].

```
ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
```

International Journal of
Soft Computing And Software Engineering (JSCSE)
e-ISSN: 2251-7545
Vol.3,No.11, 2013
Published online: Nov 25, 2013
DOI: 10.7321/jscse.v3.n11.4

```
msg.addReceiver(new AID("Application1", AID.ISLOCALNAME));
msg.setLanguage("English");
msg.setOntology("Product-price-ontology");
msg.setContent("The price is 100USD");
send(msg);
```

4.2.2 Receiver

The main task for the receiver agent is to receive the message from the agent sender of the partner application. The message will be stored in the message queue before being executed to the next process. The agent sender of the partner application is the only agent able to send the message to this agent. The message queue functionality of this agent is the same as the sender agent. In case the partner system is not available the message can be affected by delays and being dropped so the agent will return the message to the queue which will enable the reprocessing of the message for the next transmission.

Receiver

Description:

This agent receives the message from the sender of the participant application. It needs to forward the message to the agent manager so that it can analyse the message.

Protocols and Activities:

ReceiveACL, StoreMessageQueue, ForwardACL, ActiveDropDelay

Permissions:

Store	Message
Forward	ACLMessage
Active	Message Drop / Delay

Responsibilities:

Liveness:

Receiver = (ReceiveACL, StoreMessage, ForwardACL, ActiveDropDelay)

Safety:

Message $\neq$	NULL
ACLMessage	TRUE

Figure 6. Role Schema for Receiver Agent

Figure 6 shows the role schema of the agent receiver based on the AUML modelling. It shows that the receiver would use four protocols to interact with the other agents. These protocols are (initially) ReceiveACL, StoreMessage, ForwardACL and ActiveDropDelay. Additionally, this agent cannot be initiated unless the message is valid and the rules have been extracted as shown in the Safety section. Furthermore, the agent sender at runtime automatically posts messages in the receiver's private message queue. An agent can pick up messages from its message queue by means of the receive() method. This method returns the first message in the message queue (removes it) or null (does nothing) if the message queue is empty and immediately returns [50].

International Journal of
Soft Computing And Software Engineering (JSCSE)
e-ISSN: 2251-7545
Vol.3,No.11, 2013
Published online: Nov 25, 2013
DOI: 10.7321/jscse.v3.n11.4

```
ACLMessage msg = receive();

if (msg != null) {

// Process the message
```

4.2.3 Manager

The manager agent initiates the process by analysing the message that has been received from both agents which are the translator and receiver agents in different communication procedures. It is to ensure the three main responsibilities of this agent which are Failure Recovery, Guaranteed transmission and Scalability. The manager agent sends the notification when it detects an execution that violates the message content. It will then send the response report to the owner agent of the message. This response consists of three things: the message block that has been violated, the execution that violated the transmission, and the action that should be applied based on the owner of the message. When the manager agent sends the report, it expects that the partner will send a response back to cancel the transmission or resend the violated message again. This is to achieve the three main responsibilities noted above. After the verification process, the management agent will receive the response from the owner of the original message about their decision and it will forward the message to the next agent to process and will save it in the transmission status log file. Therefore, the management agent is the agent to counter some serious communication violations; this is like a manger role in an organization. The functionality of the management agent can be extended to be involved in any Negotiation or Service level agreement (SLA) process which is out of the scope of this study [33, 37, 51].

Manager
Description: The manager agent receives the message from two directions: (1) from the agent receiver that received the message from the agent sender of the partner application and (2) from the agent translator that needs to forward the message to the partner application. The agent manager will manage the message for some purposes, such as guaranteeing message transmission, scalability and failure recovery. **Protocols and Activities:** Initialise, GetMessage, AnalyseMessage, SaveMessage, ForwardMessage **Permissions:** **Reads** Message Content **Saves** Record **Responsibilities:** **Liveness:** Manager = (GetMessage, AnalyseMessage, ForwardMessage) SaveMessage = Analyse, SaveMessage **Safety:** Message $\neq$ NULL Manage Complete

Figure 7. Role Schema for Manager Agent

International Journal of
Soft Computing And Software Engineering (JSCSE)
e-ISSN: 2251-7545
Vol.3,No.11, 2013
Published online: Nov 25, 2013
DOI: 10.7321/jscse.v3.n11.4

In addition, Figure 7 shows the role schema of the Manager Agent based on the AUML modelling. It shows that the manager would use four protocols to interact with the other agents. These protocols are (initially) GetMessage, AnalyseMessage, SaveMessage and ForwardMessage. This agent also cannot be initiated unless the message is valid and the rules have been extracted as shown in the Safety section.

4.2.4 Translator

The message mapping definitions obtained from the WSIG (Web Service Integration Gateway) will be used by the translator agent to evaluate and match the message content for the translation process based on the mapping library. The translator has the role to translate the message from WSDL to ACL and vice versa. The message from the attached SOA application that has been described by the standard Web service format which is the WSDL (Web Service Description Language) will be translated into the ACL (Agent Communication Language). Therefore, ACL is the standard agent communication language proposed by FIPA that will be the generic language in the proposed communication method. This is to support the different types of SOA applications that may be included in the future where there is a standard communication language and standard Web service description. Furthermore, the ACL message from the agent manager will be translated into WSDL for matching with its own Web service format. In the case study-based implementation, the authors evaluated only three different types of messages which were REST, SOAP and CORBA. For other types of messages will be included in a future study [33, 38, 46]

Translator

Description:

The translator agent gets the messages from two sides, from the agent manager and from the attached SOA application. Then, it checks the content of each message and responds to the message request.

Protocols & Activities:

GetMessage, ReadMessage, Analyse, TranslateMessage, ForwardMessage

Permissions:

Reads	ACL, Request
Translates	ACL, WSDL

Responsibilities:

Liveness:

Translator = (GetMessage. ReadMessage. AnalyseContent. ForwardMessage)

TranslateMessage = (GetMessage. Analyse. Translate)

Safety:

Message $\neq$ NULL

Translation Complete

Figure 8. Role Schema for Translator Agent

International Journal of
Soft Computing And Software Engineering (JSCSE)
e-ISSN: 2251-7545
Vol.3,No.11, 2013
Published online: Nov 25, 2013
DOI: 10.7321/jscse.v3.n11.4

The role schema of this agent is shown in Figure 8. It states that this agent executes two processes which are analysing the messages and translating the messages which is explained in the Liveness section as the following: In the next section, an implementation and validation of the proposed framework will be presented in detail.

