I.J. Information Technology and Computer Science, **2014, 02, 64-71**
Published Online January 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2014.02.09

Grid Approach with Metadata of Messages in Service Oriented Architecture

Jamal, S. M. Khalid
Department of Computer Science, University Of Karachi, Karachi, Pakistan
E-mail: s.m.khalid@uok.edu.pk

Asif, M.
Department of Computer Science, University Of Karachi, Karachi, Pakistan

Abstract— Service Orientation Architecture is playing a vital role in the middleware and enterprise organizations through its key design principles and there are many benefits of it as well. Loose Coupling between services is one of its design principles that help system or architecture to maintain its efficiency because services are less dependent on each other. Stopping of one service could not affect other service and system but there is a fact that services have less knowledge of other services that's why services cannot be able to properly communicate with each other and Service Oriented Architecture is totally based on communicative messages between services in order to perform their functionality. This paper provides a basic solution to control the messaging criteria between services. This paper holds the brief description of messaging criteria of services in service oriented architecture by defining the process of message that fulfills the guaranteed delivery of message to its other end while communicating or asking for a service.

Index Terms— Service Oriented Architecture, Metadata, Loose Coupling, Guaranteed Delivery of Messages

I. Introduction

The purpose of establishing Service Oriented Architecture (SOA) is to provide an architectural model to pinpoint services as key functionality that boosts efficiency and abundance of an organization [1-5].

Primary concept of SOA is interrelationships of service-orientation design paradigm with service-oriented architecture are key concept in the region of service oriented computing platform. If SOA is discussed in terms of technology architecture, it is clear that service oriented architecture can be a fusion of technologies, Application Programming Interface (APIs), products, sustaining road and rail network and much more. The dispose of SOA is exclusive in respective enterprise, by introducing further new technologies that is the cornerstone of creation, implementation and development of service-oriented solutions [6-7]. Creating technology architecture by using Service Oriented Architectural Model carries out an upbringing that is appropriate for solution logic while considering Service Oriented Design Principles as its primary result.

1.1 Overview

For interpretation and analysis of detail description of service oriented computing, it is more preferable to understand design phrasing [8-11]. Following sequence follows the design phrase:

1.2 Design Characteristics

Characteristics generally referred to as an attribute or essence of something. The design characteristic is an explicit aspect or essence of an anatomy of solution logic that is to be documented in a particularization of design and map to grasp development.

Service oriented architecture accentuates the creation of specific design characteristics. It is to be noticed that each explored design characteristic is accomplishable to definite dimensions.

Notice that almost every explored Characteristic of Design is achievable to a definite measure. It does not mean that logic of the solution either has definite characteristic or not; it's about to which degree a Design characteristic be accomplished.

Characteristics of each system can be rare but our primary goal is to implant familiar design characteristics. As familiarity increases, it tends to increase consistency rate and try to make different kinds of logic solutions more concurrent. More concurrency means more predictable. Predictable design characteristics directed towards predictable behavior.

1.3 Design Principle

A principle is a generalized, accepted industry practice. Or it can be said that a common objective lead to something that others are doing or promoting in association.

A design principle is representation of highly suggested guideline to assemble logic of a solution in an assertive way and with assertive goals in mind. One or more design characteristics are inaugurated which are ordinarily associate with these goals.

1.4 Design Paradigm

Term "Paradigm" has different meanings in different aspects. It can be a pathway to achieve some goal or it can be a sequence of rules that are adapted by a predefined boundary. Object-oriented design is a classic example of an accepted design paradigm.

Service orientation has its own design paradigm sketch. Similar to Object-Orientaton, Service - Orientation is a paradigm that is implemented to distributed solution logic, but its principles are different from those which are associated with object-orientation in the result of which different types of design characteristics are created.

1.5 Design Pattern

From detailed discussion we have concluded that service orientation is a design paradigm incorporate certain design rules, all of these accommodate a generalized rule that is realization of certain design characteristics. For successful implementation in actuality, thoughtful of its principles required.

While applying design paradigm in the real world, service designers will have to face different interruptions and difficulties. As because the recognition of the preferred Design Characteristics is convoluted by different aspects, which includes:

1) Technology that is using to build the units of logic forced some Constraints.
2) Organized units of logic of the solution next to which technology exists forced some Constraints.
3) Few constraints are forced by units of solution logic based on requirements and priorities of project.

