International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 – 5161

Available at http://inpressco.com/category/ijcet

Research Article

Enterprise Application Integration using Service Oriented Architecture with Web Service Aggregation Pattern

Nilesh Vishwasrao Patil†* and M. C. Kshirsagar†

Accepted 10 Aug 2015, Available online 20 Aug 2015, **Vol.5, No.4 (Aug 2015)**

Abstract

Enterprise Application Integration (EAI) is assembling heterogeneous applications to communicate each other and reducing their complexity. Service Oriented Architecture (SOA) has been offered standard, principle and architectural design style with best IT enterprises align. As market growing, the new technologies, systems and databases has been introducing rapidly to achieve the demands and current market goals. Legacy systems are still playing the significant role in today's electronic business even introduces new and powerful systems, applications and databases. But still it could not replace Legacy system because huge amount of money invested by organizations. The proposed system will have an integration layer which is exposed as a service to achieve loosely coupled architecture using service oriented architecture. It can be web service based or Messaging Queue based scenario to achieve this. One of the main objectives for efficient service oriented architecture is to achieve interoperability between the disparate systems. The motivation behind of integration layer is to save money and time in future for electronic business by providing generic pattern to minimize efforts of developer to do work from scratch.

Keywords: *SOA: Service Oriented Architecture, WS: Web services, WSDL: Web Service Description/Definition Language, W-SOA: Web Service Oriented Architecture, ESB: Enterprise Service Bus*

1. Introduction

Today's business applications rarely work in isolation. The customer and clients is believed in the instant access to all business applications which offered by an enterprise, without worrying about which systems provides the functionality. The integration solution will be required for integrating disparate and homogeneous systems or applications to be connected with each other. The enterprise integration is usually achieved through the middleware between disparate applications. Middleware is used for routing request, transforming data and data transport.

The electronic business is becoming more and more dynamic which experiencing the major changes, since the market is in hurry to develop of new systems, databases, technologies for providing or adding efficient and dynamic nature to electronic business over the legacy system and databases. As per IDC (International Development Corporation) survey todays 80 percent of electronic data generated in last two year is mostly unstructured and requires extra complexity to analyze it.

Enterprise Application Integration (EAI) is a computing business term used to plan, manage and integrate homogeneous or heterogeneous systems inside and outside of the electronic business. Enterprises may have legacy systems (like Mainframe), existing databases, and systems, but may want to achieve functionality, effectiveness of new invented databases or systems. Enterprise is also need to communicate with other organization which may be same or disparate systems. For a business organization investing money cannot find the replacement of existing legacy systems as a good solution to add new functionalities in turn the effectiveness to business they carry out electronically. Instead of replacing the legacy systems, develop an integration layer between those systems with newly systems and provide conversion of data format logic in integration layer which will allows the systems communicate with each other without regards of homogeneous or heterogeneous systems. For example system A is Mainframe which expects data in fixed length (COBOL copybook) format and wants to communicate with system B which will require data in XML format. In this case integration layer will allows to both systems for communication even both are disparate by providing conversion logic in the integration layer.

The business environment should be efficient, dynamic and integrated IT systems for disparate applications in along with out of the organization. It leads to the complexity of different systems and applications have increased, primarily due to it is becoming more and more dynamic [Martin Potocnik *et*

*Corresponding author **Nilesh Vishwasrao Patil** is a ME Student **and M. C. Kshirsagar** is working as Assistant Professor

al, 2014] [M.P. Papazoglou *et al*, 2007]. A lot of industrial enterprises have acquired the disparate applications and systems over the decades. They need to integrate these heterogeneous systems and applications which will always be prominent for satisfying business requirements and needs [Wu He *et al*, 2014].

For numerous industries enterprises, the situation is imperative for such industry applications and systems to work together for succeeding definite business goals. Enterprise Application Integration (EAI) has been developed to help for achieving the quality integration to address integration problems [D. S. Linthicum, 2000] [Wu He *et al*, 2014].

