An Effective Approach to Web Services Composition When Large Scales of Data Flows are Available

Azam. Andalib, Shahriar. Mohammadi, and Mehregan. Mahdavi

Abstract— Undoubtedly, Web is one of the most significant technologies in the last decade that provides a wide information publication and has made it possible to present services via computers in the whole world. Also, it is considered as one of the main factors of entering computers to the daily life of humans. The web services are the applicable programs that are accessible via web, by human being and other programs, independent from programming language. Moreover, they can be located and invoked through web. Development of providing services by institutions and organizations via internet has increased the demand for commercial interactions and links (especially Business to Business-B2B interactions) and the new technologies like web services are presented to make these links. With improvement of web services and development of their usage, the new concepts regarding the combination of simple services to make a complicated one, have been raised. Our research is concerned with developing an efficient model for composing web services when large scale data flows are available.Our goal is to enhance the potential of web services by focusing on new aspects of their composition by using of Deputy Servers.

Index Terms—web services, web services composition, Deputy Server, Orchestration

1 Introduction

Nowadays, the main usage of World Wide Web is to have an interactive access to the documents and applicable programs[1]. Web services (also called simply services) are self-describing, platform-agnostic computational elements that support rapid, lowcost and easy composition of loosely coupled distributed applications [9]. From a technical standpoint, Web services are modular applications that can be described, published, located, invoked and composed over a variety of networks (including the Internet): any piece of code and any application component deployed on a system can be wrapped and transformed into a network-available service, by using standard (XML-based) languages and protocols (e.g., WSDL, SOAP, etc.). The promise of Web services is to enable the composition of new distributed applications/solutions: when no available service can satisfy a client request, (parts of) available services can. If the web develops in order to support the links between Web Services, it will considerably improve in terms of range and power.

The composition of web services to handle crackly transaction like as finance services is gaining considerable momentum as a way to enable business-to-business (B2B) collaboration [13]. Web services allow organizations to share costs, skills, and resources by joining their applications and systems [3].

World Wide Web was proposed by Tim Berners-Lee in 1989[3] in order to publish the web pages and make links between them and it has unbelievably expanded during these years [2]. Web services, are web based applicable programs that could be established or invoked independently by software or other services [4]. Service Oriented Architecture (SOA), has facilitated the procedure of flexible and loosely coupled programming, hence this architecture as a fundamental factor for interoperability will be discussed[5]. Infact the main and basic services of organizations are available in the form of Web services through their portal for their visiting users. Ease of service combination and combined construction of service applications is discussed as basic a feature in service oriented architecture [15].

The purpose of this paper is to emphasis on the requirements of applying the web services in order to achieve a more valuable goal. Then we explain current solutions for compositing web services briefly to achieve this goal and finally we introduce a new model for improving implicit methods.

2 Introduction To Web Services

Every service that is accessible via Internet, uses the message transferring system based on the XML (Extensible Markup Language) standards and does not depend on an operating system or a programming language is a Web service [7].

The World Wide Web Consortium which is the prestigious reference in Web, defines the Web services as fol-

- *A.Andalib –Department of It, University of Guilan , Rasht and Department of SE, Islamic Azad University-Roudsar Branch, Roudsar, Guilan, Iran. E-mail: azam_andalib@ yahoo.com.*
- *S.Mohammadi–K.N.Toosi University of Technology,Tehran, Iran.E-mail: smohammadi40@yahoo.com.*
- *M.Mahdavi– Departmant of Computer Science and Engineering. University of Guilan, Rasht, Guilan, Iran. E-mail:mehregan.mahdavi@gmail.com.*

lows: "a Web service is a software system that is designed to protect machine to machine interactions on a network and have a definition which is processed by a machine called WSDL (Web Services Description Language). This service has an interface that is presented as an understandable language for the machine. The other systems make connection with this service by the method which is defined in this interface, and through sending message" [8]. In other words, a Web service is a software service that is realized with a URL (Uniform Resource Locator) and its public interfaces are described and identified by using XML.

