Hindawi
Modelling and Simulation in Engineering
Volume 2020, Article ID 8862249, 13 pages
https://doi.org/10.1155/2020/8862249

Research Article

Resolving Interoperability Issues of Precision and Array with Null Value of Web Services Using WSIG-JADE Framework

Jaspreet Chawla,[1] **Anil Kr Ahlawat**,[2] **and Jyoti Gautam**[3]

[1]*U.P.Department of Computer Science & Engineering, JSS Academy of Technical Education, AKTU, Noida, India*
[2]*U.P.Department of Computer Science & Engineering, Krishna Institute of Technology and Management, AKTU, Ghaziabad, India*
[3]*U.P.Department of Computer Science & Engineering, JSS Academy of Technical Education, AKTU, Noida, India*

Correspondence should be addressed to Jaspreet Chawla; jaspreetchawla1@rediffmail.com

Received 20 May 2020; Revised 17 August 2020; Accepted 25 August 2020; Published 7 October 2020

Academic Editor: Agostino Bruzzone

Web services and agent technology play a significant role while resolving the issues related to platform interoperability. Web service interoperability organization (WS-I) provided the guidelines to remove the interoperability issues using basic profile 1.1/1.2 product. However, issues are still arising while transferring the precision values and an array with null values between different platforms like JAVA and .NET. As in a precision issue, JAVA supports data precision up to the 6th value and .NET up to the 5th value after the decimal and after increasing their limits, the whole number gets rounded off. In array with a null value issue, JAVA treats null as a value but .NET treats null as an empty string. To remove these issues, we use the WSIG-JADE framework that helps to build and demonstrate a multiagent system that does the mapping and conversions between agents and web services. It limits the number of digits to the 5th place after the decimal thereby increasing the precision in data sets, whereas it treats null as an empty string so that string length remains the same for both the platforms thereby helping in the correct count of data elements.

1. Introduction

Due to an increase in Internet and IT infrastructure, web services are becoming popular day by day. Web service help to build, search, and publish software applications over the Internet. The main features of web services are collaboration and interoperability, due to which one service can link with another service in a dynamic way. The web service delivers services to the end user in SOA (Service-Oriented Architecture) environment. It can be hosted on several different platform servers to provide the user a dynamic web environment [1]. SOA provides reusability, platform independence, scalability reliability, and easily maintainable features to web services [2]. It follows the three basic standards: SOAP (simple object access protocol), UDDI (Universal Description, Discovery, and Integration), and WSDL (Web Service Description Language) that makes one application to communicate with other applications [3]. A standard network protocol HTTP is used that helps to do message passing and make enterprise applications reliable and secure between two platforms. In the last few years, researchers are trying to integrate web services with an agent platform. WS-I [4] and FIPA [5] provide the guidelines to ensure that the web services and agent technology can be easily run between two platforms. The web services are designed in such a way that it is possible to interact with other client's application that is run on a different platform.

In the last few years, researchers are trying to integrate web services with an agent platform. WS-I [4] and FIPA [5, 6] provide the guidelines to ensure that the web services and agent technology can be easily deployed and run on different platforms used by clients [7]. WS-I basic profile 1.1 focuses on elementary issues of the web service, but still, some complex level issues are unspecified [8]. The large-scale business-to-business collaboration is possible only because of the interoperability of web services on different platforms. Interoperability is the major attribute of web services; without it, it is crucial to exchange information between applications, operating system, or languages [9].

In our literature studies, we have also focused on some previous and current technology to understand better on software issues. D-Bus, CORBA, Apache Thrift, etc. are the technologies that belong to high-level architecture and based on interprocess communication (IPC) but have some issues due to which they are obsolete from the market. These are not programming languages that suffer from language interoperability issues but all of these supports the programming languages to an extent.

The discussion about them is also important in this paper, as the interoperability issues are not laying only on languages but also in technologies. D-Bus is an advancement of the complex COBRA message bus system. It is a software bus for a lightweight interprocess communication system between a server and clients. D-Bus is not a high-performance IPC used only for controls, not for data. It is used specifically for Linux or Unix system and designed for local desktop applications and system messages. The plug and play application of D-Bus applies only to a small network. Processes used to connect with the D-Bus daemon and use Libdbus to control and transport messages between them but only for small programming and for large programming. D-Bus has to be more dependent on the library, tools, etc. There is a lack of security mechanism, and missing proxy services make web service a place in the market. Web service technology is used for large networks and supports many platforms. The overhead performance of the Unix socket is about 19% while the overhead of D-Bus is 32%. The main interoperability issues lie in D-Bus that it does not support the single-precision floating-point type and during the language mapping, problem arises when it does the mapping with string comparisons with the string generated locally by D-Bus. D-Bus supports several language binding like JAVA, C#, and Ruby and transfers the data with the help of the wire protocol except for a wire protocol it does not have any transporting method [10]. But in today's world, JAVA, .NET, and Python are such high-level languages that support several protocols like HTTP, TCP/IP, UDP, and sockets to serialize and deserialize the data. CORBA (Common Request Object Broker Architecture) is a platform for a distributed system, developed by Microsoft. It is a component standard for the object management group (OMG). Its purpose is to provide a language-independent, platform-independent, and vendor-independent component for a distributed system. Without a good use of the CORBA, object interoperability will be poor and reliability of the client-server program is doubtful. The CORBA supports many programming languages like JAVA and C++ to communicate between client and server but many problems were found in the CORBA during project development and testing. Thus, it may affect the reliability and portability of the program [11]. The interoperability problem always arises during language and interface mapping because every language has different basic and complex data types like JAVA which does not support an unsigned integer in an interface which can lead to problems when JAVA client communicates with the C++ server. Interoperability mainly occurs due to specifications of complex and inconsistent APIs that makes application building a tough job. Even implementation in the CORBA is quite costly and has high defect rates. The main problem arises when language clients communicate, and there is no data security and no language mapping that exist for C# and VB, no support asynchronous server-side dispatch, and complete ignorance of .NET.

After the coming of XML and SOAP in the market, this IPC was completely obsolete from the market. Apache Thrift is a software framework that supports cross-language serialization, service deployment, and 20 programming languages (JAVA, C++, Perl, Visual Basic, Python, C#, etc.) in a disturbed environment. Companies like Facebook, Twitter, Evernote, and many more distributed applications are working on it. The main key strength of Apache Thrift is its ability to simplify, centralize, and encapsulate the cross-language aspects of the system [12]. SOAP, REST, Protocol Buffers, and Apache Avro are the technologies that are used as alternatives of Apache Thrift. Moving towards this technology could enable the hardware 10 times better than others, but from the software point of view, SOAP- and REST-based services are better than Apache Thrift. An interoperability issue of Apache Thrift arises when it is connecting to asynchronous clients on interplatform and will always make some issues of exception. This technology takes more setup time, and data types do not support unsigned-numbers.

In our paper, we have used the JADE tool to build agents and all network connectivity that provides us dynamic scheduling, language interoperability, supportability, and high performance. The same properties also lie in Apache Thrift, but the language mapping, ontology mapping, and interoperability's issues of languages can easily be performed using SOAP, WSDL, ACL, and the middleware JADE-WSIG.

After several years, still, interoperability is a major issue when the exchange of information is to be done between two different platforms. Currently, achieving interoperability across multiple platforms is becoming the interest of researchers.