Service oriented architecture is used to achieve loosely coupling in enterprise application integration by providing everything work using services. The services are independent on each other so leads to achieve loosely coupling in the electronic business. It promotes the idea of assembling application components into a network of services that can be loosely coupled to create flexible, dynamic business processes and agile applications that span organizations and computing platforms [Martin Potocnik *et al*, 2014].

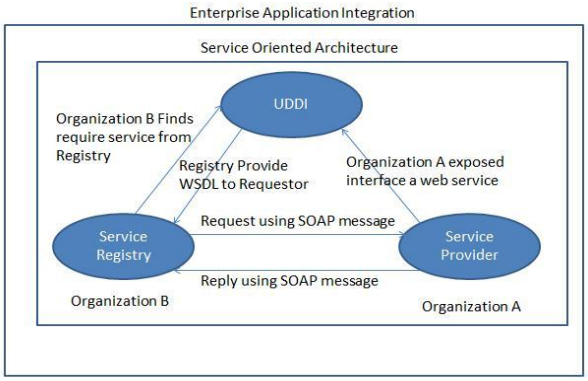

Fig. 1 Relationship between EAI, SOA, and SOAP

Enterprise integration of disparate application is very complex task. The pattern will provide the guideline for developer and designer during integration of enterprise applications. They are accepted solutions to recurring problems within a given context. Patterns are the abstract view to enterprise applications to integrate disparate applications by reducing its complexity, time and cost. The patterns will help to integration architects and developers design and implement integration solutions more rapidly and reliably [WMB, 2012].

In this, provides the web service aggregation pattern which will be responsible for providing visual notations and vocabulary (documentations about processing) for large scale and complex integrations solutions with the use of different technologies.

2. Literature Survey and Related Work

Enterprise application integration and service oriented architecture are not innovative word for organization, but everyday some ingredients are added in both to make it as efficient as possible for electronic business. These two words already knock the door of industry in last decade. Now a day electronic business is totally depends on these two concepts because of significant role in enterprise integration of disparate applications by reducing the cost and complexity of business.

The enterprise integration can be loosely coupled or tightly coupled of disparate applications. Disparate applications are communicating with each other on the basis of message-based or method-based. The communication between the applications is synchronous or asynchronous as per business need. Putting these things together in an intelligent way is essential to archive the promise electronic business.

As industry is going, the enterprise environments, most of businesses systems are typically develop through number of technologies, applications, protocols, and devices that are located across a network. IT market demands are increased, enterprises need to distribute disparate applications and systems across multiple organizational and geographical boundaries in recent years. There are heavy demands for disparate systems and applications are integrate to enhance or increase enterprises competitiveness [Wu He *et al*, 2014]. The development in integration of homogeneous or heterogeneous applications is as follows

A. Coupling architecture

The tightly coupled integration is integration where applications directly communicate with each other's by providing integration specific code in each application. As they integrate directly, need to know implementation details of each other. It leads to components of this architecture can create the dependencies on each other. Tightly coupled applications are very hard to changes because of the dependency of other application.

Loosely coupled system is those systems which are not dependent on the platform, programming language, operating systems. It divides particular integration task into multiple parts called as a service which is independent task. Even changed the service, it will not affects or no need to change the service. Loosely coupled integration has numerous advantages over the tightly coupled architecture but one con is tightly coupled system is much faster than the loosely coupled system because of invocation is done directly but it will become very complex as applications increases for integration.

B. Point-to-point Integration

In point to point integration, the integration code of another system is available. This type of integration is very difficult to manage as new applications wants to add into integration. It will make N-squared integration problem. It is also not good solution for integration as market need to change every day. It is

very complex and inefficient approach for integration [Integration, web link]

C. Hub Model with Adapters

There are several benefits of using the hub model over the point to point integration. In this each application and systems need to communicate through hub and update their code into hub only. But still it becomes tightly coupled architecture and more number of applications in integrations becomes complex and inefficient [Integration, web link].

D. Communication Layer Integration

Heterogeneous distributed applications want to integrate for the communication with each other and share the significant information between them. Those applications know the operations and status of remote applications to perform the communication task. They are typically used HTTP (Hypertext transport protocol) and IIOP (Internet Inter-Orb Protocol) protocols to facilitate information exchange among disparate applications [Wu He *et al*, 2014].