Translator= GetMessage.ReadMessage.AnalyseContent.ForwardMessage)
TranslateMessage = (GetMessage. Analyse. Translate)

5. Implementation and Validation

Web services have been accepted for enabling cross-platform communication among different SOA applications. Additionally, Agent technology can be described as autonomous, adaptive and intelligent software. It is a computer program which responds to the communication act. Therefore, interaction among applications is carried out automatically without any control from the user in certain periods of execution. There are valid significant needs of agent technology to access and translate a Web service. As a result, integration of Web services and software agents provide several advantages in enabling the communication within a cross-platform environment. In general, an agent will access a Web service to read the content and convert it into the agent supported format. Consequently, with the independent and adaptive attributes of agent software, it will support multi-domain communication[52, 53].

In addition, Web service is a piece of software that makes a service available through distributed applications. The objective is to develop a web service where the components can be used and reused by other applications. Currently, many research works have extended Web technology to support cross-platform applications. Some have adopted agent-based technology to facilitate the communication process. Therefore, integration among Web services and agent software is challenging [54]. In this thesis, an extension of the Message Oriented Middleware with Agent technology for the cross-platform environment of an SOA system has been proposed. Generally, an SOA-based application uses the Web service for their communication. However, most of the existing solutions of integration look at only a single dimension. For example, WS2JADE aims to convert SOAP messages into Java messages (ACL) and vice versa but is not able to convert others type of Web service messages [46]. Thus, this proposed framework also consists of an Agent translation model that automatically translates the Web service into a supported message of the partner application. It also supports multiple types of Web services which are based on the DF library file in the JADE Platform. In the next section the Implementation Hardware and Software Setup will be presented [55, 56].

5.1 Implementation Hardware and Software Setup

The case study used in all the experiments is composed of three communication experiments for different SOA-based applications. Each application is connected with the Agent-based MOM (ABMOM) to facilitate the communication among those that are SOAP-based applications, CORBA-based applications and REST-based applications. Those SOA-based applications represent the different partners of the SOA-based applications that are required to communicate with each other. In addition, each of SOA-based applications can be represented as Buyers or Seller in the case study to evaluate cross-platform communication. However, due to limitation of prototype, only one SOA-based application can be represented as seller at a time. The ABMOM was installed into every systems of the SOA application as a plug and play application. Then, some task was allocated to each SOA-based application to communicate with each other. The original task was from the SOA-based application that was attached with ABMOM. The request was in the Web Service format that would be described by WSDL. Afterwards, the Multi agent system of ABMOM executed a task based on each role of the agent.

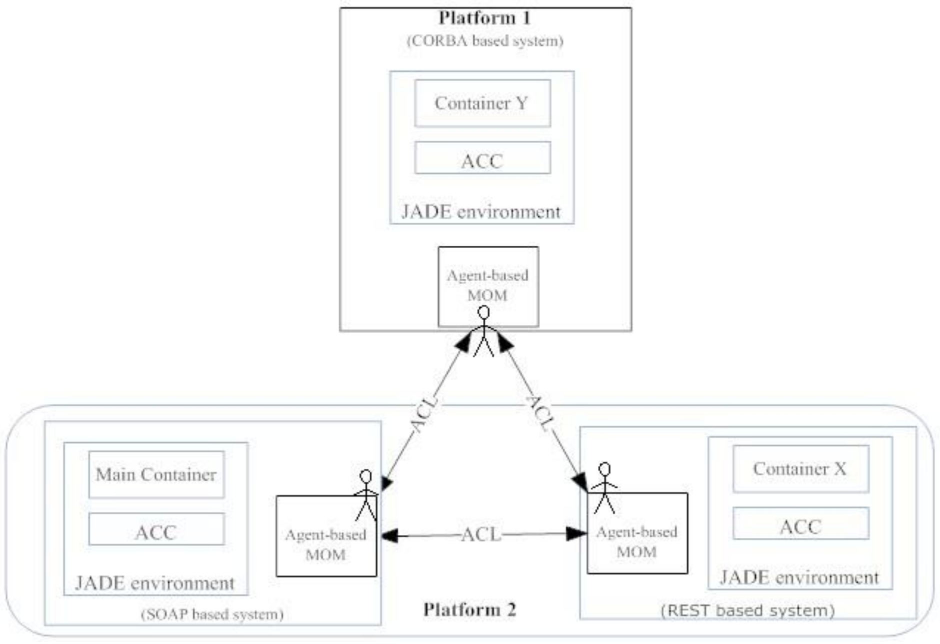


Figure 9. Simulation Metric

As shown in Figure 9, ABMOM resides in the SOAP-based application, CORBA-based application, and REST-based application. All applications run the HTTP server, JADE environment and Apache Server. Three important elements are present in the FIPA compliant platform. The Agent Management System (AMS) controls the access of the platform and the Directory Facility (DF). DF provides a library service and Agent Communication Channel (ACC) that facilitates the message transport service for FIPA ACL message delivery among agents living in different agent platforms [47, 57]. In the case study, there are two different platforms to evaluate the communication performance of ABMOM. The platform details are presented in Tables 1 and 2.

Table 1. Platform number I

Model	Dell Inspiron Desktops
Processor	CORE i3 @ 2.66GHz
Total memory	4.GB
Operating system	Window 7 Professional
OS version	OA SEA
Java	Sun SDK 1.5
JADE	3.7

Table 2. Platform number II

Model	BenQ Joybook S31v (Notebook)
Processor	Intel® Core™2 CPU T5500 @ 1.66GHz

International Journal of
Soft Computing And Software Engineering (JSCSE)
e-ISSN: 2251-7545
Vol.3,No.11, 2013
Published online: Nov 25, 2013
DOI: 10.7321/jscse.v3.n11.4

Total memory	3.GB
Operating system	Window XP
OS version	SP2
Java	Sun SDK 1.5
JADE	3.7

5.2 Prototype Implementation of Agent-based MOM

During this section, a detailed discussion is presented regarding the prototype implementation of the Agent-based MOM for cross-platform communication among different SOA-based applications.