Generally speaking, using Web services can cause a general advantage which is no need to coding again. Although, this advantage is for using the classes too, but in using the classes in every project they need to be added, while in Web services the methods can be applied. By working with classes in various applications with several methods, it is possible to work with different classes. In other words, the work that is done is different in every time which reduces the legibility and also makes the development difficult. But, by using Web service, every time an especial Web service according to a specific Web service's method is used, regardless of being in any application [7].

3 Web Services Composition Models

IT organizations need the versatility to reconcile to customer requirements and changing market situation. But existing solutions do not directly support Web services standards and, as a result, IT organizations may be captivated to take a short-term solution and create their own proprietary protocols for composing services. Web services orchestration, decentralized orchestration and choreography standards are efforts that can be long-term solutions for business communications [11]. By connecting services through open, standards-based methods, organizations withhold themselves the burden of maintaining those proprietary interfaces.we explain these standards and at the end of this article we will introduce a new efficient method for web services composition [14].

3.1 Orchestration Model

Orchestration method is an executable business process [10] that may communicate with both internal and external Web services. Service orchestration is a centralized approach which insulates control and data flow. We need Control flow to control/orchestrate the workflow, whereas data flow relates to the the tasks that compose the actual applications. It describes how Web services can communicate at the message level, including the business logic and execution order of the communications [18]. These interactions can span applications and/or organizations, so they result in a long-lived, transactional process. With orchestration, the process is always controlled from the viewpoint of one of the business parties. In this model, all communication is directed via a central process (workflow engine) for both the control and data flow [17].

In the Web service world, Web Services Business Process Execution Language (WS-BPEL) [19] has become the de facto standard for orchestration. With WS-BPEL, the workflow can be specified without the need to accommodate any of the services, with help of the central process providing the workflow logic.

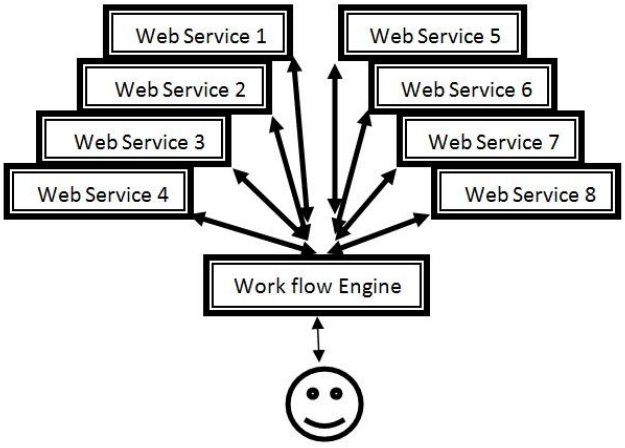


Fig. 1. Orchestration Model

3.2 Choreography Model

This method is more collaborative in nature. Each party describes the part they play in the interaction. Choreography tracks the sequence of messages that may involve multiple parties and also multiple sources. It is done with the public message exchanges that tide between multiple Web services [19].

This decentralized solution does away with the centralized process and instead each collaborating service is aware of its portion in the workflow. In Service choreography, the collaborating services exchange messages in order to coordinate execution of the workflow. We should notice that in order for this collaboration to take place, the Web services need to be corrected so that they are aware of the related workflows. The characteristics have been put forth for Web service choreography in the Web Services Choreography Description Language (WS-CDL) [21]. This method has so far not been widely used, nor are there many implementations of the specification.

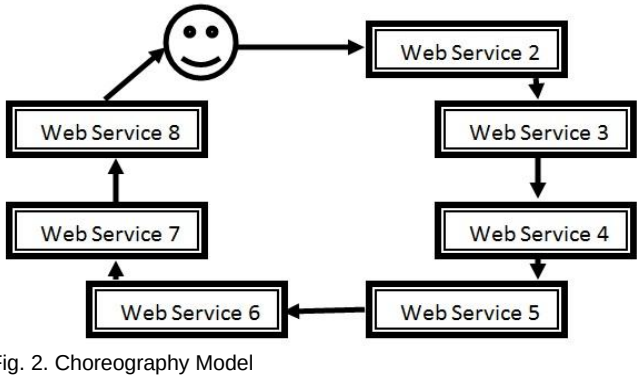


Fig. 2. Choreography Model

Orchestration differs from choreography. it describes a

process flow between services, controlled by a single party but, choreography tracks the sequence of messages involving multiple parties, where no one party truly "owns" the conversation.