E. Data Integration

The Research on the data integration is mainly dealing with moving the data between disparate or homogeneous types of applications which can be data sources [Wu He *et al*, 2014]. Data integration involves a lot of data mapping and conversion among different elements including data source schema, targeted data schema, and the mapping relationship between data source schema and targeted data schema [Wu He *et al*, 2014]. The developers have to recognize and maintain the schemas frequently to address of any changes in data or schema. This is one of the drawbacks of data integration.

F. Business Logic Integration

Business logic integration is integration which includes sub layers such as business policies and protocol, functional interfaces, basic coordination and nonfunctional properties [Wu He *et al*, 2014]. The analysis of middleware technologies shows that, the most frequently middleware used at the business layer logic for integrating disparate applications which are geographically distributed. The some important middleware technologies are as follows:

- Distributed Computing Environment (DCE)
- Distributed Component Object Model (DCOM)
- Common Object Request Broker Architecture (CORBA)
- Java Remote Method Invocation (RMI)
- Message Oriented Middleware (MOM)

Each middleware technologies have pros and cons. DCE does not have strong support of object oriented language as RPC is popularly designed for procedural. DCOM that supports only non-window operating systems could not integrate applications for heterogeneous network.

CORBA is very difficult and complicated to use it, so that the interest of developer is minimizes and automatically its lost popularity. RMI is mostly depends on the Java which very difficult to work with other programming languages like C++. MOM could not support for scalability, portability and lack of industrial standard.

G. Recent Research in EAI

As market growing, everyday there is some research going in enterprise application integration because of its huge demand and huge amount of money has been invested in disparate and legacy systems. From developers perspectives following are some technologies plays important role in EAI.

- Java 2 Enterprise Edition (J2EE Spring)
- Microsoft .NET Framework
- Web services, Service oriented architecture (SOA) and Enterprise service bus (ESB)

Each technology has some merits and demerits. As gone through J2EE and Microsoft .NET framework has provided huge amount of interfaces for efficient deliver functionality to particular task but it come up with complexity because interfaces are tightly coupled with each other and which makes disparate applications to integrate using tightly coupled architecture.

In Web service, each function perform single entity which uses the HTTP protocol for the information transport and which can be easily passes through firewalls. It is generally used to develop web based applications which can be geographically distributed and it can be basic middleware applications to integrate web based applications. With the help of J2EE and .Net, developer can implement the web service making the integration is language dependent and huge dependency on interfaces provided by Java and .NET.

SOA is web service with its underlying principles. SOA is best solution for the middleware-middleware interaction [Wu He *et al*, 2014]. SOA is recent trend to integrate homogeneous and heterogeneous applications. SOA has been provided detail information about the service such as guideline and specification for the service description, discovery and about use of service. It will use the UDDI (Universal Description, Discovery and Integration) for guideline and specification of service. Each service contains the interface which described different operations (request and reply) and type of message which could handle. SOA integration is depends on the ESB. Enterprise service bus (ESB) is capable to work across different middleware products and standard to implement enterprise wide SOA [Wu He *et al*, 2014].

Web service, ESB and SOA has been provides the favorable and valuable framework for integrating disparate applications. Service oriented integration is depends on the ESB (Enterprise service bus). ESB can protect from different protocols such as J2EE RMI, IIOP, and CORBA etc. ESB will help for disparate applications to integrate with each other using loosely coupled architecture and has provided the smooth data flow between them [Wu He *et al*, 2014]. ESB has provided integration framework for developer to integrate disparate applications and which is based on SOA. Now a days unlimited amount of research is going in ESB, different vendors come up with own ESB for commercial purpose because of huge demand (IBM Integration Bus, Oracle Fusion, Tibco, Mule ESB, Software AG webMethods etc).