5.2.1 Tools and Technologies Used

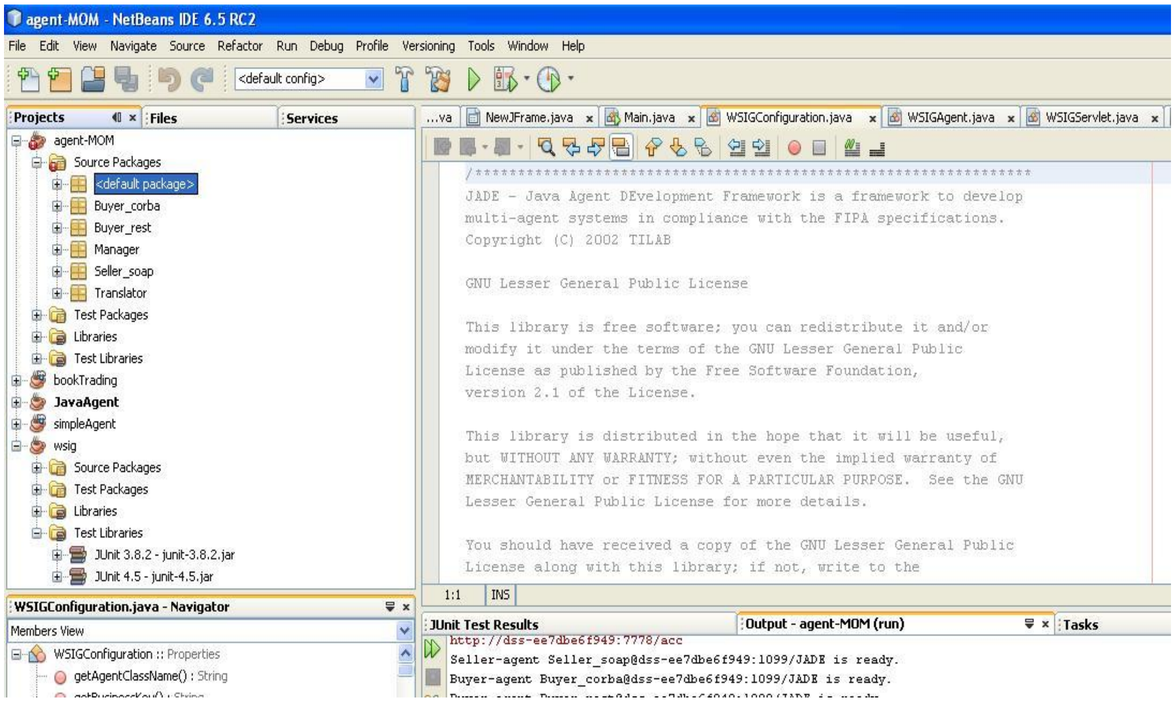
Following is a brief description of the tools and technologies used in the implementation process.

5.2.1.1 NetBeans IDE 6.5 RC2

The NetBeans IDE is an Integrated Development Environment available for Windows, Mac, Linux, and Solaris. In the java community, two particular client platforms play a fundamental role, namely the NetBeans Platform and the Eclipse Client Platform. “NetBeans IDE allows a user to easily reload the Java Model in an instance of a running program” [58]. The NetBeans project consists of an Open Source IDE and an application platform which enable developers to rapidly develop Web, enterprise, desktop, and mobile applications using the Java platform, as well as PHP, JavaScript and Ajax, Ruby and Ruby on Rails, Groovy, and C/C++. It is supported by a vibrant development community and offers a diverse selection of third-party plug-ins as shown in Figure 10. In this work, multiple Web services have been integrated with the agent software in NetBeans IDE which includes several third-party plug-ins, i.e., Web Service Integration Gateway (WSIG).

International Journal of
Soft Computing And Software Engineering (JSCSE)
e-ISSN: 2251-7545
Vol.3,No.11, 2013
Published online: Nov 25, 2013

DOI: 10.7321/jscse.v3.n11.4



```
/*****************************************************************
JADE - Java Agent DEvelopment Framework is a framework to develop
multi-agent systems in compliance with the FIPA specifications.
Copyright (C) 2002 TILAB

GNU Lesser General Public License

This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation,
version 2.1 of the License.

This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public
License along with this library; if not, write to the
```

```
http://dss-ee7dbe6f949:7778/acc
Seller-agent Seller_soap@dss-ee7dbe6f949:1099/JADE is ready.
Buyer-agent Buyer_corba@dss-ee7dbe6f949:1099/JADE is ready.
```

Figure 10. NetBeans Platform

5.2.1.2 Java Agent Development Environment (JADE)

As shown in Chapter 3, each component in the proposed framework is represented as a Multi Agent System (MAS). The JADE Platform was used that facilitates the development of the multi-agents systems. The JADE version that was used is JADE 3.7, JADE includes two main attributes which are a FIPA-compliant agent platform and a package to develop Java agents. It is made of numerous Java packages which provide application programmers with both ready-made pieces of functionality and abstract interfaces for custom applications [50]. JADE is also completely implemented in the Java language and the average system requirement is version 1.5 of the JAVA run time environment which is a free download from Sun Corporation [57]. Each agent executes several different behaviours as defined in the agents' specification. The path of each agent execution developed in JADE is shown in Figure 11.