3.3 Decentralized Orchestration

Decentralized Orchestration is another solution for being used by web services. In this model, a centralized workflow is analyzed and divided into smaller workflows [18]. Each workflow engine orchestrates its own partition of the workflow.

We need multiple workflow engines to execute the partitioned workflows (each executing its own partition). In this method, potential bottlenecks that we could see in the previous methods, will be removed. Decentralized orchestration was found to minimize the amount of traffic in the workflow but this approach increases the complexity of the workflow design and execution, while deadlock can be built [20]. Furthermore this model needs parallel programming that can cause to additional challenges.

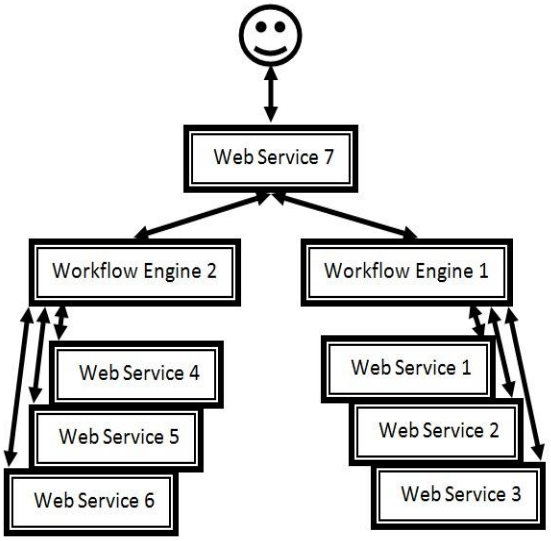


Fig. 3. Decentralized Orchestration Model

5 An Efficient Alternative Model For Web Services Composition With DeputR Servers

In our model, the data flow is decentralized [16] while the control flow remains centralized. Applications with this method have advantages compared to other centralized/distributed control and data flow. To accomplish this, Deputy Servers are used in the system. The contribution of this model is the introduction of the Deputy Server. This server is participated with the workflow engine, but strives to be a non-disruptive extension to the traditional orchestration model.The workflow engine negotiates with the Deputy Servers using the control flow. The servers carry out Web service invocations instead of the workflow engine.

Deputy Servers will need to communicate with three parts: the central workflow engine, other Deputy Servers and the Web services. Interaction of Deputy Servers with each other exposes the desired interface for them.

As mentioned earlier in our model, data flow is decentralized. Instead of sending all data back to the workflow engine, data can stay on the Deputy Servers. Because of the workflow continues, the data on a Deputy Server, should be sent to other Deputy Servers. Deputy Servers provide Web service requests and receive Web service responses. Requests are either received from the workflow engine or another Deputy Server or they may be the response from a previous request. Responses will either be stored, or forwarded to another Deputy Servers or the workflow engine. So a full SOAP protocol stack [6] is not a necessity at the Deputy Server.

Wheras in the traditional systems, the workflow engine communicates with the Web services, this is not required in our model. Instead, the workflow engine does the same work by sending workflow Web service interaction to the Deputy Servers (they would act as pass-through services). The workflow engine, should coordinate the execution of the workflow by using control messages. These messages may be sent to the Deputy Servers that require them or perhaps forwarded by other Deputy Servers.