Design patterns characterize the problems along with accommodation of analogous solution. For repeated application it fundamentally records the solution in a comprehensive format. Design patterns are a way of acknowledgment that how these problems are best deals with. To solve large scale problems, design patterns can be joined together to develop the root of pattern language or it can be further divided to develop a series of patterns.

1.6 Design Standard

Applications of Design Standards form the design characteristics with the help of Design Principles. To steer clear of the latent issues and to increase the strength of Design Solutions, Design standards promote and purify Design Characteristics with the help of Design Principles.

1.7 Best Practice

Best practice is usually a consideration of an approach that is followed to solve or prevent some problems. From industry's prior experiences, it is usually proceed.

The main difference between design principle and best practice are very precise. Design principle revolves around the design while best practice can be linked to anything in project delivery to organizational issues. A design principle could be consideration of a best practice when they relate only with solution design.

1.8 Service Oriented Design Principles

Service Oriented Architecture is the architecture that holds the mechanism of communication through messages between the services. To understand this point, let's have a look and understand the following design principles of Service Oriented Architecture:

1) Services Communication Agreement
2) Minimization of dependencies
3) Services Hide Logic.
4) Different Services aim to reuse
5) Service control on encapsulated logic
6) Delay of management information for service minimization of resource consumption
7) Addition of Communicative meta data
8) Service Consist of capable composition participants.

(1) Services Communication Agreement

Service Contract conveys the goal and potential of Services. Services are bound with an agreement that is a collection of one or more described document of services.

(2) Minimization of Dependencies

Relationship between two things is known as coupling. Measure of coupling and level of dependency are comparable components. This is the assumption which is defender in account of creation of precise relationship inside or outside service boundaries along with dependencies between the service contract reduction emphasis and its implementation including its service consumers. Service Loose Coupling assumption is basically a recommendation of the absolute design along progression of service logic and implementation although baseline interoperability are still guaranteeing including service's capabilities on which user depends.

(3) Services Hide Logic

This principle enforces need of hiding most of the elementary details as much as possible. This will lead to the preservation of previously described loosely coupled relationship and also enables directly. Service compositions are designed and positioned using service abstraction. While estimating different abstraction levels various forms of Meta data are under consideration.

(4) Different Services Aim to Reuse

When the term service orientation is use reusability is always part of its consideration. Reusability becomes a pivotal component of typical service analysis and design processes.

(5) Service Control on Encapsulated Logic

Services underlying solution logic requires a noteworthy level of control over its resources and environment to bring out their capabilities constantly and unfailingly.

(6) Delay of Management Information for Service Minimization of Resource Consumption

Main requirement is that services should remain state full only upon some requirement. For application of the principle it is prerequisite that measures of realistically attainable statelessness, making capabilities of surrounding technology architecture while considering state management delegation and deferral options as its foundation.

(7) Addition of Communicative Meta Data

By communication of Metadata between the services, they can be efficiently discovered and take to mean.

Regardless of having a service registry as a discovery mechanism an instantaneous element of the system, Service Design has to obtain the Quality of Communication and its capabilities under explanation.

(8) Service Consists of Capable Composition Participants

Regardless of the convolution and size of the Composition, Services are efficient applicant of Composition. This design principle is a utility for conclusion to carry out a severance of concerns in support of service-orientation. To solve a problem, the exemplify decomposition of solution of the logic are accumulated [24-30].

As far as this paper concerned, the primary focus is on one of the design principles which states that Service Oriented Architecture is loosely coupled in nature. Coupling termed as the Level of Dependency between the Services. In Service Oriented Architecture, services are less dependent on each other to increase the efficiency and effectiveness of the system. Requesting a service by Service Requestor to Service Provider requires a messaging framework that is loosely coupled.

"Despite the benefits, loosely coupled framework is being managed with less control over communication process, because of this unreliable & less controlled communication procedure the status of the message is not updateable among the service Requestor and Provider."

As the communication between Service requestor and service provider holds the mechanism of messaging framework that is loosely coupled, there is no way of knowing the message status between the two. This will yield in a confusion that either the message transmits by Service Requestor is delivered to Service Provider or not.

The approach to introduce a third party situated among Service Requestor and Service Provider to govern the status of the message which has to be sent or received will increase the Message Overhead.

To overcome the above problem, one can avail the concept of Grid Computing. Grid holds the Metadata of the messages that are sending and receiving. The Metadata of the message will update in Grid after the successful delivery of the message, acknowledging the Sourced Service that the message is successfully sent to its destination address.

Metadata contains a full definition of Message that has received by the Service Provider. Metadata describing a message typically contains the details of the interface of the message including identifier of the Service Requestor and Service Provider, format of requested message, format of responded message, message internet address, length and content of the message.