International Journal of
Soft Computing And Software Engineering (JSCSE)
e-ISSN: 2251-7545
Vol.3,No.11, 2013
Published online: Nov 25, 2013
DOI: 10.7321/jscse.v3.n11.4

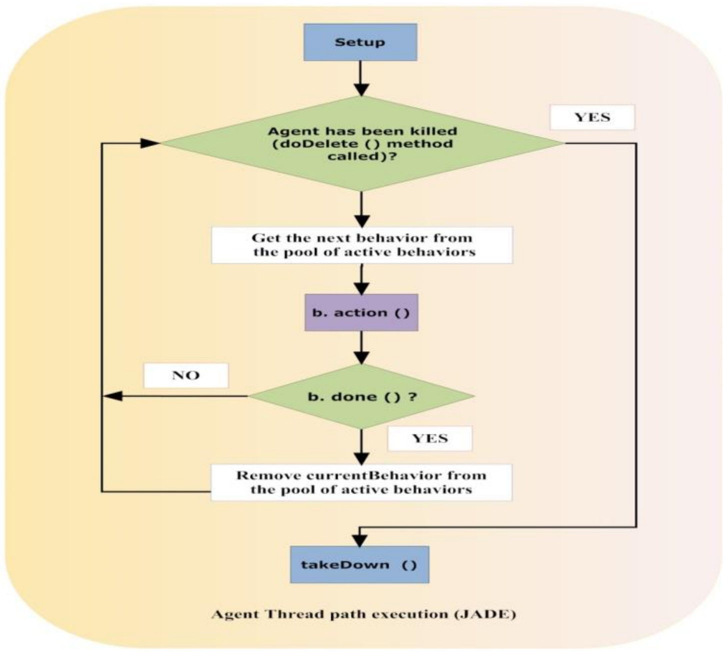

Setup
Agent has been killed (doDelete () method called)?
YES
Get the next behavior from the pool of active behaviors
b. action ()
NO
b. done () ?
YES
Remove currentBehavior from the pool of active behaviors
takeDown ()
Agent Thread path execution (JADE)

Figure 11. Agent Execution Path [50]

In addition, there are five states of an agent after it has been created which are active, waiting, suspended, transited and killed. Each agent has its own thread, thus all of the agents independently execute their behaviours in parallel. JADE architecture enables agent communication through message exchange based on the FIPA specifications of the agent communication language (ACL). Agents in the JADE environment can work collectively to achieve common objectives by coordination towards a better process. Thus, JADE simplifies the implementation of multi-agent systems. Every machine in the proposed prototype has its own JADE agent container built on its Java Virtual Machine (JVM) in order to manage the agents for specific behavioural mechanisms. The JVM provides a complete run time environment for agent execution and allows agents to be executed on the same host, concurrently. Every JADE agent container will register itself to the Remote Method Invocation which allows agents from different machines to communicate with each other [50, 57].

5.2.1.3 Web Service Integration Gateway (WSIG)

Web Service Integration Gateway is an add-on component that supports the invocation of a JADE agent and provides an integration facility for the Web service and the Agent software. WSIG is used to expose services provided by agents and published in the JADE DF library as Web services and vice versa, with no or minimal additional effort. It provides developers with flexibility and autonomy to meet specific requirements. The process involves the generation of suitable WSDL for each Web service description and is registered with the DF, then it possibly publishes the exposed service in a UDDI register. It also enables the translation of the Agent Communication Language (ACL) into WSDL and vice versa [59].

5.2.1.3.1 WSIG Architecture

WSIG supports the integration of the Web service and agent software which consist of WSDL to describe Web services in a standard format. It supports SOAP/HTTP messages for transmission and UDDI repository for publishing Web services using tModels. A tModel is a data structure representing

International Journal of
Soft Computing And Software Engineering (JSCSE)
e-ISSN: 2251-7545
Vol.3,No.11, 2013
Published online: Nov 25, 2013
DOI: 10.7321/jscse.v3.n11.4

a service type (a generic representation of a registered service) in the UDDI (Universal Description, Discovery, and Integration) registry. As shown in Figure 12 WSIG is represented as a web component including two main attributes, WSIG Servlet and WSIG Agent. The WSIG Servlet is an interface towards other applications and is responsible for incoming Web service requests and extracting the message. It prepares the corresponding agent action and passed it to the WSIG agent for preparing the response to the client. On the other hand, the WSIG agent can be described as the gateway between the Web and agent software. It is responsible for forwarding agent actions that are received from the WSIG Servlet and subscribing to the JADE DF to receive notifications. This is for the agent registration and deregistration that created the WSDL corresponding to each agent service registered with DF and published the service in a UDDI registry [46, 47, 59].

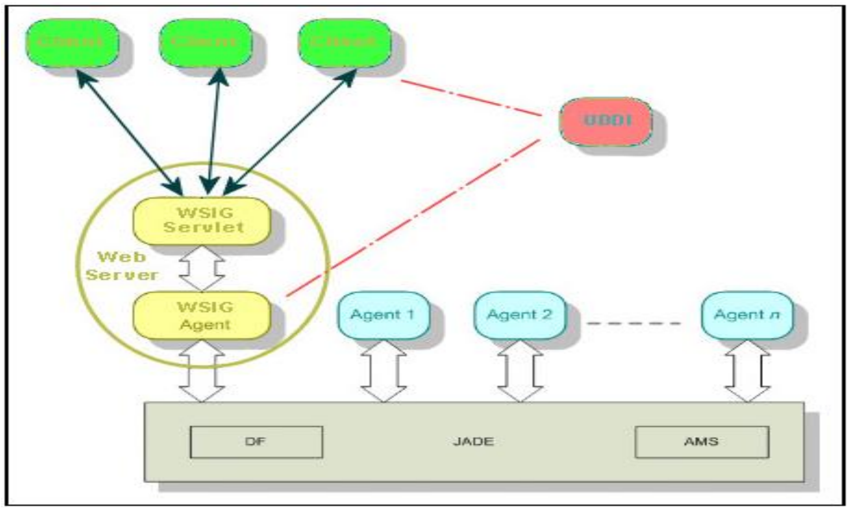


Figure 12. WSIG Architecture[59]

5.2.1.4 Web service Ontology (OWL-S)

OWL-S is a Web service ontology which contributes a core set of markup language constructs for describing the attributes and capabilities of their Web services in an unambiguous and computer-interpretable form. The OWL-S markup of Web services will facilitate the automation of Web service tasks, including automated Web service discovery, execution, composition and interoperation. Currently, the Web Service Description Language is rapidly growing to provide a foundation for interoperation among Web services. Therefore, OWL-S is developing to provide integration among them and the agent technology. It is the flexible automation of service provision and provides significant methodologies [60]. In addition, OWL-S is an ontology of the OWL-based framework in the Semantic Web, for describing Semantic Web Services. “It will enable users and software agents to automatically translate, discover, invoke, compose, and monitor Web resources offering services, under specified constraints” [61].