3. Proposed System

There are two proposed system. The first proposed system is web service aggregation scenario with domain Distributed Computing. The second proposed system is web service aggregation (generic) pattern with domain Software Engineering and Architecture. The pattern will generate the basic web service aggregation framework (interface flows) for integrating disparate applications. The proposed systems in detail are as follows

A. Web Service Aggregation Scenario: Distributed Computing

The web service aggregation scenario (Restful web service) is send request (same or different as per business logic) to multiple homogeneous and/or heterogeneous systems. Those systems are inside or outside of the organization and which accepts the request from middleware, process the request and gives reply back to middleware. The proposed middleware will aggregate replies of all systems into single reply and send to third party system (requested system). The proposed middleware is shown in following fig. 2

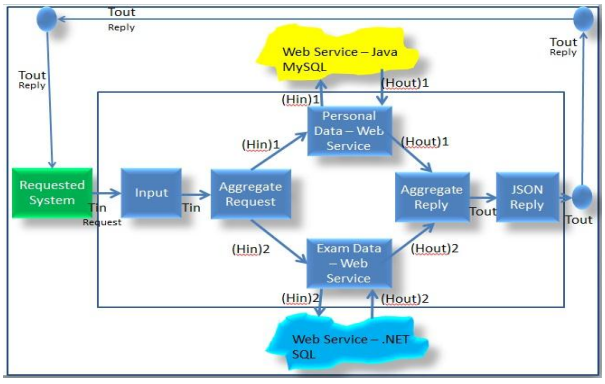

Fig. 2 Web Service Aggregation Scenario

For example, there are two systems which contain information of student: Personal data (developed in Java and MySQL on Linux platform) and result data (developed in .NET and SQL on windows platform). And requested system (third party system) need a data of students. So that require aggregation scenario which will aggregate data from both systems and reply aggregated data in single reply to requested system.

The web service aggregation scenario are

- Aggregation web service(WS)
- Personal information web service(WS)
- Student information web service(WS)

Any third party system send request to aggregation WS by providing roll number as input parameter and need personal and result data of student. Aggregation WS forwarding request to both personal and result WS to all information related to requested roll number. Personal and Result web service reply back all information of requested roll number. Aggregation WS aggregated replies into single reply and send back to requested system.

B. Web Service Aggregation (Generic) Pattern: Software Engineering and Architecture

The second proposed system is web service aggregation (generic) pattern. The pattern has provided because enterprise integration of disparate application is very complex task and to form uniformity in organization. The pattern will provide the guideline for developer and designer during integration of enterprise applications. They are accepted solutions to recurring problems within a given context.

The proposed GUI for web service aggregation pattern is shown in following fig. 3. With the help of generic pattern, develop the integration layer is to save money and time in future for electronic business by providing generic pattern to minimize efforts of developer to do work from scratch.

Web service aggregation pattern is provided following steps:

- Enter Number of systems (Heterogeneous + Homogeneous systems)
- Provide WSDL for the aggregation web service.
- Generate instance of Pattern.
- Provide WSDL for all external web services.
- Modify code as per your requirement.

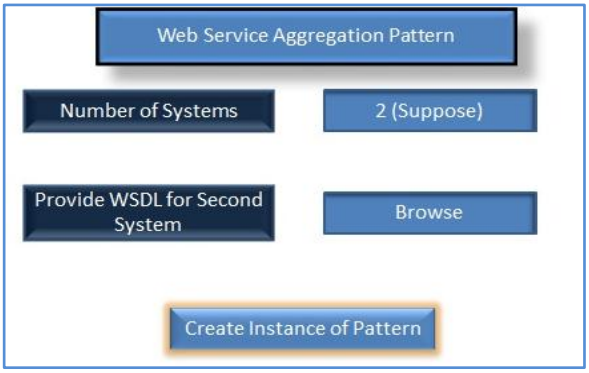

Fig. 3 Web Service Aggregation Pattern

The application of web service aggregation pattern (mentioned in fig. 3) is to generate generic framework for the web service aggregation scenario (mentioned in fig. 2). The developers only concentrate on the business logic because all basic things have been done by following standard and uniformity in recurring problems. Patterns are not invented, but rather discovered and observed from actual practice in the field. Each pattern poses a specific design problem, discusses the considerations surrounding the problem, and presents an elegant solution that balances the various forces or drivers.