Strength of the approach is that it will reduce the Message overhead in Service Oriented Architecture. This paper gives the better solution of messaging criteria in loosely coupled services.

The remainder of this paper is organized as follows: Section 2 gives detail description of messaging problem. Section 3 describes Grid mechanism. Section 4 describes Metadata of Message. Section 5 describes Messaging Criteria. Section 6 describes Grid mechanism regarding Service Provider. Section 7 describes Grid mechanism regarding Service Requestor.

Section 8 demonstrates Performance Measures. Section 9 gives the formulae. Conclusion and future work are given in the final section.

II. Related Work

Service Oriented Architecture consists of exchanging of messages detail from business rules or logic. This can be done by exchanging business documents.

The concept of WS-Reliable Messaging for guaranteed delivery of the message between services is maintained through SOAP Messages which carries message header, functionality of messages in pieces security etc with it [12-17]. Functionality of messages (pieces) is not dependent on each other. The functionality is defined in below Figure:

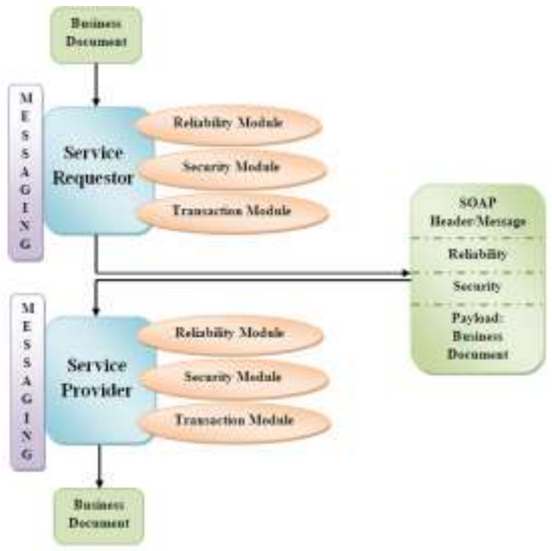

Fig. 1: SOAP Messaging Criteria

As describing in above picture:

1) Layers of Messaging and Business are not dependent on each other.

2) Functionality of message in terms of pieces is not dependent on each other.

3) Functionality of message when required carries information through message headers.

Let's have a look in below scenarios of message sending and receiving.

2.1 First Scenario

First Scenario illustrates that Message is not delivered to its Recipient and therefore obviously no acknowledgement of it is received. The Message is therefore send again successfully and acknowledgement of the message will be received by its sender. In this Scenario resending of the Message works.

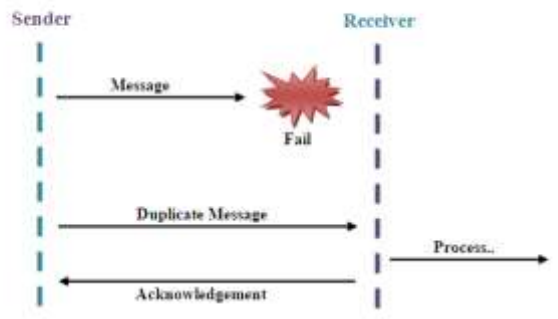

Fig. 2: Resending of a Message works

2.2 Second Scenario

Second Scenario illustrates that Message is successfully delivered to its Recipient but acknowledgement of the message is failed to reach its destination. The Message is therefore send again successfully and acknowledgement of the message will be received by its sender. In this Scenario, Receiver work on a same message twice a time.

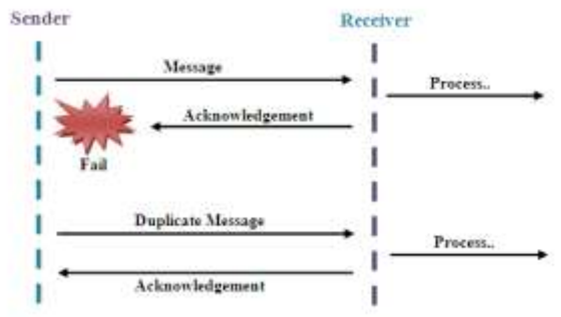

Fig. 3: Receiving of a same message twice a time

2.3 Third Scenario

Second Scenario illustrates that Message is successfully delivered to its Recipient but acknowledgement of the message is failed to reach its destination. The Message is therefore send again successfully but as the same message was already processed by the Receiver, the duplicate message is discarded and acknowledgement of the message will be received by its sender. In this Scenario, Receiver will discard the duplicate message and this is the criteria of Reliable Messaging. In short, Sender will not be serviced in this scenario.