5.2.1.5 Universal Description, Discovery and Integration (UDDI)

UDDI is a platform independent framework for describing services which is an XML-based registry where agents publish their WSDL documents. It is a public Web service registry standard hosted in a third party entity for service description and discovery. It is also an open industry initiative for

International Journal of
Soft Computing And Software Engineering (JSCSE)
e-ISSN: 2251-7545
Vol.3,No.11, 2013
Published online: Nov 25, 2013
DOI: 10.7321/jscse.v3.n11.4

discovering businesses, and integrating business services by using the Internet. Furthermore, UDDI is sponsored by the Organization for the Advancement of Structured Information Standards (OASIS) which is a global USA consortium for maintaining Web service standards. UDDI was originally proposed as a core Web service standard. It stores Web service business information in three forms: white, yellow and green pages. White pages hold Web services general information like name, address and description. Yellow pages allocate different business classifications according to the type of Web service or geographical location. Green pages contain service technical capability information that consumers use to invoke the Web service [62, 63].

5.2.1.6 AUML Development Diagram

The Agent Unified Modelling Language (AUML) is an agent software paradigm which is extended from the Unified Modeling Language (UML) proposed as a standard by the Foundation for Intelligent Physical Agents (FIPA). AUML is a well-established and trusted method. It is “a widely accepted methodology for designing software systems according to the object-oriented paradigm” [64]. AUML also “crystallizes a growing concern for agent-based modelling representations and lets designers move smoothly from software development to agent development”[65]. AUML used to enable agent oriented programming (AOP) for the efficiency of implementation. Currently, AUML has become significantly important for object oriented programming that has encouraged researchers to enhance it to support agent-based programming.

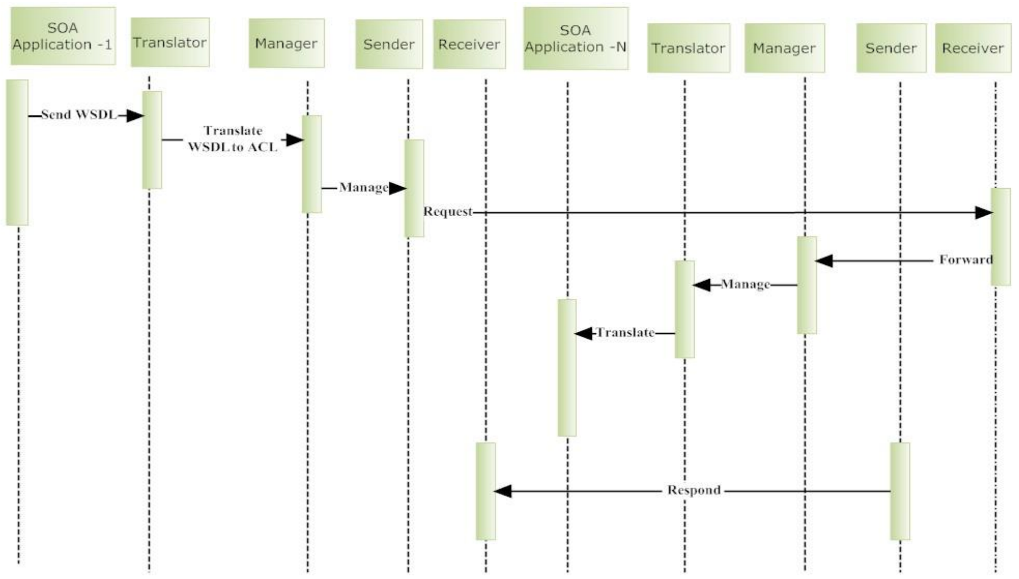


Figure 13. AUML Sequence Diagram for the Agent-based MOM

In Figure 13, the AUML Sequence diagram for the Agent-based MOM is illustrated in detail. The sequence diagram starts with SOA application-1 sending the Web service to the agent translator that has been described by WSDL and then translates it into ACL and forwards it to the agent manager. After that, the message will be forwarded to the agent sender so that the message can be sent to the agent receiver of the SOA application-N. Afterwards, the ACL message will be forwarded to the agent manager and agent translator to translate back the ACL message into the supported Web service format of SOA application-N. In addition, SOA application-N represents different applications of the SOA system which could be supported by different types of Web service technologies in the future.

6. Evaluation and Validation

The scenarios of SOA cross-platform communication are investigated and found that the significant cross-platform attributes are not included during the early stage of development of SOA applications because of two main reasons. Firstly, there is no clear identification of the generic attributes for the cross-platform communication in SOA systems. Secondly, the current existing cross-platform frameworks are problem specific of which the main goal of communication and integration is to resolve their specific problem.

There should be a formal means through which a generic cross-platform framework would be modelled and developed within the SOA environment. Having this very essential in mind The "Agent-based MOM" has been developed. The attributes for cross-platform communication have been identified and among them the most essential cross-platform attributes were picked which were necessary for constructing the proposed framework. An easyABMS was defined illustrating the cross-platform attributes and the attribute mechanisms through which these cross-platforms attributes would be realized. Afterwards, the AUML modelling technique was used for the definition of these cross-platform attributes as an activities diagram in AUML.

The domain experts in each area of the system development, who are experts only in their particular cross-platform problem, will only model and develop the framework to resolve their specific issues. Thus, there should be a comprehensive consolidated research study to develop a generic cross-platform communication framework for different SOA applications.

6.1 Comparison of "Agent-based MOM" With Related Work

Keeping in mind the guidelines discussed by Chituc, C.-M., A.R. Azevedo, and S. Toscano [29], the significant attributes for a cross-platform and distributed system have been defined. The comparision table represents the related work regarding the cross-platform communication in an SOA application. Out of the sixteen most significant attributes for a cross-platform, a generalization has been made of some overlapped attributes to include in this proposed framework. Table 3 presented research work which is very close to this proposed framework and illustrated the details in the discussion which is provided below.

1. *Standardization and Autonomy*: A cross-platform communication framework should be generic and autonomic, it should not be meant for only a particular application domain. Standardization means generic and accepted by multiple domains to overcome the issues whereas autonomic means the level of flexibility to execute the task.