4. Implementation Environment

The selected implementation environments for proposed system are:

- IBM Integration Bus 9.0.0.2
- IBM WebSphere Message Broker Toolkit 9.0.0.2
- IBM WebSphere Message Queue.
- Extended Structured Query Language (ESQL)
- Java (For Personal web service and Pattern)
- NET (For Result web service)

5. Mathematical Modeling

The mathematical model for proposed system is as follows. Before moving to the mathematical model, visualize all parameters like input, output etc. with the help of fig. 4 are as follows.

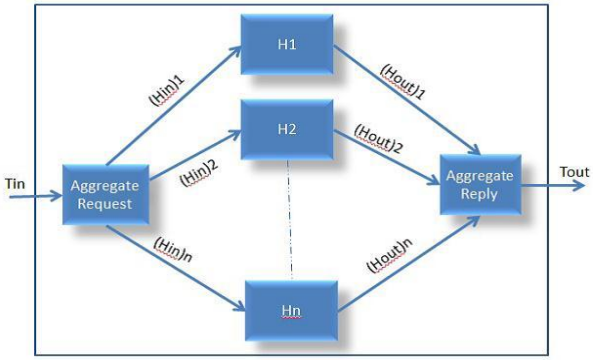

Fig. 4 Visualization of Mathematical Modeling

LetT_{in}: Service request (tuple) from requested service

H_i : Set of Heterogeneous/Homogeneous systems

$(H_{in})_i$: Request message to i'th system.

$(H_{out})_i$: Response message from i'th system.

T_{out}: Service response to requested system.

There are two possibilities:

1. Same request message is send to each heterogeneous/homogeneous system:

$$T_{in} = (H_{in})_i = (H_i)_2 = (H_{in})_3 = \cdots = (H_{in})_n$$

2. Send request message to each system after dividing as per system wants

$$T_{in} = (H_{in})_1 \cup (H_{in})_2 \cup (H_{in})_3 \cup \ldots \cup (H_{in})_n$$

The reply message will be:

$$T_{out} = (H_{out})_1 \cup (H_{out})_2 \cup (H_{out})_3 \cup \ldots \cup (H_{out})_n$$

The durable memory for the proposed system is calculated by considering all variables which are in queue, saved and out.

Let,

- T_A : Application durable memory.
- T_{Saved}: All the tuples under control of service.
- T_{queued}: All the tuples that are in queued for passing.
- T_{out}: All the tuples that are passed.

The total memory requires:

$$T_A = T_{Saved} \cup T_{queued} \cup T_{out}$$

6. Data Set and Results

The data set for proposed generic pattern first parameter is number of systems (homogeneous + heterogeneous) and the second parameter is WSDL for aggregated web service.

To use generic pattern for web service aggregation, first need to download into pattern view of IBM WMB toolkit with the help of 'Download' link. The pattern view is shown in following fig. 5.

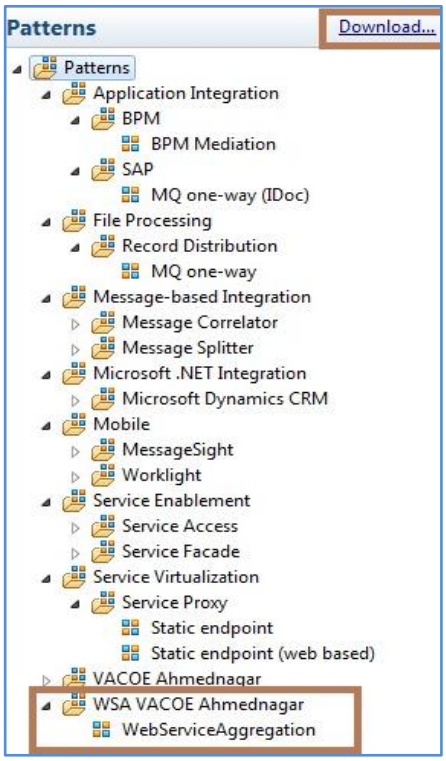

Fig. 5 IBM WMB toolkit pattern view

Require input pattern parameters are shown in following fig 6.