1.1. Interoperability. It means that software modules that are written in one language are supposed to be used by an application that may be written in a different language [13]. Due to the interoperable property of web services, they are capable of working together with applications of different domains like JAVA and .NET. Web services can be easily built and deployed on the JAVA or .NET platform using XML. As XML is a universal language, so SOAP-based web services coded in XML can be easily invoked through all the platforms. Still, issues arise at semantic and syntactic levels when passing the data structures and formats from one platform to another platform [14]. There are five interoperability issues found like precision value, an array with null value, complex data structure, date format, and unsigned data types when passing web service to the JAVA and .NET platform [15]. Researchers have used a WS2JADE [16] concept for integrating web services with the JADE platform. But in this paper, our focus is on resolving two main interoperability issues of web services; one is precision value, and the other is an array with null value issue. To remove the interoperability issues, researchers used the JADE (JAVA Agent Development

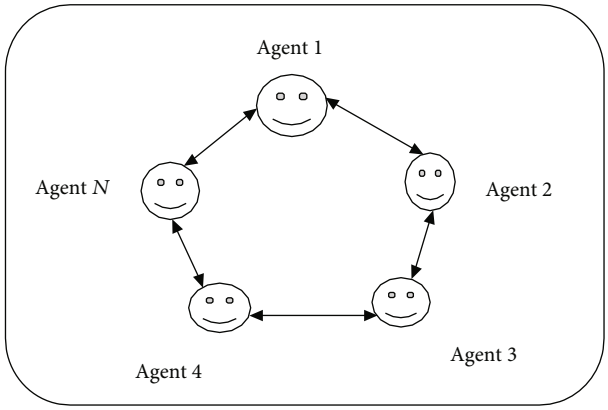

Figure 1: A multiagent system used for resolving interoperability issues. Here agent container contains agent 1: resolve precision value issue; agent 2: resolve array with null value issue.

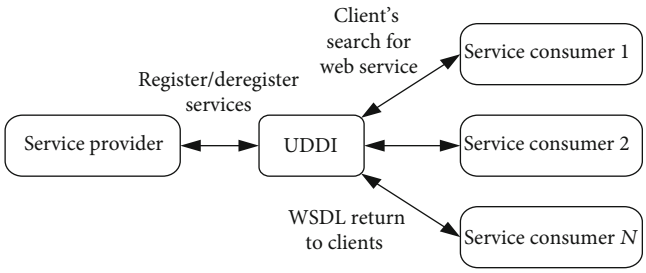

Figure 2: Web service framework.

1. User management web service
2. Stock management web service

Name	Date modified	Type	Size
StockManagementService.wsdl	5/7/2019 8:33 PM	WSDL File	7 KB
UserManagementServiceService.wsdl	5/7/2019 8:33 PM	WSDL File	12 KB

Figure 3: WSDL files for User Management and Stock Management web service.

Framework) software tool that helps in building and implementing a multi-agent system.

An agent is a software module that works on behalf of the user [17]. It provides an interoperable interface that behaves like a human agent. It collaborates and coordinates with other agent to complete its goal. Therefore, it is called the multiagent system [18] as shown in Figure 1. An agent can be autonomous or exhibits that the sharing property depends on the tasks given to them [19, 20]. The key properties of an agent are social, reactive, proactive, mobile, truthful, and benevolent [17]. The agent basically focuses on two main things: one is agent communication with other agents that is possible with ACL (agent communication language), and the other is AMS (agent management system) composed of creation, migration, operations, and location and decay of agents [21].

All the agents are stored in an agent container, and description of agents is published in DF (Directory Facilitator) [22, 23]. Every agent is assigned with an Agent Identifier (AID) that identifies an agent from other agents [24] . In DF, all agents are searched and located only through AID. In this paper, we used two agents to resolve two inter-platform issues. One agent will help the web service to resolve the precision value issue, and another helps to resolve an array with the null value issue. The purpose in using agent technology is to replace web services with better-quality web services [25] so that whenever a web service is called, a corresponding accurate WSDL must be given to clients either from DF or from UDDI.

1.2. Web Service Framework. The web service is the type of service that is accessible over the Internet. It possesses the property of loosely coupled, so it can easily call by any application and sometimes, it works as a standalone application also [4]. To send a request and receive a response from clients, the web service follows standards like XML, SOAP, UDDI, and WSDL as shown in Figure 2. Service consumers request for web services from the service provider. The service provider maintains all services and provides the services to clients in the form of WSDL file. Service providers can register/deregister, find, bind, and publish its web services to UDDI [26].

1.3. UDDI. It is a repository of web service data. The service consumer and provider can find, bind, and publish web service in an XML format.

1.4. WSDL. It describes the web service structure, description, and interface in an XML format. WSDL structure consists of six elements like input, output, message type, operations, port type, and binding.

1.5. SOAP. It is a message communicating protocol that is used with HTTP. A SOAP envelope consists of a header and a body part. A header part contains the service address, and a body part is used to transport web services using an XML structure. It can easily access remote web services and pass to client applications through SOAP [18].

In the web service framework, there is no role of an agent that is shown. Every time when a client calls a web service, a corresponding WSDL is searched from UDDI and the corresponding web service is returned to the client. If a client request for a web service from UDDI that has the same platform as that of client platform, then the client gets the correct formatted web service, but if services invoked are of different platforms, then sometimes, the interoperability issues arises [2].

1.6. Interoperability and Its Types. Interoperability arises when different modules, operating systems, and applications try to connect and coordinate with each other to share data and resources [2, 27]. There are two types of interoperability that usually occurs between different domains.

1.7. Syntactic Interoperability. It arises when two web services have different data structures and data formats [13, 28] and are deployed on different languages or domain, but apart from it, they share their data.

1.8. Semantic Interoperability. The data passed between web services should be in an understandable and meaningful format.

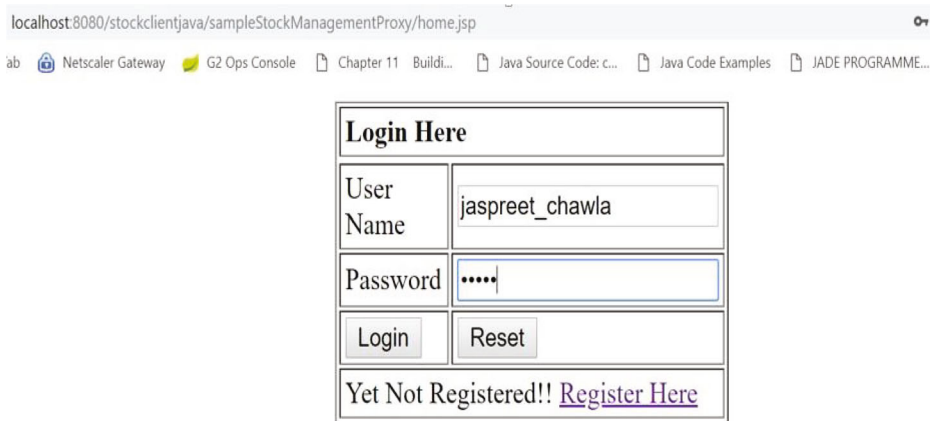

Figure 4: Testing of user authentication

Figure 5: Snapshot of stock and user data tables.