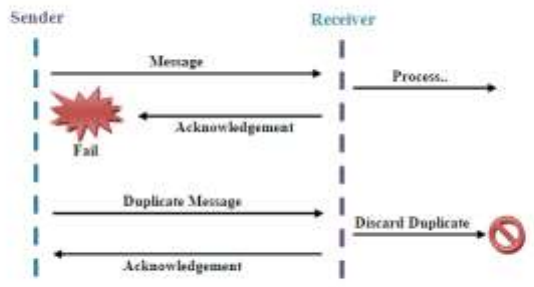

Fig. 4: Reliable Messaging Solution

III. Message Metadata in Grid

To overcome the problem of message between the Service requestor and Service Provider, use the term Metadata with the Grid. The Messaging Framework has to have a form of storage area like Table/Grid-view. When a Service Requestor sends a message to Service Provider, the following three will result:

1) Message successfully reaches to its destination.

2) Message doesn't reach to its destination.

3) Message sends twice (Duplicate Message).

After the successful delivery of the Message to its other end, its metadata will store in the Grid-view along with its content and the identifier of Service Requestor and Service Provider. It is a responsibility of Destined Service to update the Grid-view with the metadata of the Received Message. The Sourced Service will acknowledge the status of message according to the updated Grid-view. If the metadata of the message not store in the Grid-view, the message is not deliver to its destination and if the metadata information of the message in the Grid-view repeats, it is clear that a message has been sent twice.

IV. Metadata of the Message

Metadata is the data about data.Here the data is 'Message' which has to be sent to either end and its information is metadata. Data (Message) information includes [24-27]:

1) Format of the Requested Message

2) Format of the Responded Message

3) Length of the Message

4) Contents of the Message

5) Identifier of the Service who sends the Message.

6) Identifier of the Service who receives the Message

The above information represents the Message to its optimum level and it can be easily managed and stored in the Grid located in any local storage area where the Service Oriented Architecture has to be implemented. The Grid-view contains above attributes like below:

Table 1: Message Metadata in Grid

MsgID	Format Req-Msg	FormatRes-Msg	Len-gth	Con-tent	ID Sender	ID Receiver
M1	Demand for a book	Book issued	10Bytes	Send me the book	Service A	Service B
……	……	……	……	……	……	……
……	……	……	……	……	……	……
……	……	……	……	……	……	……

V. Message Sending and Receiving Criteria

When a Service Requestor request for a Service and send the message to Service Provider, the Service Provider will update the Grid with the information of the Received Message immediately after getting the message. As the coupling between services is loose in nature, updating the received message in Grid will not require any information of the Sender except its identifier and Service Requestor will acknowledged by viewing its sent message metadata in Grid-view[18-23].

The following figure illustrates the Messaging Criteria between the Service Requestor and Service provider:

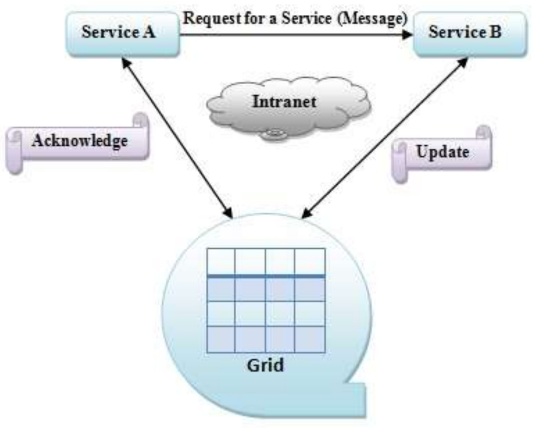

Fig. 5: Messaging Criteria

VI. Service Provider Updates the Grid

Immediately receiving the message, the Service Provider fills up the record of the message in the Grid. Every record which entered in the Grid assigns an

Identifier to each record. Unfortunately, if the Service Provider not be able to receive any message, the Grid will not be updated. Similarly, if the Service Provider receives a message more than once, the same metadata will be updated again.

Grid can be located in any local storage media or where suitable to manage by the organization who is going to implement Service Oriented Architecture.

VII. Service Requestor Checks the Grid

Soon after sending of the message, the Service Requestor continuously checks the Grid for the time being through the attribute – "Identifier of the Service who sends the Message". If the metadata of its sent message found in the Grid, the Service Requestor acknowledged the successful delivery of its message. If not found, it is clear that the delivery of the message to its other end fails.