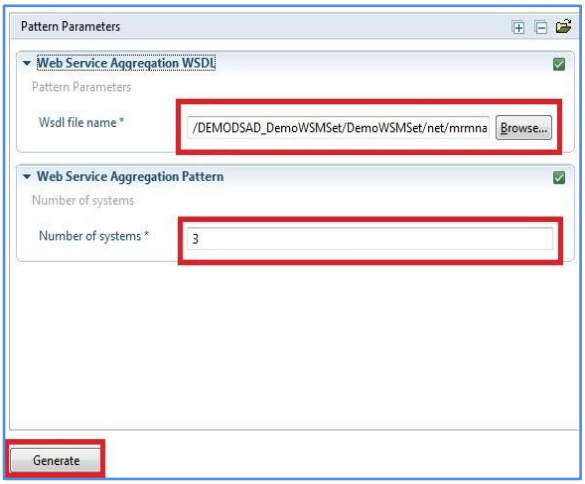

Fig. 6 Input Parameter for pattern

After providing values to both input parameters, 'Generate' pattern instance button is activated to create generic flows for aggregation scenario which is shown in following fig 7. (Suppose number of systems is 3)

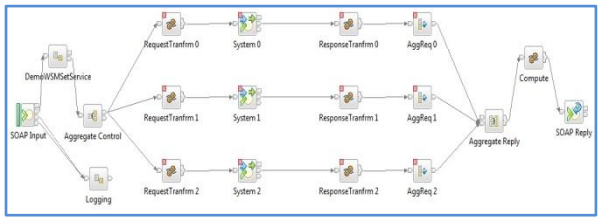

Fig. 7 Generated Flow using pattern (Result)

The flow has been generated as shown in fig 7. Now need to provide WSDL for each SOAPRequest node and change the business logic in compute node as per your requirement. This flow is generated in IBM Integration Bus 9.0.0.2.

The data set for proposed web service aggregation shown in fig. 2 is roll number of student. It will return personal and result information of requested roll number.

7. Discussion on Performance

For web service aggregation performance, there are many ways to consider performance parameter.

- Provider side parameter:
  - Latency: It is time between service request and service response.
  - Server throughput: It measures number of request process by server per second.
  - Throughput: It measures average byte of flows per unit of time including latency.
  - Error rate: It identifies depend abilities of service.
- Requestor side parameter:
  - Latency: Considering network latency plus web service processing time.

Network parameter need to consider:

- Number of hops between requestor and provider.
- Size of message
- Message rate

Above parameters are generic for all type of web services.

The following factors need to be considered for web service aggregation when developed using generic pattern.

- Web service uses aggregation scenario: Forwarding message to multiple system and aggregating replies into single reply.
- Flow is exposed as service: SOAPInput and SOAPReply node is used.
- Flow is calling one or more external web service: SOAPRequest node is used.

On the basis of following parameters the result can be calculated: Apache Jmeter is used to calculate results

- Number of requests (sample)
- Average : The average time of a set of results
- Median: Time middle of set of results.
- 90 % Line: Time at 90% samples process
- Min: The shortest time for samples
- Max: The longest time for samples
- Error %: % of errors with requests
- Throughput: Number of request process per second
- Kb/sec: Throughput measured in Kb/sec

By considering 150 samples (number of request) for results with the help of Jmeter as shown in following table and the graph:

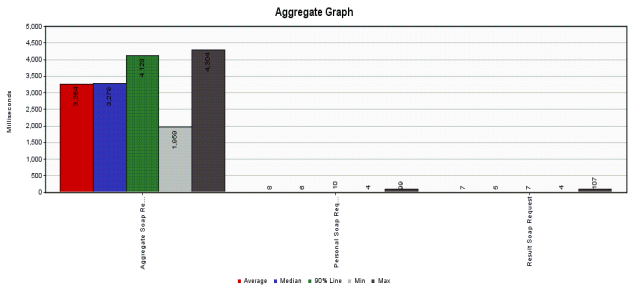

Fig. 8 Result using Apache Jmeter

Table 1: Result table

Name of Web Service(Samples)	Avg	Median	90% Line	Min	Max	Error %	Throughput	Kb/sec
Aggregate WS (150)	3264	3278	4128	1959	4304	0	27.1/sec	13.1
Personal WS (150)	8	6	10	4	99	0	46.7/sec	18.8
Result WS (150)	7	5	7	4	107	0	47.0/sec	18.8

Conclusion and future scope

The electronic business has achieved the loosely coupled architecture directly or indirectly using SOA. The disparate systems will efficiently communicate with each other using service oriented architecture by creating everything as service. The proposed integration layer will communicate with heterogeneous systems, take require information from those systems, aggregate information into single reply and send back to requested system. The second proposed architecture is generic pattern which is reusable solution to web service aggregation scenarios. EAI is very complex task. The pattern will generate generic flows for web service aggregation. They are accepted solutions to recurring problems within a given context. It will use to save money and time for electronic business by providing generic pattern to minimize efforts of developer to do work from scratch. There are several issues that could be seen in future: Security aspects of web services and Messaging Queue, Queue monitoring, web service monitoring, and logging/Auditing messages with sending email about failure messages etc.

Acknowledgment

I would like to sincere thanks to Government Polytechnic Ahmednagar, RO Nashik, DTE Maharashtra, Mr. M. C. Kshirsagar, My guide. Thanks for great help from beginning to end, My Family, Mr. Pradeep Patil (PP) and Mr. Prashant Patil (PP) andMy Friends.

References

Martin Potocnik, and Matjaz B. Juric (2014), Towards Comple Event Aware Services as Part of SOA, *IEEE Transaction on services computing,* 7(3) pp. 486-500.

Wu He and Li Da Xu (2014), Integration of Distributed Enterprise Applications: A Survey, *IEEE Transactions on Industrial Informatics*, 10(1), pp 35-42

Nilesh Vishwasrao Patil (2014), Pattern Authoring : Web Servie Oriented Architecture, *Impressco IJCET,* 04(5), pp 3143-3146.

Nilesh Patil, M. C. Kshirsagar, and P. C. Jaypal (2014), Enterprise Application Integration Using Service Oriented Architecture with Generic Pattern, *Impressco IJCET*, 04(5), pp 3540-3545.

Jitendra Joshi, Nisha Singh and Manjuri Kumara (2012), Web Service Oriented Architecture Modelling with Pattern for Electronic Business Organization, *IJARCSSE*, 02(09), pp 456-460.

Mohammed Said, Osama Ismail and Hesham Hassan (2013), Web Service Composition and Legacy Systems: A Survey, *IJCA*, 69(16), pp 9-15.

M.P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann (2007), Service-Oriented Computing: State of the Art and Research Challenges, *Computer*, 40(11), pp 38-45.

Z. Laliwala and S. Chaudhary (2008), Event-Driven Service-Oriented Architecture, *in Proc. Int. Conf. Service Syst. Service Manage*, pp 1-6.

Chunhua Gu, and Xueqin Zhang (2010), An SOA Based Enterprise Application Integration Approach, *in Third Int. Symposium on Electronic Commerce and Security.*, pp 324-327.

Xu He, Hongqi Li, Qiaoyan Ding, and Zhuang Wu (2009), The SOA-Based Solution for Distributed Enterprise Application Integration, *in Int. Forum on Computer Science-Technology and Applications*, pp 330-336.

Pramod Nutan Dhara, and Rishi Sayal (2012), Integrated Framework using SOA and Web Services, *IJARCSSE*, 02(11), pp 171-174.

Venkat N Gudivada and Jagadeesh Nandigam (2005), Enterprise Application Integration Using Extensible Web Services, *in Proc.of the IEEE ICWS.*

Michael N. Huhns and Munindar P. Singh (2005), Service-oriented computing: Key concepts and principles, *IEEE Internet Computing,* 9(1), pp 75-81.