WS2JADE, the IEEE FIPA Approach to Integrating Software Agents and Web services, and the SOA Compliant FIPA ACL are [33, 36, 38, 46] presented as an integration and communication among the different Web service applications. The frameworks constructed under their works concentrate only to integrate and enable the communication between the SOAP Web service protocol and the software agent which are not considered in different types of Web service technologies. An Agent Platform for Reliable Asynchronous Distributed applications [34] and An Agent XML based Information Integration Platform [41] also use the same approach to resolve only their specific problem. Yoe Jin Yoon et al. [43] have extended the Communication System among Heterogeneous Multi-Agent Systems which does also not include the requirement attributes for a generic and standard cross-platform framework.

As compared to some other works, the proposed Agent-based MOM for cross-platform communication is more generic and flexible which enables it to support multiple application domains. Its main goal is not to resolve a particular integration and communication concern instead it looks at the general requirements of the cross-platform to develop the framework.

International Journal of
Soft Computing And Software Engineering (JSCSE)
e-ISSN: 2251-7545
Vol.3,No.11, 2013
Published online: Nov 25, 2013
DOI: 10.7321/jscse.v3.n11.4

2. *Communication type and Message type*: The cross-platform communication framework will be flexible in communication and use a standard messaging system. The flexibility of the communication refers to communication that proceeds without any response from a partner and is able to communicate even though the partner application is in the inactive mode. The standard message type can be described as the message that has been accepted by a recognized organization. Therefore, the cross-platform communication should not adapt only some specific technique or technology to decipher a particular problem.

No	Works	Comparison of the proposed work by different Researchers							
		Communication type	Availability	Autonomic level	Message type	Software failure recovery	Guaranteed transmission	Scalability	Standardization
1	An Agent Platform for Reliable Asynchronous Distributed [34]	Asynchronous	Active / Non-active mode	No	ACL	Yes	No	No	No
2	Agent-Based Middleware for Web Service Dynamic [33]	Synchronous	Active mode	No	WSDL	No	No	Yes	No
3	WS2JADE [46]	Asynchronous	Active / Non-active mode	Yes	SOAP/ ACL	No	Yes	No	Yes
4	IEEE FIPA Approach to Integrating Software Agents and Web Services [38]	Asynchronous	Active / Non-active mode	No	SOAP / ACL	Yes	No	No	Yes
5	An Agent XML based Information Integration Platform [41]	Synchronous	Active Mode	No	SOAP	No	No	No	No
6	A Cross-Platform Agent-based Implementation [42]	Synchronous	Active mode	No	ACL	Yes	No	Yes	No
7	Communication System among Heterogeneous Multi -Agent Systems [43]	Synchronous	Active mode	No	ACL	No	Yes	Yes	Yes
8	Multi-agent System for Distributed environments [35]	Synchronous	Active mode	No	ACL / KQML	No	Yes	Yes	No
9	SOA Compliant FIPA Agent Communication Language [36]	Asynchronous	Active / Non-active mode	Yes	SOAP/ ACL	No	No	Yes	Yes
10	An Agent-Based Distributed Information System Architecture [31]	Synchronous	Active mode	No	ACL	No	No	No	No
11	Proposed Agent-based MOM	Asynchronous	Active / Non-active mode	Yes	WSDL/ ACL/ SOAP	Yes	Yes	Yes	Yes

Table 3. Comparison of the proposed work by different Researchers

The framework constructed under Agent-Based Middleware presented by Aizhong Lin and Piyush Maheshwari [33] used the synchronous type of communication where every sent message requires a response from a partner application before proceeding to the next message. Xiangyu Li [41], Enric Cervera [42], and Bo Chen et al. [32] also used the same approach in their work for a communication system. Martin Purvis et al. [31] have extended the Agent-Based Distributed Information System Architecture for cross-platform communication; however, the constructed framework also used the synchronous type of communication.

As compared to some other works, this proposed cross-platform communication framework has adapted the requirement attributes for a generic and flexible communication which uses the Asynchronous type of communication and supports multi-domains.

Table 3 Comparison of Agent-based MOM with Related Works Based on Cross-platforms

3. *Availability and Scalability*: SOA applications are basically a distributed application that requires the availability and scalability of both partner applications. Only sending and receiving methods of applications are not guaranteed of a successful communication. Availability and Scalability are significantly important for cross-platform communication where different applications may have different system environments and requirements that require them to communicate and execute the tasks. Therefore, it is essential for the system to communicate any time whether the partner is in the active or inactive mode. WS2JADE and the IEEE FIPA Approach to Integrating Software Agents and Web Services [38, 46] have been used in some of the previously proposed frameworks with a flexibility of communication; however, they still lack the scalability that is required for the system to handle a growing number of tasks. In the rest of the previously proposed frameworks, the significance of availability and scalability were not considered for their systems which are the basic requirement attributes for cross-platform communication.

In this proposed framework, eight requirement cross-platform attributes were identified which are essential to develop a cross-platform communication framework for different SOA-based applications.

4. *Failure recovery and Guaranteed transmission*: A cross-platform communication framework is a generic and flexible system which should consist of essential attributes when constructed. System recovery and the guarantee of transmission are also significant attributes for cross-platform over a large area of a network. Hence, the system will be able to recover from any communication failure and ensure the message transmission among systems.

In most of the previously proposed frameworks, there is a lack of consideration to include software failure recovery and guaranteed transmission for their works [31, 33, 34, 41]. Therefore, they are unable to roll back the transmission for a recovery and provide a successful communication system. As a result, in this proposed framework, these attributes have been considered as essential mechanisms to construct the framework.

7. Conclusion

In this research, an Agent-based Message Oriented Middleware has been suggested to ensure the communication among different SOA-based applications. In order to validate and evaluate the system performance and its effectiveness, a system experiential test had to be executed. The experiential test was challenging based on a lot of extended solutions being invented in the cross-platform with different particular perspectives. Most of the research works in literature were proposed to solve some particular issue which would fix their specific problem. Therefore, three dimension case study implementation was conducted to prove how generic and flexible the proposed cross-platform communication framework. In the proposed solution, the concentration was on generic and autonomic

International Journal of
Soft Computing And Software Engineering (JSCSE)
e-ISSN: 2251-7545
Vol.3,No.11, 2013
Published online: Nov 25, 2013
DOI: 10.7321/jscse.v3.n11.4

cross-platform communication which would be able to communicate among more than two different SOA-based applications.

