Experiment No.: 4

Integrate Software Components using middleware

Learning Objective: Student should be able to integrate software components using middleware

Tool:- Corba

Theory:

Software system integration is essential where communication between different applications running on different platform is needed. Suppose a system designed for payroll running with Human Resource System. In that case employees' data need to be inserted in both systems. The system integration benefits a lot in these cases where data and services needed to be shared.

Web services are becoming very popular to share data between systems over the network and over the internet as well. In software industry the software integration carried same steps as software development and hence demands same kind of development procedures and testing.

This ensures the meaningful and clear communication between the systems. Systems integration becomes inevitable in Enterprise Systems where the whole organization needed to share data and services and give the feel to user as one system. The core purpose of integration is to make the systems communicate and also to make the whole system flexible and expandable.

The integration of different softwares written in different language and based on different platforms can be tricky. In that situation a middleware is necessary to enable the communication between different softwares. The middleware enables the software system not only to share data but also share the services.

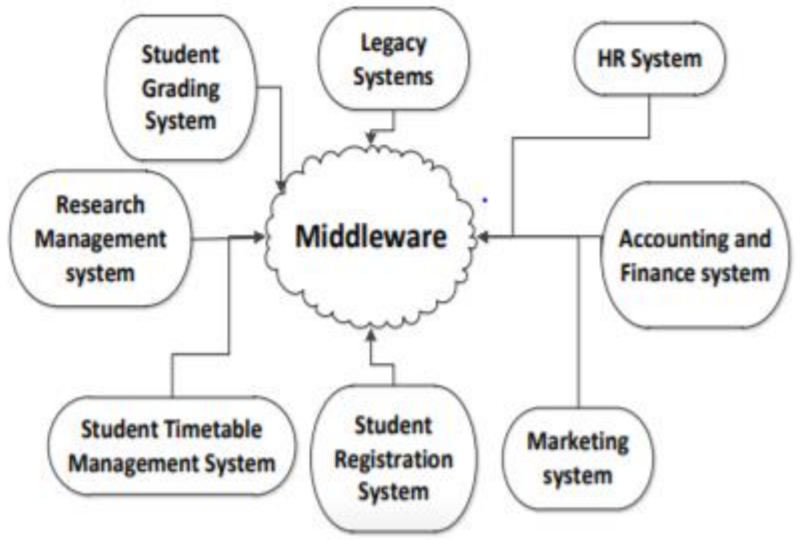

Student Grading System
Legacy Systems
HR System
Research Management system
Middleware
Accounting and Finance system
Student Timetable Management System
Student Registration System
Marketing system

Figure 1: Middleware

A . Middleware

Independently written software systems need to be integrated in large system for example industry, institution, etc. These systems need to agree on common method for integration. To get them communicate there is need to have something in between them. The middle thing is termed commonly as middleware.

B .Service Oriented Architecture

Service Oriented Architecture (SOA) is the architectural design and pattern which is used to provide services to different applications. Its goal is to achieve loose coupling between interacting components of applications. Web Services, Corba, Jini, etc are the technologies used to implement SOA.

Main benefit of SOA is that it can provide the means of communication between completely different applications (built in different technologies). The services are also completely independent and reusable and the nature of reusability provide the less time to market. All the services technologies needs to be implemented using the SOA design pattern and need to be designed on the basis of SOA to get maximum benefit.

C .Web Services

Web service is the SOA technology with additional requirements of using internet protocols (HTTP, FTP, SMTP, etc.) and using XML (Extensible Markup Language) for message transmission. [7] These are application components which communicate between different applications using open protocol. Open protocol is the web protocol for querying and updating information. These components can be used by different kinds of applications to exchange information. HTML (HyperText Markup Language) and XML are the basics of web service implementation.

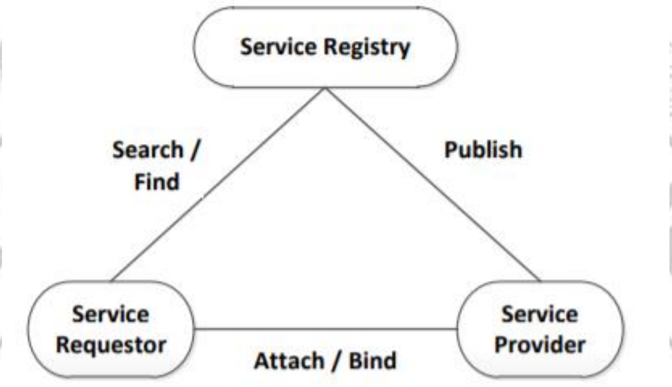


Figure 2: Web-Service Architecture

D.WSDL (Web Service Descriptive Language)

WSDL is a language to describe web services and providing the link to access these web services. It is acting as a publisher in web service architecture. It is the XML document which is recommended by W3C (World Wide Web Consortium) in 2007. In WSDL XML describes the service and the address from where applications can access the service. [8] WSDL predecessors were COM and CORBA

E. UDDI (Universal Description, Discovery and Integration)

It is directory service / registry service where applications can register and look for web services. It is Platform independent framework. It is acting like service register in web service architecture. It uses HTML, XML and DNS (Domain Name Server) protocols which enables it to become the directory service. It defines the keyword search, categories and classification for an application and registers it into business directory. In that way it is making the application easier to be approached by the customers online