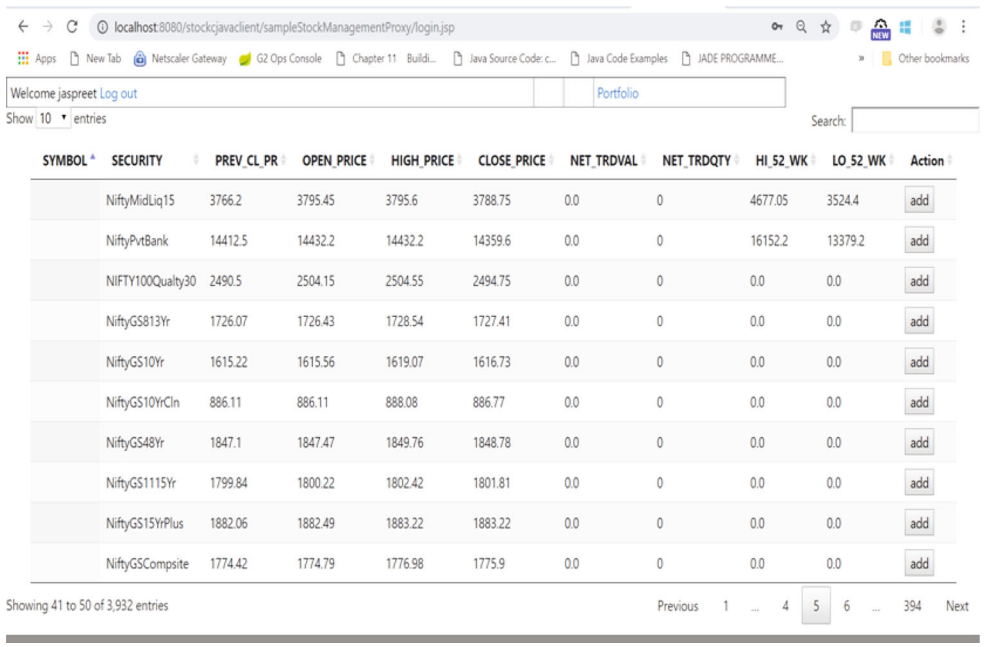

Figure 6: Snapshots of stock data.

2. Interoperability Issues between Different Platforms

In this paper, our focus is on the resolution of two interoperability issues: precision value issue and array with null value issue that generated when passing web services from one platform to another platform [29].

To demonstrate the issues, we have worked on two platforms JAVA and .NET [2] and parallel used the JAVA's software tool JADE to display the corrected result. We built the

service at the JAVA client side, the same set of service deployed at .NET client also, so that issues between two different platforms can be easily understandable. To demonstrate the web service issues, we have used online real-time stock market data [30] and design the following two web services as shown in Figure 3 by using JAVA software (Eclipse Oxygen).

2.1. User Management Web Service. In this web service, the user can add/delete or update the stock records from the user's portfolio. This web service will check user credentials before logging in to the user portfolio. A registered user has to confirm its user ID and password as shown in Figure 4. After that, stocks can be added or deleted from the user account.

2.2. Stock Management Web Service. This web service fetches the stock records from an online database of the stock market [30] in the form of .csv file. By using MySQL 5.5, data is converted into a tabular and readable format as shown in Figure 5.

Figure 6 shows the stock data of one day only that consists of 3992 rows and almost 14 columns. From the stock data sheet, researchers basically worked on stock name (referred to as security in the database), open price of a stock (open_price), and closing price of a stock (close_price) and through web service, they call the total price (sum) method that adds the open price and close price data and shows the precision value interoperability issue. After executing the web services, two corresponding WSDL files are created. These WSDL files can be registered and published in JUDDI (JAVA-UDDI).

JUUDI is the public and open-source directory that is used to register, deregister, search, and publish the web services on the Internet like other private UDDI [18]. Weather information and currency conversion are examples of web services (WSDL) that are available on the Internet, and clients can easily invoke them using their API in web applications.

3. Issues of Web Services at Different Platforms

To check the interoperability issues of the web service, client-side web services (WSDL of stock and user management) are passed to JAVA and .NET clients. The following are issues that arise at the client side.

3.1. Precision Value Issue. JAVA accepts the precision value of digits up to the sixth decimal place. When the 7th digit is placed after the decimal, the JAVA compiler would not accept it and round off the decimal to the whole number. This issue has shown through stock management web service when it fetches the online stock data [30] from the database. When stock web service is deployed on the JAVA platform, the stock's total price or sum (by adding open price and close price) accepts the precision up to six digits after the decimal; if the seventh digit is added after the decimal, then either it shows the abrupt result or it will round off the whole number [29]. Therefore, precision will go off at the seventh digit after the decimal, while the .NET precision remains consistent up to the fifth digit after the decimal, but it goes off as we add the sixth digit after the decimal. For example, a value 10.999999 works well in JAVA as it shows the precision up to 6 digits, but as the 7th digit is placed after the decimal like 10.9999999, the whole number will be rounded off and show the result as 11. However, when the same value is passed to the .NET client, the precision goes off at the 6^{th} place after the decimal and 10.999999 is rounded off to 11.

Hence, data precision always vary at different language platforms. When clients try to fetch and deploy remotely built web service in our application, the precision of data always depends upon the client's platform. Figures 7 and 8 below show the snapshot of the JAVA and .NET web service at the client side. In the JAVA web service, the sum of open price and close price will be rounded off as the price value reaches up to 7^{th} digit after the decimal as shown in Figure 7. However, in the .NET web service, the sum will be rounded off at the 6^{th} place after the decimal as shown in Figure 8.

3.2. Array with Null Value Issue. These issues also vary from one platform to another platform. To show the null value concept, researchers updated few security columns with null in rows of stock database as shown in Figure 9. There are four stocks (security) that are added in the user portfolio as shown in Figure 10. The JAVA compiler treats the null as a value and shows the correct array string length. When JAVA clients directly call the array with a null value method in their code and invoke these four stocks, it displays the string length 4 correctly. However, when a JAVA client calls the array with the null value method from the SOAP-based web services and assigned it to a new array, the null value will be lost in this case and the length of service array becomes 3.

Even when the null value method is written inside the SOAP-based web service, it prints the length of SOAP array as 3. It means that only a hard-code JAVA null value gives the correct results but calling the JAVA null value method through the web service will lose the correct string length value. Hence, it proves that there is loss of data at the client side when passing web services to them.

The same is the case with .NET; null is treated as an empty or zero-length string. As shown in Figure 9, out of four stock data, one value of security is made null in the security column. When a .NET client adds this stock data to the user portfolio, there is a blank space at the 3rd row as shown in Figure 11. The output result of .NET is the same as that of JAVA. As there are four stocks in the user portfolio, so .NET will print the string length of array 4 from a hard code and print the string length 3 in the case of array with a null value method which is calling from the outside or built inside the SOAP-based web service. Hence, in .NET, there is again a loss of data when calling the web service from outside clients.

The value of string length in both JAVA and .NET should be 4 when calling the array with the null value method through web services. But both are displaying the string length 3. Hence, this proves that there is an array with a null value issue that exists when passing web services to an inter-platform network. To resolve this issue, a JADE agent helps both JAVA and .NET clients.