In addition, according to the literature review, it was shown that there are no exact similar studies within the same scope of cross-platform communication in an SOA system which conducted a comparative study. Therefore, it was necessary to take other similar studies from different domains of research, such as Grid Interoperability Solution[66], Interoperability between Heterogeneous Multi-Agent Systems[44], A Cross-Platform Agent-based Implementation[42] and so on. The approach to evaluate the proposed framework would be to have different SOA-based applications that need to communicate with each other and then the framework would be applied in that environment. However, such an approach is out of the scope for this research work that involves heavy and complex message content. In the experiential testing, the framework was implemented with the implementation of the case study presented. Currently, we are working to include more difference message types into the framework which will support more different types of SOA application for cross-platform communication.

References

1. KANCHANAVIPU, K., An Integrated Model for SOA Governance, in Applied Information Technology. 2008, IT University of Göteborg Chalmers University of Technology and University of Gothenburg: Göteborg.
2. Hentrich, C. and U. Zdun, A Pattern Language for Process Execution and Integration Design in Service-Oriented Architectures, in TPLOP I. 2009, Springer: Verlag Berlin Heidelberg. p. 136 – 191.
3. Papazoglou, M.P., et al., Service-Oriented Computing:State of the Art and Research Challenges. 2007, IEEE.
4. TEO, L.K.Y., D.W. TEH, and B. CORBITT, SERVICE ORIENTED ARCHITECTURE (SOA): IMPLICATIONS FOR AUSTRALIAN UNIVERSITY INFORMATION SYSTEMS CURRICULUM. 2006, IEEE.
5. Srinivasan, L. and J. Treadwell, An Overview of Service-oriented Architecture, Web Services and Grid Computing. 2005.
6. Louridas, P., SOAP and Web Services, in the IEEE Computer Societ. 2006, IEEE.
7. Laitkorpi, M., J. Koskinen, and a.T. Syst¨, A UML-based Approach for Abstracting Application Interfaces to REST-like Services, in Working Conference on Reverse Engineering. 2006, IEEE.
8. Dodero, J.M. and E. Ghiglione, ReST-Based Web Access to Learning Design Services, in TRANSACTIONS ON LEARNING TECHNOLOGIES. 2008, IEEE.
9. Feng, X., J. Shen, and Y. Fan, REST：An Alternative to RPC for Web Services Architecture, in First International Conference on Future Information Networks. 2009, IEEE.
10. Gang, Y., A Research on Semantic Geospatial Web Service Based REST, in International Forum on Computer Science-Technology and Applications. 2009, IEEE.
11. Song, Y., K. Xu, and K. Liu, Research on Web Instant Messaging Using REST Web Service. 2010, IEEE.
12. Glover, A., ed. Build a RESTful Web service. 2008, IBM.
13. Yanga, Q.Z. and Y. Zhangb, Semantic interoperability in building design: Methods and tools. Computer-Aided Design, 2006. **38**.
14. Hughes, A., Middleware for managing a large, heterogeneous programmable network. BT Technology Journal, 2002. **Vol 20 No 4**.
15. Chan, L., S. Karunasekera, and A. Harwood, Middleware for Complex Service-Oriented Peer-to-Peer Applications, in MW4SOC '07. 2007, ACM: Newport Beach, CA, USA.

16. Kuppuraju, S., A. Kumar, and G.P. Kumari, Case Study to Verify the Interoperability of a Service Oriented Architecture Stack, in International Conference on Services Computing. 2007, IEEE: India.
17. Issarny, V., M. Caporuscio, and N. Georgantas, A Perspective on the Future of Middleware-based Software Engineering, in Future of Software Engineering. 2007, IEEE.
18. Happe, J., H. Friedrich, and S. Becker, A Pattern-Based Performance Completion for Message-Oriented Middleware, in WOSP'08. 2008, ACM: Princeton, New Jersey, USA.
19. Ahn, S. and K. Chong, A Case Study on Message-Oriented Middleware for Heterogeneous Sensor Networks, in IFIP International Federation for Information Processing. 2006, IFIP. p. 945 – 955,.
20. Banavar, G., et al., A Case for Message Oriented Middleware, in Verlag Berlin Heidelberg, Springer, Editor. 1999, Springer: Berlin p. 1–17.
21. Goel, S., H. Shada, and D. Taniar, Asynchronous Messaging Using Message-Oriented-Middleware, in IDEAL 2003. 2003, Springer: Verlag Berlin Heidelberg. p. 1118-1122.
22. Yang, H., et al., Message-Oriented Middleware with QoS Awareness, in ICSOC-ServiceWave 2009. 2009, Springer: Verlag Berlin Heidelberg. p. 331–345.
23. Sachs, K., S. Kounev, and S. Appel, Benchmarking of Message-Oriented Middleware, in DEBS'09. 2009, ACM: Nashville, TN, USA.
24. Xu, Y.-z., D.-x. Liu, and F. Huang, Design and Implementation of a Workflow-Based Message-Oriented Middleware, in APWeb Workshops. 2006, Springer: Verlag Berlin Heidelberg. p. 842 – 845.
25. Maheshwari, P., T.N. Kien, and A. Erradi, QoS-Based Message-Oriented Middleware for Web Services, in WISE 2004 Workshops, LNCS 3307. 2004, Springer: Verlag Berlin Heidelberg. p. 241–251.
26. Taton, C., et al., Self-optimization of Clustered Message-Oriented Middleware, in OTM 2007, LNCS 4803. 2007, Springer: Verlag Berlin Heidelberg. p. 540–557.
27. Wang, J. and J. Bigham, Anomaly Detection in the Case of Message Oriented Middleware, in MidSec'08. 2008, ACM: Leuven, Belgium.
28. Elmroth, E., F. Hernández, and J. Tordsson, Three fundamental dimensions of scientific workflow interoperability: Model of computation, language, and execution environment. Future Generation Computer Systems, 2009.
29. Chituc, C.-M., A.r. Azevedo, and S. Toscano, A framework proposal for seamless interoperability in a collaborative networked environment. Computers in Industry, 2009: p. 22.
30. M.Ibrahim, N.b. and M.F.b. Hassan, A Survey on Different Interoperability frameworks of SOA Systems Towards Seamless Interoperability, in ITsim '10. 2010, IEEE: Kuala Lumpur, KLCC.
31. Purvis, M., et al., an Agent-Based Distributed Information Systems Architecture, in Proceedings of the 33rd Hawaii International Conference on System Sciences. 2000, IEEE.
32. Chen, B., H.H. Cheng, and J. Palen, Integrating mobile agent technology with multi-agent systems for distributed traffic detection and management systems. Transportation Research Part C, 2009. **17** p. 1–10.
33. Lin, A. and P. Maheshwari, Agent-Based Middleware for Web Service Dynamic Integration on Peer-to-Peer Networks, in AI 2005, LNAI 3809. 2005, Springer: Verlag Berlin Heidelberg. p. 405 – 414.
34. Bellissard, L., et al., An Agent Platform for Reliable Asynchronous Distributed Programming. 1999, IEEE: FRANCE.
35. Ahmad, H.F., Multi-agent Systems: Overview of a New Paradigm for Distributed Systems, in HASE'02. 2002, IEEE.
36. Raja, M.A.N., H.F. Ahmad, and H. Suguri, SOA Compliant FIPA Agent Communication Language. 2008, IEEE.