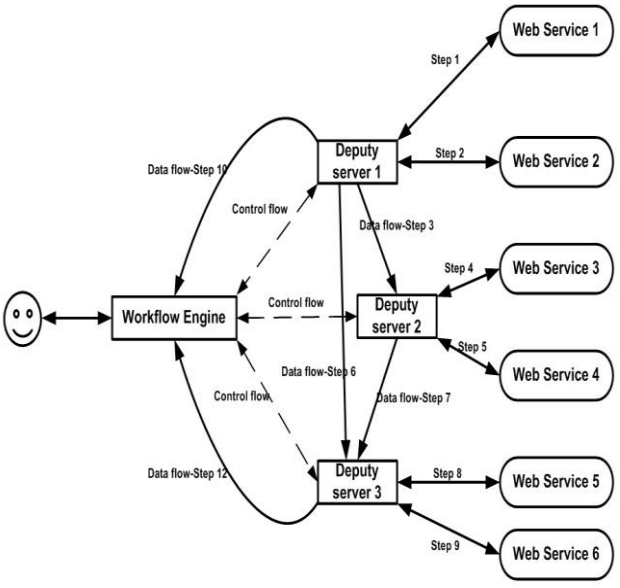


Fig. 4. A New Model with Deputy Servers

5.1 Concept of Deputy Server

Deputy Servers need to interact with three actors: the workflow engine, other Deputy Servers and the workflow web services. The expectation is that the minimising the cost of data handling and communicating control flow messages is not as beneficial as minimising the cost of communicating the large data flows associated with the workflows. In Deputy Server with another Deputy Server (D-D) interactions of our model, data flow is decentralized. Instead of sending all data back to the workflow engine, data may remain on the Deputy Server. In order for the workflow to continue, the data on a Deputy Server, would have to be sent to another Deputy Server. This requirement shows that a mechanism exists for exchanging data messages between Deputy Serv-

ers.

The second interaction is related to Deputy Server and workflow web servise. Deputy Servers make web service requests and receive their responses. Although, Deputy Servers neither create web service requests (they may construct them from available data) nor do they use the responses. Requests are either received from another Deputy Server or the workflow engine, or alternatively, they can be the response from a previous request. Responses on the other hand will either be stored, or forwarded to another Deputy Server or the workflow engine. Therefore a full SOAP protocol stack is not a obligate at the Deputy Server.

The last case is the interaction between Deputy Server and workflow engine. The workflow engine remains the centralized orchestrator for the workflow. Whereas traditionally in systems like BPEL [12], the workflow engine interacts with the web services, this is not required in our model. Instead, the workflow engine may accomplish the same tasks by delegating workflow web service interaction to the Deputy Servers, which would act as pass-through services. The workflow engine must be able to coordinate the execution of the workflow with control messages (now directed at Deputy Servers). These control messages could be sent directly to the Deputy Server that requires them, or may be forwarded by other Deputy Servers. For the purposes of this paper, a decentralized control flow is not considered, and thus the workflow engine would communicate directly with Deputy Servers.

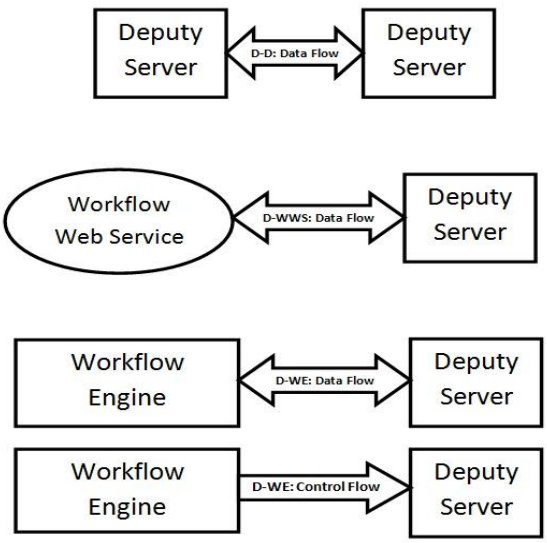


Fig. 5. Deputy Server Ineractions

Deputy Servers should accomplish two basic functions: invoking web services operations and allowing the workflow engine to dictate the data flow. Being able to invoke web services has a simple solution, by making the Deputy Server as a web service client. Of course, the workflow engine would dictate when and how a Deputy Server would invoke a web service. Also the workflow engine would be responsible for controlling the data flow between itself and the Deputy Servers and for controlling the data flow between Deputy Servers.

5.2 Storage in Our Model

The Deputy Server needs to save responses from web services. As scientific workflows may deal with large scales of data, and the Deputy Server may be handling multiple requests concurrently, keeping responses in memory may not be possible. Whether or not to store responses to permanent storage cannot be specified by the workflow engine, because the Deputy Server may be handling requests from multiple workflow engines. This role will instead be the responsibility of the Deputy Server and will be based on the current state and available resources of the machine on which it executes. This presents a level of non-determinism into the performance evaluation of our model since the diffrent workload of the machine effects the performance of the Deputy Server. A force-write policy could be enforced so that all data is always store to permanent storage, although this too may hamper performance.