If the record found more than once, it is clear that the message has sent twice of the time. Now, it's up to the Service Requestor that it wants to discard duplicate message or not. If it is willing to discard the duplicate message, the Service Requestor can send the message again to tell the Service Provider about to discard the duplicate one.

VIII. Performance Analysis

Some factors that are dependent in order to know the performance criteria of Reliable Messaging and solution of Messaging Criteria given in this paper are as follows:

1) CPU Utilization

2) Network Utilization

3) Throughput

8.1 CPU Utilization

CPU Utilization is the average amount of time in which CPU is busy in order to fulfill the message passing between the services. For Example, if the CPU is busy for 0.5 seconds overall out of 1 second communication of services, it's mean that CPU Utilization is 50% for Messaging Criteria between the services.

8.2 Network Utilization

Network Utilization is the average of the total number of bytes sent and received for the communication purposes between the services.

8.3 Throughput

Throughput is the average of time in which Service Requestor is being served immediately after the request issued.

Above mentioned are the dependent variables in order to know the relative cost of the said approaches.

8.4 Formulae for the General/Proposed Execution

Following formulae are the basic representatives of the performance of CPU Utilization, Network Utilization and Throughput.

IX. Formulae

9.1 General Formulae

$$\text{CPU Utilization} = 100\% \\ - \%\text{ of CPU idle time} \tag{1}$$

$$\%\text{ of CPU idle time} = \frac{100\% \;*\; \text{Average time period of executing processes without any load}}{\text{Average time period of executing processes with some load}} \tag{2}$$

9.2 CPU Execution Time with WS-ReliableMessaging

$$\text{CPU Utilization}(\mathbf{x}) = 100\% \\ - \%\text{ of CPU idle time} \\ - \%\text{ of CPU Utilization by WS} \\ - \text{ReliableMessaging} \tag{3}$$

9.3 CPU Execution Time with Paper's Adopted Solution

$$\text{CPU Utilization}(\text{y}) = 100\% \\ - \%\text{ of CPU idle time} \\ - \%\text{ of CPU Utilization by Non} \\ - \text{Parsing Grid Approach} \tag{4}$$

9.4 Network Utilization

$$\text{Network Utilization} = \left(\frac{\left(\begin{array}{l}\text{Message Sent}\\ +\text{Message Received}\end{array}\right)}{\text{Bandwidth}} * \text{Time in seconds} \right) * 100 \tag{5}$$

9.5 Throughput

$$R = \frac{I}{T} \tag{6}$$

Above formulae are used to analyze the performance of CPU by determining the CPU Execution Time for the WS-Reliable Messaging through **(3)** that gives a value **(x)** and CPU Execution Time for the paper's solution through **(4)** that gives a value **(y).**It is observed that the value of **x** is greater than value of **y** that proves the CPU Utilization in the paper's solution is decreased and performance is comparatively increased. The value of x is greater than y because:

Reliable Messaging →Parsing is done at sender side to convert it in SOAP Message in order to maintain the guaranteed delivery of messages between the Service Requestor and Service Provider.

Paper's Solution →Approach adopted in this paper doesn't require parsing (which is previously being done by "WS-Reliable Messaging"), so this approach essentially reduces the CPU Utilization as well as Network Utilization by approximately 50%.

(5) Can be used to determine the Network Utilization where:

Message Sent → Size of data in bits that has to be sent in a specified time period.

Message Received →Size of data in bits that has to be received in a specified time period.

Bandwidth → Band-width of Network Card

Time in seconds → The Duration time period in seconds of test to calculate the number of bits that is send and/or received.

Similarly, **(6)** can be used to determine the Throughput (Little's Law) where:

I →Number of packets in the System

T →Overall Flow time of packets

R →Throughput Rate

The table below gives the summary of the approximate analysis:

Table 2: Performance Analysis Summary

Factors/Dependent Variables	Reliable Messaging	Paper's Solution
CPU Utilization	50% - Because of parsing of a Message	25% - No Parsing
Network Utilization	50% - Because of the creation of SOAP Envelopes	25% - No need of making Envelopes
Throughput	25% - Less Efficient –takes time for parsing or for SOAP Messages	75% - More Efficient

By Analyzing Table 2, following formula is constructed:

$$\begin{aligned}\text{Performance } &= \alpha(\text{CPU Utilization})\\ &+\beta\ (\text{Network Utilization})\\ &+\text{Throughput}\end{aligned} \tag{7}$$

Where α and β are the weighing factors which are applied on it according to the business rules and resources of the Organization that is going to implement Service Oriented Architecture. If the CPU Utilization Value is greater than Network Utilization Value then α is greater than β and vice versa.