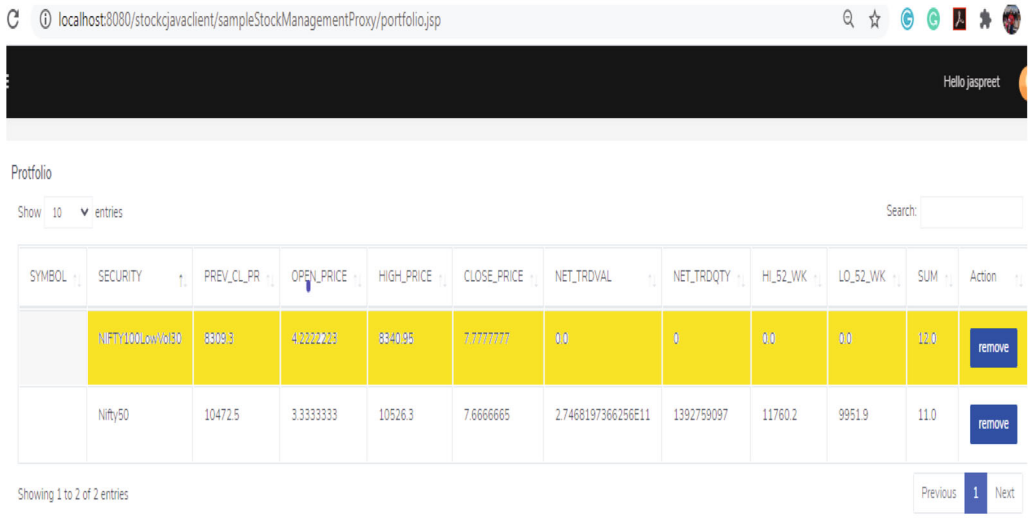

Figure 7: Precision issues in JAVA at the 7th place after the decimal.

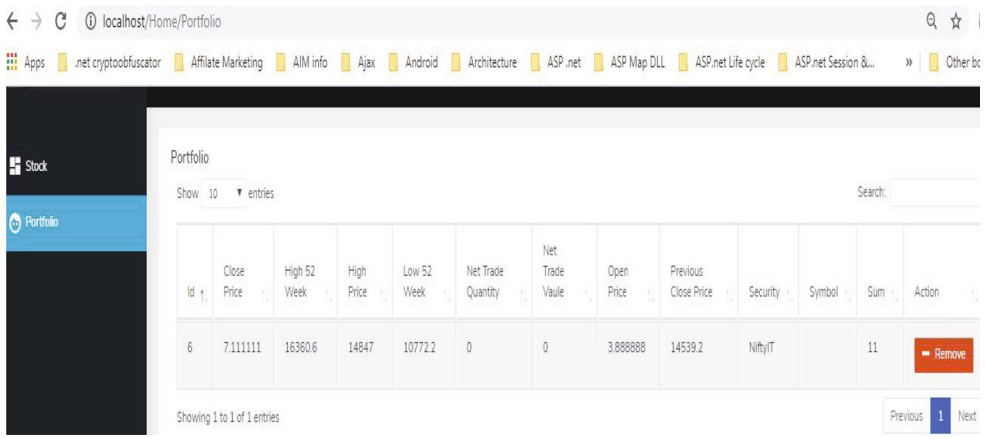

Figure 8

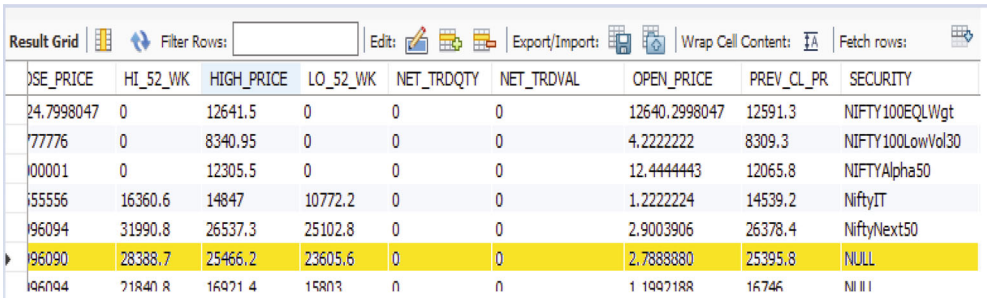

Figure 9: Adding null to stock data.

4. Framework of the Multiagent System with Web Services

To resolve the interoperability issues discussed above, a multiagent framework is used with web service [15]. Agents and web service both work on different standards and guidelines. So, a middleware is used that acts as a bridge to establish a link between them as shown in Figure 12.

JADE-WSIG (JADE-Web Service Integration Gateway) is a gateway that is used to connect and communicate web

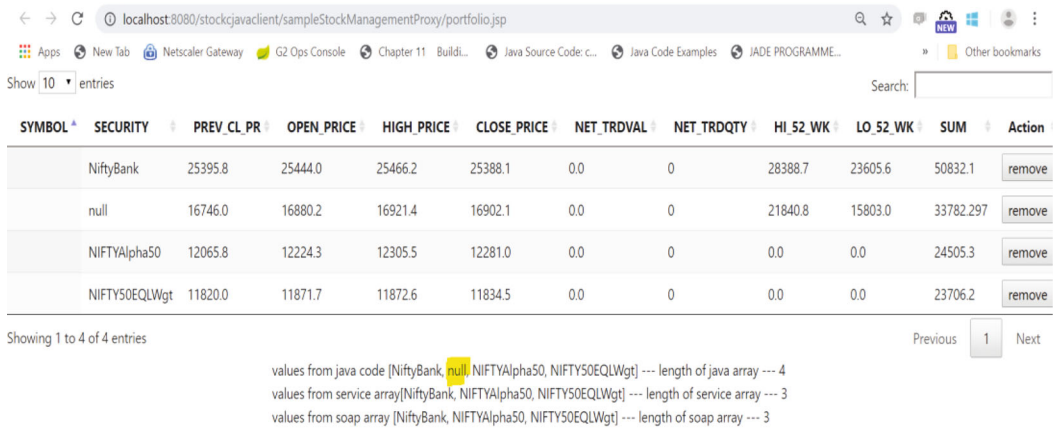

Figure 10: Array with null value in JAVA.

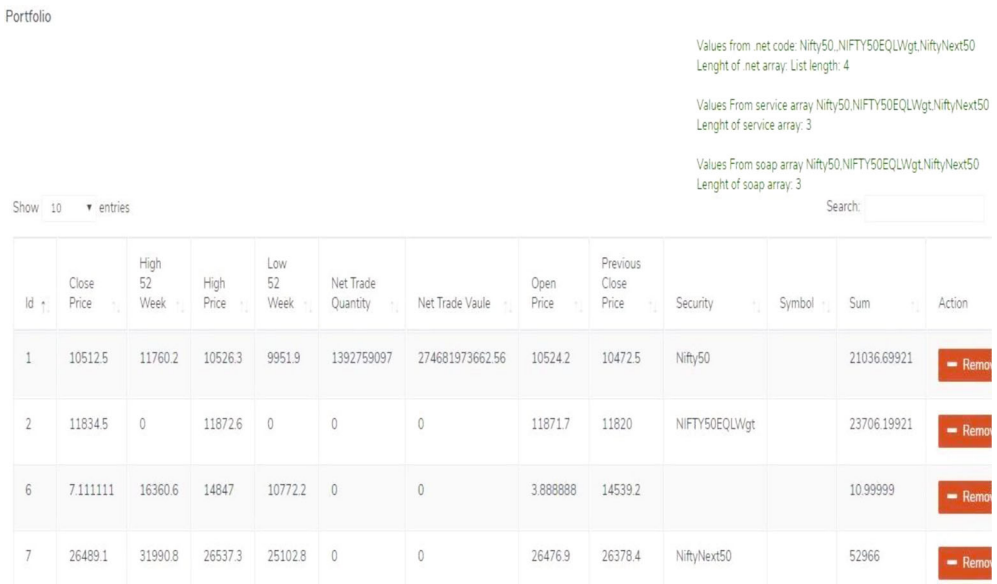

Figure 11: Array with null value issue in .NET.

services with the multiagent system [31]. WSIG is an add-on component that provides support to JADE for invoking the web services from the client side [17].

Like UDDI of web services, agents used DF to register, deregister, build, invoke, and publish software agents [29].