5.3 State in Our Model

Whether or not Deputy Servers require to maintain state largely depends on whether a synchronous or asynchronous communication method is done for Deputy Server-workflow engine interactions. With asynchronous communication, the workflow engine instructs the Deputy Server on what tasks to execute and then terminate its link. The Deputy Server would accomplish the required action and once completed, initiate a new connection with the workflow engine. To execute this, it would have to maintain state information for pending requests, by mapping tasks identifiers to network connections.

If synchronous communication is used, it is possible to relax the state maintenance requirement. In synchronous communication, the workflow engine keeps its connection to the Deputy Server open for per request and waiting for its response.

Regardless of the communication method used, some state will always need to be maintained at the Deputy Server. The reason is because it will need to store web service response which may be used for any number of various web service requests. In other words, there is no guarantee that a web service response will be consumed immediately. Also the Deputy Server is in a position to help the workflow engine for optimising the workflow through using of statistics. Although analysis of the statistics is not the responsibility of the Deputy Server, but the Deputy Servers should record metrics that can be analysed. For example, for each request-response pair, the size of the request and response should be recorded, also the delay between making the request and when the first bytes of the response are received. This is especially true, where the performance of our model needs to be analysed.

5.4 Data Handling In Our Model

As web services can be developed by multiple organisations, the web services involved in a workflow may not share a common interface. This could extend both to the message and type formats used. There are some scenarios where data

transformations might be needed: Typed values [22]: One web service may indicate a boolean value with an integer type, another with a text value, and yet another could use boolean value. Headers: One web service may require headers for all its messages, whereas another might not. General transformations: The output of one web service operation cannot be supplied as input to another web service operation unaltered. It might be necessary to drop elements, add new ones or somehow modify the response.

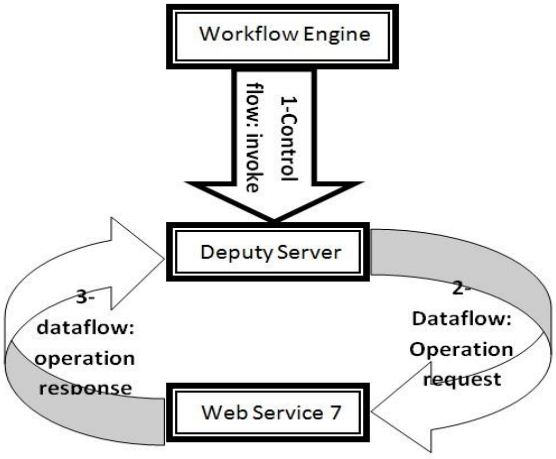


Fig. 6. Invoking a Web Service

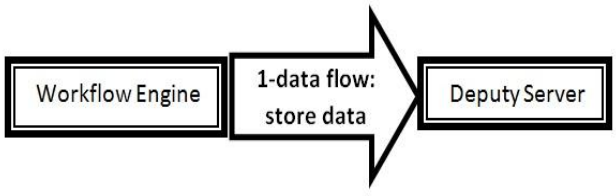


Fig. 7. Storing Data on a Deputy Server from a Workflow Engine

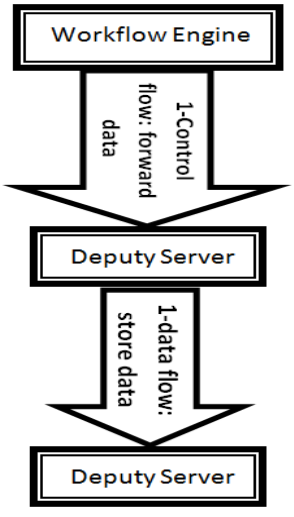
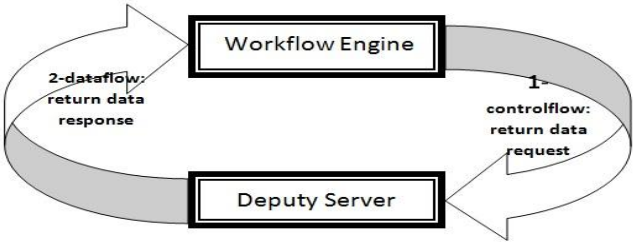


Fig. 8. Storing Data on a Deputy Server from a Deputy Server

5.5 Concept of Workflow Web Services

Web services used in existing workflows may not be available inside the experiment test bed for this project. Therefore, web services will have to be otherwise provided. One such way is by introducing a simple web service into the system that can be used to construct new workflows.