International Journal of
Soft Computing And Software Engineering (JSCSE)
e-ISSN: 2251-7545
Vol.3,No.11, 2013
Published online: Nov 25, 2013
DOI: 10.7321/jscse.v3.n11.4

37. Poggi, A., M. Tomaiuolo, and P. Turci, An Agent-Based Service Oriented Architecture. 2007, IEEE.
38. Greenwood, D., et al., The IEEE FIPA Approach to Integrating Software Agents and Web Services, in 2007 IFAAMAS. 2007: USA.
39. Feng, Q. and G. Lu, FIPA-ACL Based Agent Communications in Plant Automation. 2003, IEEE.
40. CHUSHO, T. and K. FUJIWARA, A Form-based Agent Communication Language for Enduser-Initiative Agent-Based Application Development. 2000, IEEE.
41. Li, X., An Agent/XML based Information Integration Platform for Process Industry, in 2010 2nd International Conference on Computer Engineering and Technology. 2010, IEEE.
42. Cervera, E., A Cross-Platform Agent-based Implementation. 2005, IEEE.
43. Yoon, Y.J., K.H. Choi, and D.R. Shin, Design and Implementation of Communication System among Heterogeneous Multi-Agent System, in Fourth International Conference on Networked Computing and Advanced Information Management. 2008, IEEE.
44. Suguri, H., et al., Assuring Interoperability between Heterogeneous Multi-Agent Systems with a Gateway Agent, in Proceedings of the 7th IEEE International Symposium on High Assurance Systems Engineering (HASE'02). 2002, IEEE
45. Brown, J.A., Middleware Interoperability in SOA Applications. 2004.
46. Nguyen, X.T. and R. Kowalczyk, WS2JADE: Integrating Web Service with Jade Agents, in Verlag Berlin Heidelberg 2007. 2007, Springer: Berlin.
47. E.Cortese, F.Quarta, and G.Vitaglione, Scalability and Performance of JADE Message Transport System. 2002.
48. Yuan, P. and H. Jin, A Composite-Event-Based Message-Oriented Middleware, in GCC 2003, LNCS 3032. 2004, Springer: Verlag Berlin Heidelberg. p. 700–707.
49. Bakar, M. and S. Ghoul, A Methodology for AUML Role Modeling, in Fourth International Symposium on Innovation in Information & Communication Technology. 2011, IEEE.
50. Caire, G., JADE PROGRAMMING FOR BEGINNERS. 2009.
51. Mata, A., et al., MACSDE: Multi-Agent Contingency Response System for Dynamic Environments. 2009, Springer: Verlag Berlin Heidelberg. p. 50-59.
52. S.Muthaiyah and L.Kerschberg, Dynamic Integration and Semantic Security Policy Ontology Mapping for Semantic Web Services (SWS). 2006, IEEE: USA.
53. Chester, T.M., Cross-Platform Integration with XML and SOAP, in IT Pro September , October 2001. 2001 IEEE.
54. Pullena, J.M., et al., Using Web services to integrate heterogeneous simulations in a grid environment. Future Generation Computer Systems 21, 2009: p. 9.
55. a, D.C., G.D. b, and F.o. Vernadat, Architectures for enterprise integration and interoperability: Past, present and future. Computers in Industry, 2008. **59**: p. 12.
56. Ying-pei, W. and S. Ting-ting, Research on Information System Integration in Colleges Based on SOA. International Conference on Advances in Engineering, 2011. **24**(Procedia Engineering): p. 345 – 349.
57. Nikraz, M., G. Caire, and P.A. Bahri, A Methodology for the Analysis and Design of Multi-Agent Systems using JADE. International Journal of Computer Systems Science & Engineering special 2006(Software Engineering for Multi-Agent Systems).
58. Gregersen, A.R. and B.N.r. J0rgensen, Module Reload through Dynamic Update - The Case of NetBeans. 2008, IEEE.
59. Board, J., JADE WSIG Add-On GUIDE. 2008.
60. Martin, D., M. Paolucci, and S. McIlraith, Bringing Semantics to Web Services: The OWL-S Approach, in SWSWPC. 2005, Springer: Berlin.
61. Saadati, S. and G. Denker, An OWL-S Editor Tutorial, S. International, Editor. 2005: Menlo Park, CA.
62. Mintchev, A. and S.L. Bulgaria, Interoperability among Service Registry Implementations: Is UDDI Standard Enough, in International Conference on Web Services. 2008, IEEE

63. Maamar, Z., H. Yahyaoui, and Q.H. Mahmoud, Messengers for the Dynamic Management of Distributed UDDI Registries. 2004, IEEE.
64. Bergamaschi, S., et al., Experiencing AUML for the WINK Multi-Agent System. 2007: Italy.
65. Huhns, M.N., Agent UML Notation for Multiagent System Design, in INTERNET COMPUTING. 2004, IEEE.
66. Kertész, A. and P. Kacsuk, Grid Interoperability Solutions in Grid Resource Management. IEEE SYSTEMS JOURNAL, 2009. **3**.