4.1. JADE-WSIG. It is a dynamic, integrated, and bidirectional approach between web service and agent technology. JADE is used to build and deploy a multiagent system in compliance with FIPA (Foundation of Physical Intelligent Agent) [19]. FIPA promotes agent technology to resolve interoperability issue with other applications.The gateway does three transformations [4, 28, 29]:.

(i) Service description transformations (convert WSDL to FIPA-ACL and vice versa).

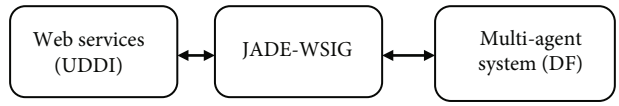

Figure 12: Framework of JADE-WSIG.

(ii) Protocol conversion transformations (convert SOAP to FIPA-ACL and vice versa).

(iii) Ontology management (convert semantic and vocabulary of JADE to web service semantic and vocabulary by passing messages between agents and web services and vice versa).

//Register codec/onto

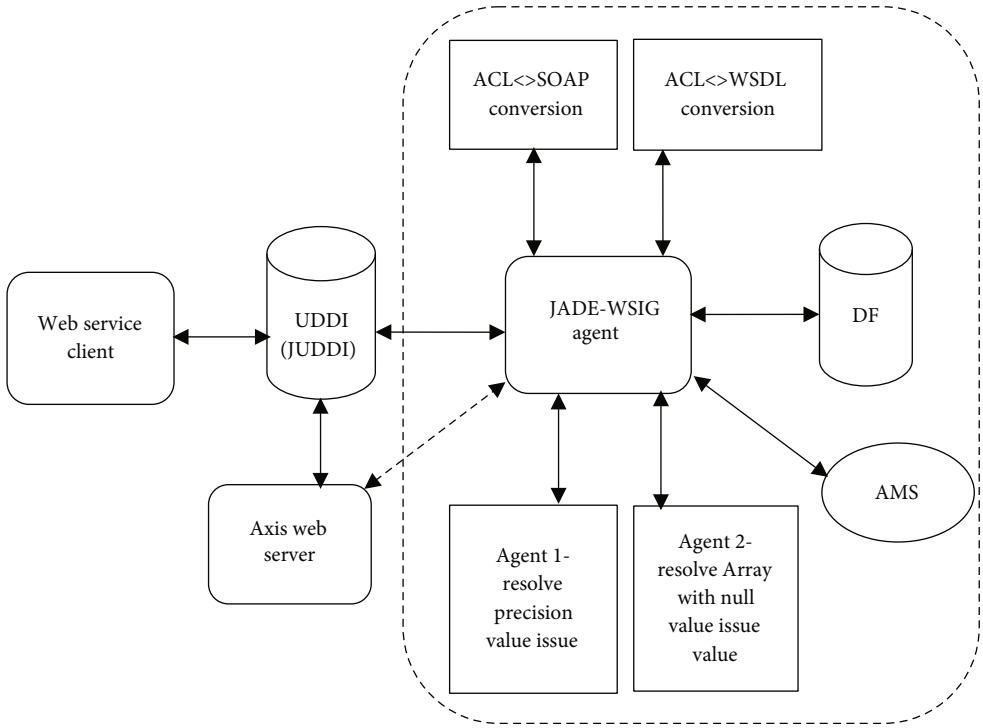

Figure 13: An architecture of JADE-WSIG using the multiagent system.

```
getContentManager().registerLanguage(co-
dec);      getContentManager().registerOntology
(FIPAManagementOntology.
getInstance());  getContentManager().regis-
terOntology(onto);
```

WSIG provides support to JADE agents to communicate with web services [18]. It helps to do the mapping between UDDI directory of the web service and DF directory of the multiagent system and web service description (WSDL) to JADE DF service description [32]. It extracts the SOAP message from the web service and passes it to the agent for taking action and converting the result back to SOAP message and pass to the web service client.

ACL (Agent Communication Language) is used by agents to coordinate, communicate, and exchange information and knowledge between them [33]. ACL is the only way to send messages of agents to middleware WSIG that actually converts the messages to a web service desirable format. The objective of WSIG is to publish interoperable agent services to JADE DF, and the same web services have to be published in UDDI also [34]. In our implementation, we used WSIG-4.6 to resolve web service issues with the multiagent system.

```
//DF registration
try {
DFService.register(this, dfad);
} catch (Exception e) {
logger.log(Level.SEVERE, "Problem during DF
registration", e); doDelete();}
```

In Figure 13, an integrated architecture of the web service with the multiagent system is shown. A web service client can register/deregister web service in UDDI [1]. To register a web service in UDDI, it requires an axis web server. There are a number of platforms currently available in the market (like JAVA or .NET) from a client that can easily fetch these SOAP-based web service from UDDI. As the web service is written in XML, clients can easily deploy these services into their application [35].

The interoperability issue arises on the client side while fetching the precision value and an array with null value services (methods) from UDDI. After the establishment of WSIG, when any of these services would be invoked from UDDI, the client requests to be forwarded to a JADE gateway agent [36].

To fulfill the request, WSIG lookup is performed in DF for available agent services; if it finds a suitable match of agent service for a particular web service, then it returns the service back to the web client; otherwise, WSIG-JADE build and deploy the agents that will help the clients to resolve the issues. AMS is a compulsory part of the agent system; all agents have to register with AMS to get an agent ID. From this agent ID, AMS can check the status of an agent like alive and dead [37].

JADE is compliant with FIPA standards, so JADE agents can interoperate with other agents easily [19].

The WSIG supports WSDL for web service description, SOAP messages for transportation, and UDDI for publishing the web services. The main features of the JADE-WSIG agent are as follows:

(i) WSIG is designed in such a way that it is accessible by both multiagent system and web services from any remote location. The main objective of WSIG

is to call a web service on behalf of the agent service from UDDI and call an agent service on behalf of the agent service from DF

(ii) It forwards the WSDL to agents [19] that are built by the JADE software tool. For each WSDL, a corresponding agent is built by JADE, i.e., agent 1 and agent 2, that helps to resolve interoperability issues related to JAVA and .NET platform

(iii) Agent 1 resolves the precision value issue, and agent 2 resolves the array with the null value issue of web services with the help of JADE-WSIG [5, 8]

(iv) Interaction between agents and WSIG is possible only with ACL, so a codec (also called converter) is used to convert SOAP to ACL and vice versa and WSDL to ACL and vice versa

(v) JADE-WSIG maintains the ontology mapping between web services and the multiagent system. It maps service description, vocabulary, and input-output parameter. It performs actions as per agent request to the web service and the web service gives response back to agents and vice versa

(vi) WSIG contains various operational components like DF and AMS and UDDI. These components are used to maintain agent and web service description that are registered/deregistered, fetched, and searched by WSIG

(vii) All JADE agents have to register their interoperable services in the DF repository. Now, WSIG translates the agent service written in ACL to WSDL and ACL to SOAP by using convertors or translators and maintain a copy of interoperable web services to UDDI for future use

4.2. Server Configuration. To install JADE-WSIG, JAVA (JDK) toolkit needs to be installed first. To demonstrate and resolve the platform issues, researchers use three servers. The server configuration [28] is shown in Table 1 below.

Table 1: Server configuration.

Platform	Server/s	Version
JAVA/JSP	JBoss-WildFly	8.X
.NET (C#)	Microsoft-IIS	10.0
JADE-WSIG	Apache Tomcat	8.5

5. JADE Result

JADE-WSIG resolves two issues with the help of a multiagent platform. Agent technology implemented accurate and heterogeneous web services [28] and proves the better result for the JAVA and .NET platform.