Computation costs of the WWS are not an issue for this project, since its purpose is to examine the effect of performance due to communication costs. As existing web services (or data sources) will not be available in the experiment test bed, the web services used should be able to



generate data to be included in their responses. To be able to be used as input for other web services, the specification of the input and output data should be compatible (considering data transformations will not be used). To allow for variability in the experiments conducted, the amount of data returned by WWS should be configurable.

Fig. 9. Retrieving Data Located on a Deputy Server

5.6 Concept of Workflow Engine

In the traditional models, the workflow engine may maintain state (e.g., for each current workflow execution instance, what are the requests that are pending) of the different WWS in order to determine what to do next in the workflow. The introduction of Deputy Servers however, necessitates that state be maintained for them as well. State information in this case could be what data exist on which Deputy Servers, with what identifiers are they tagged, etc.

The implemented workflow engine provides the functionality of a traditional workflow engine, as well as the ability to interact with Deputy Servers of our workflow architecture as all interactions with Deputy Servers and workflow web services are of the form of web service operations, the workflow engine includes web service clients to interact with both of them. In addition to controlling the workflow, the workflow engine implementation serves the purpose of logging the results of the workflows. The workflow engine uses the log results to calculate certain statistical metrics at runtime. These logs and metrics form the basis of the experimental results and their analysis. The workflow engine is also responsible for loading the experiment configurations.

4 Conclusion

As applications continue to move away from the desktop and onto the network, the importance of well-performing web applications increases. Whereas many of the migratory desktop applications make their transition as lightweight web applications with few data demands, not all applications are created equal.

It can be claimed that the valuable steps are the review of the posed challenges in area of combining the services and optimizing them. Applying the composite Web ser-

vices has many commercial benefits for organizations. Firstly, it reduces the time of producing process that cause to reduce the time for products to reach to the market. Secondly, producing the applications with available services decreases the business risks of software producing and avoids dealing with new software errors. Thirdly, by using available Web services, organizations' need for several new skills will decrease which also cause to save in cost of software production. Finally, by using available Web services it is possible to choose the best service from several options which leads to better softwares in terms of quality. Therefore we explained the advantages of combining Web services and existing methods and finally discussed a model for Web services composition in order to optimize former compound methods, based on the use of deputy servers.

In the first methods, cost and data exchange, time reduction, response time improvement, and centralized control of communication between different services to provide appropriate services to the customer is aimed. In the improved models main goal is, the cost and time reduction associated with large current data (especially repetitive data). One of our model considerable advantages, is in a large volume data transaction where simultaneous respond to many requests is needed, the result of traditional storage in this volume (Work Flow Engine) may be impractical while in this case the optimized approach will partition data into different deputy servers leading recording volume decrement and makes it practical. Another important advantage is that, in Web services without a deputy server, data exchange should only operate with the Work Flow Engine that might be located in far distances. As a solution near domain deputy servers can be identified to transfer data through them in a lower time with less cost. In the optimized model entire data base has been distributed on multiple servers so the processing power is increased and lateral services can be provided without any traffic enhancement or work overflows.

In other word this paper attempts to evaluate the performance of a proposed alternative to existing workflow execution models. Our effective orchestration model attempts to eliminate the bottlenecks in today's centrally-coordinated methods. It does so my relieving the load of the central bottleneck by keeping data closer to where it is used. It means the placement of the Deputy Servers can improve the performance of our model as compared to the traditional model. For example, in traditional model, all data have to be sent to the workflow engine which can be at a remote situation (compared to the Web services). In our model a Deputy Server can be placed near the Web services (for example in the same domain), so they cause to a lower cost for data transfer.

REFERENCES

[1] N. Milanovic, and M. Malek, "Cuurent Solutions for Web Service Composition," *Proc. IEEE Symp. Interner Computing*, November-December 2004.