X. Conclusions

It is concluded that with the help of Grid and Metadata of the Message, the loose coupling of services that results in no notification of message that whether it is deliver to it's either end or not can be handled in easiest manner with no message overhead. An organization that following Service Oriented Architecture easily maintains Grid with their business rules is the best part of this approach.

To suggest the better solution of handling guaranteed message delivery between the services in Loosely Coupled Service Oriented Architecture is the future work recommendation of this paper.

References

[1] Mike P. Papazoglou · Willem-Jan van den Heuvel: Service oriented architectures: approaches, technologies and research issues

[2] David S. Frankel: Model Driven Architecture: Applying MDA to Enterprise Computing, Lead Standard Architect – Model Driven Systems – SAP Labs

[3] Grace A. Lewis, Dennis B. Smith and Kostas Kontogiannis: A Research Agenda for Service-Oriented Architecture (SOA): Maintenance and Evolution of Service-Oriented Systems (2010)

[4] Design Factors for Service-oriented Architecture Applied to Analytical Information Systems: an Explorative Analysis - 17th European Conference on Information Systems

[5] ManojMansukhani: Service Oriented Architecture White Paper (2005)

[6] Service Oriented Architecture (SOA) and Specialized Messaging Patterns – Technical White Paper

[7] RajuAlluri: SOA Adoption Challenges

[8] Thomas Erl: Service-Oriented Architecture: Concepts, Technology, and Design

[9] Thomas Erl: SOAPrinciples of Service Design

[10] Service-oriented architecture, http://en.wikipedia.org/wiki/Service-oriented_architecture

[11] Service Oriented Architecture (SOA),http://www-01.ibm.com/software/solutions/soa/

[12] Service Oriented Architecture : SOA Features and Benefits, http://www.opengroup.org/soa/source-book/soa/soa_features.htm

[13] Understanding Service Oriented Architecture, http://msdn.microsoft.com/en-us/library/aa480021.aspx

[14] Service Oriented Architecture (SOA) Definition, http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html

[15] SOA Manifesto, http://www.soa-manifesto.org/

[16] Service Oriented Architecture, http://www.webopedia.com/TERM/S/Service_Oriented_Architecture.html

[17] Metadata-Driven Application Design and Development, http://msdn.microsoft.com/en-us/library/aa480019.aspx

[18] Michael Niemann, Christian Janiesch, Nicolas Repp, and Ralf Steinmetz: Challenges of Governance Approaches for Service-Oriented Architectures

[19] Rapid, cost-effective deployment of data services in a seRvice-oRientedaRchitectuReInformatIon you need, from data sources you have, for applIcatIons that run your enterprise

[20] Fang Li and Xuanjing Huang: An Intelligent Platform for Information Retrieval

[21] Cloud Computing Vs. Grid Computing by SeyyedMohsenHashemi,AmidKhatibiBardsiri

[22] Jian Yu ·Quan Z. Sheng ·Yanbo Han: Introduction to special issue on cloud and service computing

[23] Dean of the Software Engineering and Artificial Intelligence Department , Science and Research Branch, Islamic Azad University, Tehran, IRAN

[24] An Oracle White Paper –Metadata Services in Fusion Middleware 11g

[25] Owen Conlan and Vincent Wade: Evaluation of APeLS - An Adaptive eLearning Service based on the Multi-model,Metadata-driven Approach

[26] Peter Conner et al: Service Oriented Architecture (2006)

[27] International Journal of Geographical Information Systems, Improving locational specificity of map data—a multi-resolution, metadata-driven approach and notation (1996)

Authors' Profiles

Jamal, S. M. Khalid: S. M. Khalid Jamal is an Academic Scholar, Researcher, Faculty Member and Information Technology & Services Consultant and is associated as Assistant Professor with Department of Computer Science, Karachi University (the largest and primeval Higher Education institution of Pakistan) since 2001. He is currently author of various research publications at National & International level.

Asif, M.: Post-graduate student for masters' degree for computer science in University of Karachi.

How to cite this paper: Jamal S. M. Khalid, Asif M.,"Grid Approach with Metadata of Messages in Service Oriented Architecture", International Journal of Information Technology and Computer Science(IJITCS), vol.6, no.2, pp.64-71, 2014. DOI: 10.5815/ijitcs.2014.02.09