As shown in Figure 14, JADE-WSIG is in an active state and builds two agents named as precision handler (agent 1) and null handler agent (agent 2).

5.1. Precision Handler Agent. To resolve the precision value issue of both JAVA and .NET platforms, JADE designed, built, and implemented a precision handler agent. As JAVA rounds off the data values of the web service at the 7th place after the decimal and .NET rounds off the data values of web service at the 6th place after decimal, so, the JADE agent made a common solution for both the platforms. If the data values of interplatform web services after the decimal are either six or seven digits or above, the JADE precision handler agent controls the precision and rounds off incoming request of web service data up to the 5th decimal place and gives response back to the client/s. Hence, whenever JAVA or .NET clients request (call) stock management web service from UDDI, it redirects the request to the agent's DF through WSIG and fetches the precise and updated WSDL of the 5^{th} digit place from the agent platform. The copy of the updated WSDL is also placed in UDDI for further reference of clients. Hence, precision would not be lost in both the cases as shown in Figures 15 and 16.

As shown in Figure 15, when a stock management web service is called by the .NET client, the precision handler agent controls the parameters of the open price (7.111111) and close price (3.888888) up to 5 digits so that the sum of both (open price and close price) will retain the precision up to 5th digit place after the decimal. Hence, the resultant value of the sum (10.99999) is up to the 5^{th} digit only; the 6^{th} digit after the decimal is waived off by the precision handler agent.

```
   private void serveSumAction(Sum sum, Action
actExpr, ACLMessage
   msg) throws RemoteException {
   DecimalFormat        df        =        new
DecimalFormat("#.#####");
   df.setRoundingMode(RoundingMode.FLOOR);
float soapResult = Float.parseFloat(df.format
(sum.getFirstElement()))
   +                              Float.parseFloat
(df.format(sum.getSecondElement()));
   ACLMessage notification = prepareNotificatio-
n(actExpr, msg,
   ACLMessage.INFORM,df.format(soapResult)); noti-
fication.addUserDefinedParameter("USER-PARAM",
"Test value passed as user defined parameter");send
(notification);
```

The same result of precision up to the 5^{th} decimal place is shown in Figure 16 with the help of a precision handler agent when a JAVA client calls the web service from UDDI.

5.2. Null Handler Function Agent. To resolve the interoperability issue of an array with a null value, JADE built and deployed an agent called null handler function.

Both JAVA and .NET give the same results of an array with the null value method as discussed above. JADE provides [5] a common solution for both the platforms. Whenever the array with the null value is encountered in a web service method, JADE control and replace the null value with an empty string and print the string length 4 as shown in Figure 17. It means that if there is any row of data showing

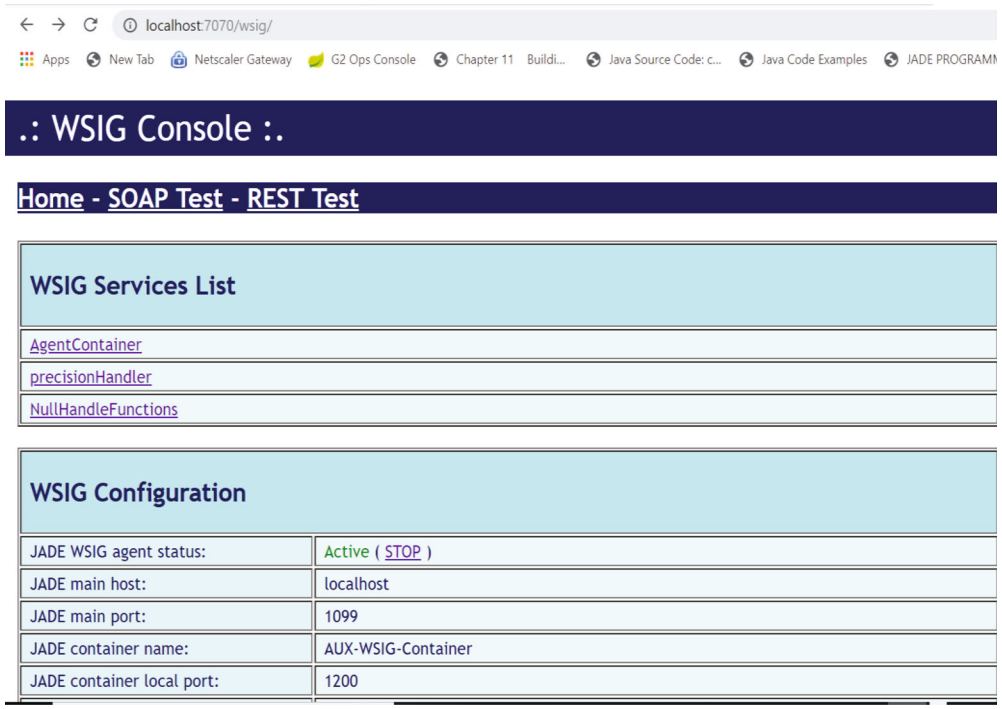

Figure 14: WSIG-agent service list.

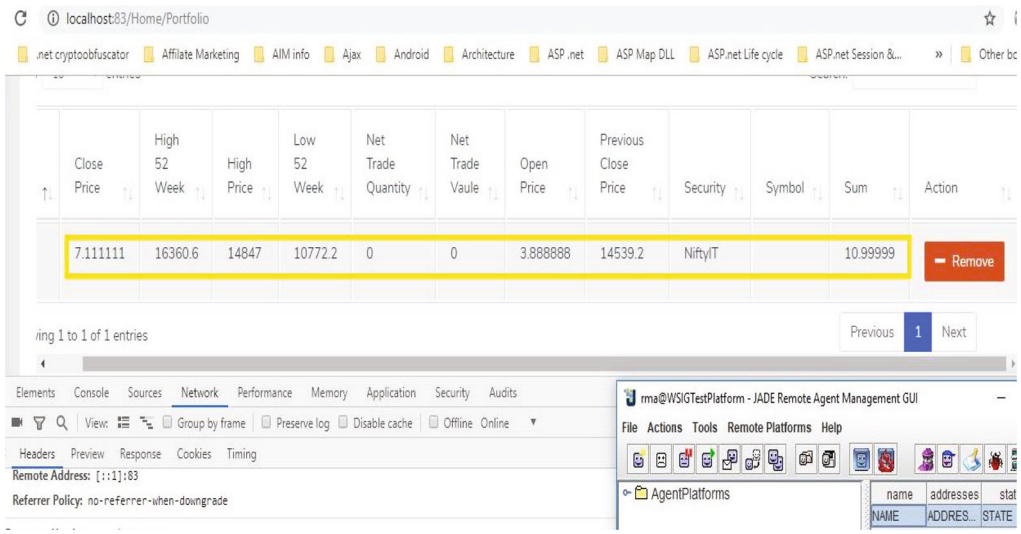

Figure 15: .NET precision value by JADE.

a null value in the user's portfolio, it will return an empty string as output.