[2] T. Berners-Lee and J. Hendler, and O.Lassila "The Semantic Web," *Scientific American,* 501(5):pp. 28-37, May. 2001.

[3] A.Masri, and E.Mahmoud "Investigating Web Services on the World Wide Web," *Proc. 17th International Conf. World Wide Web (www)-for QWS-WSDLs Dataset Version 1.0*, pp. 795-804, April 2008.

[4] R. Pessoa, "A survey of Service Composition Approaches," *Proc. Conf. Enterprise Distributed Object Computing Conference Workshops*, pp. 238-251, 2008.

[5] Z. Mahmood, "Enterprise Application Integeration based on Service Oriented Architecture," *International Journal of Computers,* vol. 1, no. 3, pp. 135-139, 2007.

[6] E. Cerami, *Web Services Essentials-Distributed Applications with XML-RPC, SOAP, UDDI, and WSDL.* Publisher: O'Reilly, pp. 1-24, ISBN:0-596-00224-6, First Edition February 2002.

[7] H. Hass, and A. Brown, "Web Service Glossary," Technical Report, WWW Working Group Note, Feb. 2004.

[8] D. Booth, and C. Liu, "Web Services Description Language (WSDL) Version 2.0," W3C Technical Reports and Publications, Primer W3C Working Draft, August 2005.

[9] A. Oussalah, and M. Lina, "Toward the Definition of the Loose Coupling Notion in a Composite Service," *Proc. IEEE Symp. Computer Engineering and Applications (ICCEA)*, pp. 339-343, March 2010.

[10] A. Alamri, M. Eid, and A. Elsaddik, "Classification of the state-of-the-art Dynamic Web Services Composition Techniques," *J. Classification,*University of Ottawa,Canada,Int.J Web and Grid Services, vol. 2, no. 2, , Apr. 2006.

[11] J. Ge YuhUi Qiu Shiqun Yin, "Web Services Composition Method Based on OWL," *Proc. International Conf. Computer Science and Software Engineering*, pp. 74-77, 2008.

[12] B. Mathew, and P. Sarang, *Business Process Execution Language for Web Services.* An Architect and Developers Guide to Orchestration Web Services Using BPEL4WS, pp. 92-104, 2006.

[13] B. Benatallah, Q.Z Sheng and M. Dumas "The Self-Serve Environment for Web Services Composition," *Proc. IEEE Symp. Internet Computing (SCC '06)*, January 2003.

[14] S. Dustdar, and W. Schreiner "A Survey on Web Services Composition," *Technical University of Vienna Systems-Information Systems Institute,* http://www.infosys.tuwien.ac.at. 2004.

[15] L.J Zhang, "SOA and Web Services," *Proc. IEEE Symp. Service Computing (SCC '06)*, Sept 2006.

[16] B. Benatallah and Q. Sheng, "The Self-Serv Project Uses a P2P-based Orchestration Model to Support the Composition of Multienterprise Web Services," *IEEE Computer Society Trans. Internet Computing*, vol. 3, pp. 40-48, January 2003.

[17] N. Kyprianou, "Web Service Orchestration," Master of Science dissertation, Dept. of Informatics., Edinburgh Univ., 2008.

[18] Y. Rao, B. Qin Feng, J. Cang Han, and Z. Chao Li "A Security Integrated Model for Web Services," *Proc. IEEE Symp. Machine Learning and Cybernetics (ICLMC '05)*, pp. 2953-2958, Aug 2004, doi:10.1109/ICLMC.2004.1378538.

[19] G. Chafle, S. Chandra, V. Mann, and M. GowriNanda "Decetralized Orchestration of Composite Web Services," *Proc. 13th. International World Wide Web Conf. Alternate Track Papers And Posters*, pp. 8-16, 2004.

[20] N. Kavantzas, D. Burdett, G. Ritzinger, T. Fletcher, and Y. Lafon "Web Services Choreography Description Language Version 1.0," W3C Technical Report, W3C Working Draft, Dec. 2004.

[21] M. Weski, *Business Process Management: Concepts, Languages, Architectures.* Springer, pp. 123-135, Sep. 2007.

[22] Nikos. Kyprianou" Hybrid Web Services orchestration," Maaster of Science thesis, school of information,2008.