S. Vinoski (2003), Integration with web services, *IEEE Internet Computing*, 07(6), pp 75-77.

Mike P. Papazoglou (2003), Service Oriented Computing: Concepts, Characteristics and Directions, *in IEEE 4th Int. Conf. Web Information Systems Engineering*.

Sukhpal Singh, and Inderveer Chana (2012), Enabling Reusability in Agile Software Development, *IJCA,* 50(13), pp 33-40.

S Chatla, S Kadam, D Kollaru, S Sinha, A Viswadhuni and A Vaidya (2011), Complex networks and SOA: Mathematical modelling of granularity based web service compositions, *Sadhana*, 36(04), pp 441-461,.

Mikhail Perepletchikov, Caspar Ryan, Keith Frampton, and Heinz Schmidt (2008), Formalizing Service-Oriented Design, *Journal of Software*, 03(02), pp 01-14.

Dustdar, S., Wolfgang S. (2005), A survey on web services composition, *International Journal of Web and Grid Services*, 01(01), pp 01-30.

Sneed, H. M. (2012), Integrating legacy software into a service oriented architecture, *in CSMR*, pp 01-14.

S. Vinoski (2003), Integration with web services, *IEEE Internet Comput.*, 07(06), pp 75-77.

K. Qureshi (2005), Enterprise application integration, *in Proc. IEEE 2005 Int. Conf. Emerging Technologies, Islamabad, Pakistan,* pp 340-350.

D. S. Linthicum (2000), Enterprise Application Integration. Essex, U.K.: Addison-Wesley Longman Ltd.

B. Benatallah and H. R. Motahari-Nezhad (2008), Service oriented architecture: Overview and directions, *Advances in Software Engineering, A. Ferro and E. Boerger, Eds.*, 5316, Lecture Notes in Computer Science, pp 116-130.

S. Izza (2009), Integration of industrial information systems: From syntactic to semantic integration approaches, *Enterprise Inform. System,* 3(1), pp 1-57.

F. Daniel, J. Yu, B. Benatallah, F. Casati,M. Matera, and R. Saint-Paul (2007), Understanding UI integration: A survey of problems, technologies, and opportunities, *IEEE Internet Computing,* 11(2), pp 59-66.

WhebSphere Message Broker V8.0 For Window 64-bit Performance report V1.0 April 2012.

Web Service performance study, GANG DU Msc Information Systems 2004.

Website links:

- http://www.eaipatterns.com/eaipatterns.html
- http://www.ibm.com/developerworks/websphere/library/techarticles/1010_stewart/1010_stewart.html
- http://www.perfectxml.com/Integration.html
- http://www.tutorialspoint.com/webservices/web_services_characteristics.htm
- http://publib.boulder.ibm.com/infocenter/wsdoc400/v6r0/index.jsp?topic=/com.ibm.websphere.iseries.doc/info/ae/ae/cwbs_soawbs.html
- http://www.mulesoft.com/resources/esb/enterprise-application-integration-eai-and-esb
- http://www.w3.org/TR/wsdl
- http://www.tutorialspoint.com/wsdl/wsdl_elements.htm
- http://paulstovell.com/blog/integration/messaging

Biography

Nilesh Vishwasrao Patil is pursuing Master of Engineering with specialization in Computer Engineering from Vishwbharti Academy College of Engineering Ahmednagar under Savitribai Phule Pune University, Pune. He has more than five years of experience, in which around two years of industrial experience as Enterprise Application Integration (EAI) developer. Now he is working as System Analyst in Government Polytechnic, Ahmednagar since November 2011. He has published four papers in international journals (Apache Hadoop – Resourceful Big Data Management, Dynamic Memory Allocation: Role in Memory Management, Pattern Authoring: Web Service Oriented Architecture and Enterprise Application Integration Using Service Oriented Architecture) in reputed journal and presented two papers in National conference.

M. C. Kshirsagar is working as assistant professor in Vishwbharti Academy College of Engineering Ahmednagar in Computer Engineering Department. He has completed Master of Engineering in Computer.