F. SOAP (Simple Object Access Protocol)

It is a messaging / invoker in web service architecture. It is use to send messages in between the service and service consumer; and in between service consumer and service registry. It used to communicate with web service. It is a message framework which transfers the information between sender and receiver. SOAP doesn't define the service but defines the mechanisms for messaging. It binds the client to the service

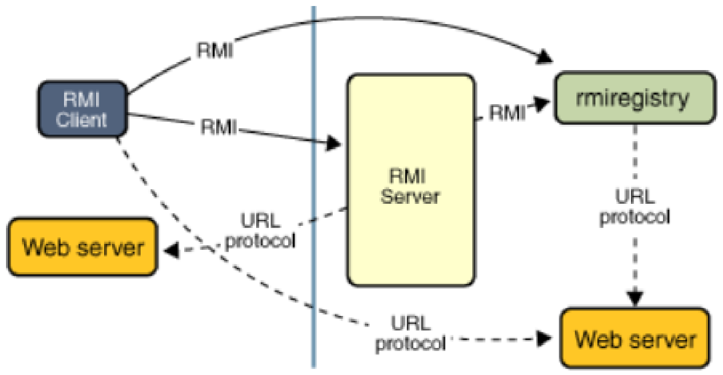
G.SOA - A solution to spaghetti architecture

In a fairly medium scale to large software architecture where there is need to integrate or communicate between different kinds of applications the introduction of links can make the architecture messy and it is called spaghetti architecture [9]. Figure 4 shows that problem in detail. It makes the system less flexible and expandability is the nightmare. Service Oriented Architecture (SOA) makes the architecture flexible and expandable.

Result and Discussion:

1.An Overview of RMI Applications :

describes the RMI system and lists its advantages. Additionally, this section provides a description of a typical RMI application, composed of a server and a client, and introduces important terms.
RMI applications often comprise two separate programs, a server and a client. A typical server program creates some remote objects, makes references to these objects accessible, and waits for clients to invoke methods on these objects. A typical client program obtains a remote reference to one or more remote objects on a server and then invokes methods on them. RMI provides the mechanism by which the server and the client communicate and pass information back and forth. Such an application is sometimes referred to as a *distributed object application*.



2.Writing an RMI Server walks through the code for the compute engine server. This section will teach you how to design and to implement an RMI server

Designing a Remote Interface

This section explains the `Compute` interface, which provides the connection between the client and the server. You will also learn about the RMI API, which supports this communication.

Implementing a Remote Interface

This section explores the class that implements the `Compute` interface, thereby implementing a remote object. This class also provides the rest of the code that makes up the server program, including a `main` method that creates an instance of the remote object, registers it with the RMI registry, and sets up a security manager.

3. Creating a Client Program

The compute engine is a relatively simple program: it runs tasks that are handed to it. The clients for the compute engine are more complex. A client needs to call the compute engine, but it also has to define the task to be performed by the compute engine.

Two separate classes make up the client in our example. The first class, `ComputePi`, looks up and invokes a `Compute` object. The second class, `Pi`, implements the `Task` interface and defines the work to be done by the compute engine. The job of the `Pi` class is to compute the value of π to some number of decimal places.

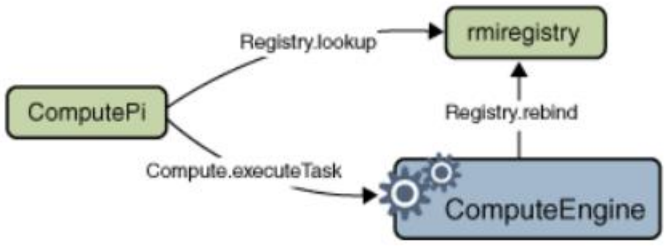
The non-remote `Task` interface is defined as follows:

```
package compute;

public interface Task<T> {
    T execute();
}
package client;

import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.math.BigDecimal;
import compute.Compute;

public class ComputePi {
    public static void main(String args[]) {
        if (System.getSecurityManager() == null) {
            System.setSecurityManager(new SecurityManager());
        }
        try {
            String name = "Compute";
            Registry registry = LocateRegistry.getRegistry(args[0]);
            Compute comp = (Compute) registry.lookup(name);
            Pi task = new Pi(Integer.parseInt(args[1]));
            BigDecimal pi = comp.executeTask(task);
            System.out.println(pi);
        } catch (Exception e) {
            System.err.println("ComputePi exception:");
            e.printStackTrace();
        }
    }
}
```



4. Compiling and Running the Example

Now that the code for the compute engine example has been written, it needs to be compiled and run.

- Compiling the Example Programs

In this section, you learn how to compile the server and the client programs that make up the compute engine example.

- Running the Example Programs

Finally, you run the server and client programs and consequently compute the value of π.

Learning Outcomes: Students should have be able to

LO1: Define Middleware.

LO2: Identify different components in middleware.

LO3: Explain Software Components using middleware.

Course Outcomes: Upon completion of the course students will be able to understand middleware and its components.

Conclusion: We were able to understand middleware and its components. We were able to integrate components using Middleware.

For Faculty Use

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	