Whenever the JAVA or .NET platform calls this stock management web services to carry a null value in their applications, both will get the same result from JADE DF. So no issue would arise in both the cases. By deploying a JADE agent, both the platforms will get the same result and there will be no loss of data.

```
private void serveNullHandlerAction(NullHandler nullHandler, Action actExpr, ACLMessage msg)throws RemoteException {
    String firstElement = Objects.toString(nullHandler.getSecondElement(),"");
    ACLMessage notification = prepareNotification(actExpr, msg, ACLMessage.INFORM, firstElement);
    notification.addUserDefinedParameter("USERPARAM", "Test value passed as user defined parameter");send (notification);
    }
```

After resolving the issues of web services through JADE agents, the updated web service carrying array with the null

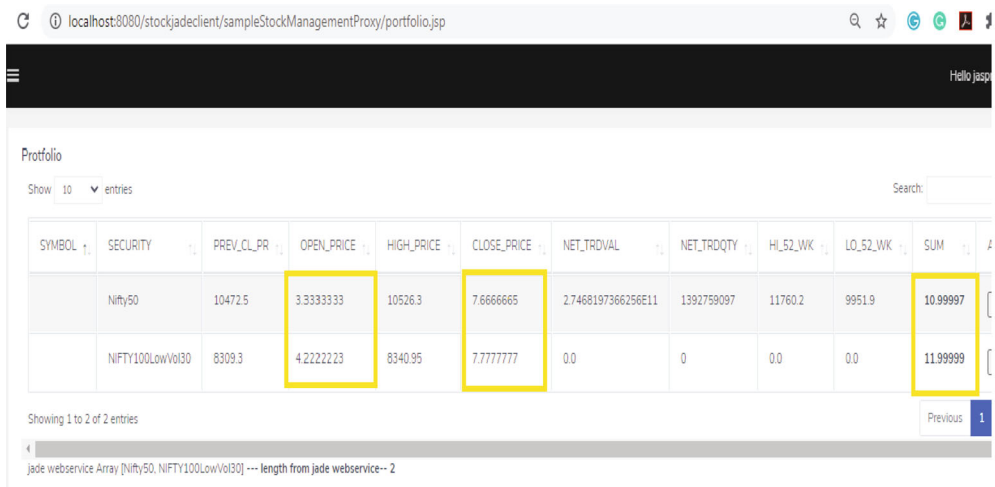

Figure 16: JAVA precision value by JADE.

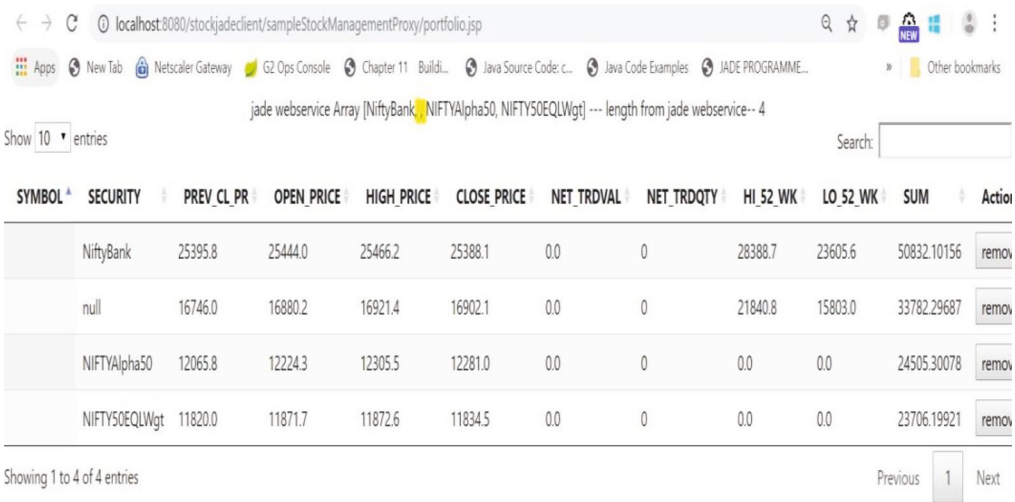

Figure 17: JADE result for null values.

value method is published in agent's directory DF and through WSIG, the replica of web services is also published in UDDI.

6. Conclusions

Today, various versions of software language and gateways are available in the market for the removal of interoperability issues of web services but WSIG-JADE is the well-suited middleware between web services and agent technology. It supervise the multiagent system and used it to resolve interoperability issues of interplatform. Issues like precision value and array with a null value of web services are accurately resolved by JADE agents. In this paper, we tried to design and implement a common approach using JADE for JAVA and .NET platform for web service interoperability issues. Due to the distributed and dynamic behavior of agents, at run time, they do ontology mapping of web service description (WSDL) and agent service description (DF), control the service parameters, and return the correct result of precision up to the 5^{th} decimal place and the array with a null value as an empty string to client's application.

7. Current and Future Scope

As in current development of the multiagent system, we have used an agent for each interoperability issue. The number of agents would be increased as we try to resolve more issues. The future development would be to minimize the number of agents. A single controller agent would take care of all the issues and if necessary would take the help of other agents under its control. Therefore, hierarchy of agents would be built in a controlled environment.

In future scope, we will try to increase the scaling of JADE-WSIG using cloud computing. Web services are building to handle enormous workloads. But JADE-WSIG could not introduce any bottleneck for scaling of the application as shown in Figure 18. Cloud scaling can be done either in

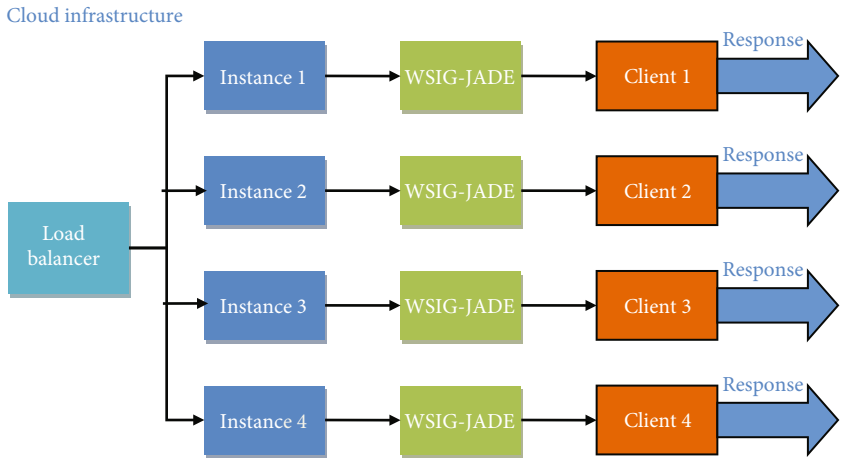

Figure 18: Load balancing of JADE-WSIG with multiple instances.

a horizontal or vertical way. In vertical scaling, we can only change the configuration of the server without making its multiple instances. The cloud has the self ability to change the architectural style of web services like storage capacity and even the number of processors, RAM, etc. without changing the code. In horizontal scaling, more servers will be added by the cloud at run time as a load of data becomes large in size on one server or instance. Multiples instances are made by clouds depending upon the load-bearing capacity of instance. With load balancing, all devices and processes perform the same amount of work in an equal amount of time. The load balancer connects to instances as the load increases and will distribute the load on multiple instances (servers) depending upon demand. A load balancer can increase or decrease the traffic on servers automatically by adding or removing servers with minimal disruption of traffic flow. So a bottleneck can never be made when using JADE-WSIG, even it will increase the throughput in minimum response time.

Web service instance on the cloud works like other service instances, divides the server load to another server, and increases its efficiency and performance.

In our next research paper, we will focus on to resolve other interoperability issues of JAVA/.NET like unsigned numbers, date format, and collection of complex data types and try to implement web services with JADE-WSIG on the cloud.

Data Availability

The data used to support findings of this study can be accessed from https://www1.nseindia.com/products/content/equities/equities/archieve_eq.htm.

Conflicts of Interest

The authors declare that they have no conflict of interest.

Acknowledgments

The author, Jaspreet Chawla, would like to express her thanks and gratitude to her PhD. Guide Dr. (Prof.) Anil Kumar Ahlawat who encouraged her to work on these issues and helped her in writing the script. Also, she would like to thank Dr. Jyoti Gautam, who provided her the required support for framing and proofreading of manuscripts and guided her in resolving the platform issues.

References

[1] F. Ahmad, A. Sarkar, and N. C. Debenth, "Analysis of dynamic web services," in *2014 International Conference on Computing, Management and Telecommunications (ComManTel)*, pp. 275–279, Da Nang, Vietnam, 2014.

[2] X. Wei, "Communications solutions for heterogeneous systems based on SOA," *Physics Procedia*, vol. 25, pp. 1738–1742, 2012.

[3] T. Vitvar, M. Zaremba, and A. Mocan, "Execution model for heterogeneous web services," in *2008 IEEE Congress on Services - Part I*, Honolulu, HI, USA, 2008.

[4] "Web service interoperability organization," Feb. 2019 http://www.ws-i.org/.

[5] "Foundation for intelligent physical agent," June 2005, http://www.fipa.org/.

[6] June 2002, http://JADE.tilab.com/papers/2005/JADEWorkshopAAMAS/AAMAS05_JADE-Tutorial_WSIG-Slides.pdf.

[7] I. A. Elia, N. Laranjeiro, and M. Vieira, "Understanding interoperability issues of web service frameworks," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 323–330, Atlanta, GA, USA, 2014.

[8] "WS-interoperability," April 2007 http://www.ibm.com/developerworks/webservices/tutorials/wsunderstand-web-services6.

[9] P. A. Pico-Valencia and J. A. Holgado-Terriza, "An agent middleware for supporting Ecosystems of heterogeneous web services," *Procedia Computer Science*, vol. 94, pp. 121–128, 2016.

[10] https://dbus.freedesktop.org.

[11] https://docs.oracle.com/cd/E15261_01/tuxedo/docs11gr1/tech_articles/CORBA.html.

[12] https://livebook.manning.com/book/programmers-guide-to-apache-thrift.

[13] N. M. Ibrahim, M. F. Hassan, and Department of Computer Science and Information Systems CSIS, University of Limerick, Limerick, Ireland, "A comprehensive comparative study of MOM for adaptive interoperability communications in service oriented architecture," *International Journal of Trend in Scientific Research and Development*, vol. Volume-3, no. -Issue-3, pp. 23–30, 2019.

[14] H. R. M. Nezhad, B. Benatallah, F. Casati, and F. Toumani, "Web services interoperability specifications," *Computer*, vol. 39, no. 5, pp. 24–32, 2006.

[15] N. M. Ibrahim, M. F. Hassan, and Z. Balfagih, "Agent based MOM interoperability framework for integration and communication of different SOA applications," *International journal of new computer architecture and the applications*, pp. 412–427, 2011.

[16] S. Nguyen, R. Kowalczl, M. B. Chhetri, and A. Grant, "WS2JADE: A tool for run-time deployment and control of web services as JADE agent services," in *Software Agent-Based Applications, Platforms and Development Kits*, pp. 223–251, Birkhäuser Basel, 2005.

[17] H. S. Nwana and D. T. Ndumu, "A brief introduction to software agent technology," in *Agent technology book chapter*, pp. 29–47, Springer, Berlin, Heidelberg, 1998.

[18] H. S. Nwana, "Software agents: an overview," *Knowledge Engineering Review*, vol. 11, no. 3, pp. 205–244, 1996.

[19] J. M. Mendes, P. Leitão, F. Restivo, and A. W. Colombo, "Service-oriented agents for collaborative industrial automation and production systems," in *Holonic and Multi-Agent Systems for Manufacturing*, pp. 13–24, Springer, Berlin, Heidelberg, LNCS Springer, 2009.

[20] G. Ali, N. A. Shaikh, and A. W. Shaikh, "A research survey of software agents and implementation issues in vulnerability assessment and social profiling models," *Australian Journal of Basic and Applied Sciences*, vol. 4, no. 3, pp. 442–449, 2010.

[21] T. X. Nguyen and R. Kowalczyk, "W**S**2JADE: integrating web services with JADE agents," in *Service-Oriented Computing: Agents, Semantics, and Engineering*, pp. 1–11, Springer, Berlin, Heidelberg, 2006.

[22] F. Leonand and C. Badica, "A freight brokering system architecture based on web services and agents," in *Lecture Notes in Business Information Processing*, vol. 247, pp. 537–546, Springer, Cham, 2016.

[23] B. S. KiM, B. H. Son, S. W. Han, and H. Y. Youn, "Agent platform-based directory facilitator for efficient service discovery," in *2009 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pp. 101–107, Zhangijajie, China, 2009.

[24] M. N. Huhns, "Software agents: the future of web services," in *Lecture Notes in Computer Science*, pp. 1–18, Springer, Springer, Berlin, Heidelberg, 2003.

[25] A. Canito, G. Santos, J. M. Corchado, G. Marreiros, and Z. Vale, "Semantic web services for multi-agent systems interoperability," in *EPIA Conference on Artificial Intelligence*, pp. 606–616, Springer, Cham, 2019.

[26] R. Udayakumar, K. P. Thooyamani, and V. Khanaa, "A comparison of J2EE and NET as platforms for developing E-government applications," *International Journal of Engineering Research and Development*, vol. 7, no. 1, pp. 116–121, 2003.

[27] F. Cohen, "Understanding web service interoperability," *IBM-Developer work*, 2002.

[28] J. Chawla, A. Ahlawat, and G. Goswami, "A review on web services interoperability issues," in *2018 5th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, pp. 1–5, Gorakhpur, India, 2018.

[29] X. T. Shetty and S. Vadivel, "Interoperability issues seen in web services," *IJCSNS International Journal of Computer Science and Network Security*, vol. 9, no. 8, pp. 160–169, 2009.

[30] "National stock exchange of India pvt. Ltd," 2017 https://www1.nseindia.com/products/content/equities/equities/archieve_eq.htm.

[31] D. Greenwood, M. Lyell, A. Mallya, and H. Suguri, *The IEEE FIPA-approach to integrate software agents with web services*, AAMAS, Honolulu, Hawai'I, USA, 2007.

[32] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa, "JADE: a software framework for developing multi-agent applications. Lessons learned," *Information and Software technology*, vol. 50, no. 1–2, pp. 10–21, 2008.

[33] Y. Labrou, T. Finin, and Y. Peng, "Agent communication languages: the current landscape," *IEEE Intelligent Systems*, vol. 14, no. 2, pp. 45–52, 1999.

[34] X. T. Nguyen and R. Kowalczyk, "Enabling agent-based management of web services with WS2JADE," in *Fifth International Conference on Quality Software (QSIC'05)*, Melbourne, Victoria, Australia, Australia, 2005.

[35] M. S. Hemayati, M. Mohsenzadeh, M. A. Seyyedi, and A. Yousefipour, "A framework for integrating web services and multi-agent systems," in *2010 2nd International Conference on Software Technology and Engineering*, pp. 1–4, San Juan, PR, USA, 2010.

[36] N. Haile and J. Altmann, "Evaluating investments in portability and interoperability between software service platforms," *Future Generation Computer Systems*, vol. 78, pp. 224–241, 2018.

[37] D. Greenwood and M. Calisti, "Engineering web service - agent integration," in *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, The Hague, Netherlands, 2004.

[38] "JAVA agent development framework," June 2017 http://www.JADE.tilab.com.