2009 International Conference on Computer Engineering and Technology

Dynamic Web Service Composition

Faisal Mustafa
School of Computing and Engineering,
The University of Huddersfield,
Queensgate, Huddersfield HD1 3DH, UK
f.mustafa@hud.ac.uk

T. L. McCluskey
School of Computing and Engineering,
The University of Huddersfield,
Queensgate, Huddersfield HD1 3DH, UK
lee@hud.ac.uk

***Abstract*—A major area of recent web-related research concerns automated web service composition. A major advantage of web services technology lies in the potential of creating value-added services by combining existing ones to achieve customized tasks. How to combine these services efficiently into an arrangement that is both functionally sound and architecturally realizable is a very challenging topic that has founded a significant research area within computer science. Our research contribution lies in the area of dynamic composition services selection. We have started by collating and analyzing current outstanding problems within the dynamic composition area. To help explain the background to these challenges we compare, firstly, the key components of distributed computing technologies with web service composition. Then we define the 'execute ability' problem - the key idea that preconditions of web services must be satisfied before or as a result of composition. We discuss data distribution strategies among services, how they can be used to overcome problems in dynamic composition problems and how they relate to the quality of service. Finally, we present our proposed framework model to handle the process of web service composition and execution. We propose that this framework eliminates the main problems as discussed in the paper.**

Keywords- Dynamic Web Service, Composition, Quality of Service

I. INTRODUCTION

People use the internet daily to look up financial market quotations, to buy products, etc. Most of the information on the web is designed only for human use [1,6,23,24]. Humans can read HTML documents and understand them, but their inherent meaning is not shown to allow their interpretation by computers. In other words the essential text-based web does not support software interactions. How we give meaning to the text based web is precisely the objective of the Semantic Web – to make possible the effective processing of Web information by computers [24]. The Semantic Web is a future vision of an extension of the www, in which information is given machine-readable semantics, enabling computers and people to work in cooperation [4,7,19]. Internet based applications need to be capable of performing search and automatically interact with other internet-based application. The goal is to enable software systems to automatically perform operations that previously required human intervention, such as searching for and buying goods and services while optimizing user criteria such as a resource (price, time etc). All these examples come from relatively different areas but still share some fundamental characteristics [1,6,22].A service is offered by a service provider, an organization that procures the service implementation supplies its service description (WS Register), and provides technical and business support to clients [7,11,16].

Previous research [14,17,21,23,29] has divided the process of composition of services into static and dynamic. Static composition is purely manual i.e. firstly, the user problem must be defined and then a manual selection of services according to desired outputs is performed. In dynamic composition, automated tools are used to analyze a user problem, select and assemble web service interfaces so that their composition will solve the user problem [4]. In other words, from a user perspective, this composition will continue to be considered as a simple service, even though it is composed of several web services [30].

In our research project, we are aiming to create methods for the automatic, dynamic composition of web services. In particular the methods will be able to cope with problems associated with the distributed, independent, and uncertain nature of the web. Individual service availability, reliability and quality are factors which make composition difficult. This paper discusses the main issues faced by web services composition researchers and proposes some solutions.

The paper is structured as follows: the first part discusses web services, distributed computing technologies and the execute ability problem as discussed in [15]. This is to do with determining whether the preconditions of the actions which make up a composite service can be satisfied in the "open world" environment of the web, where information is incomplete. We examine the execute ability issue because it is a common problem in both approaches (web services and distributed application development methods). We postulate a data distribution strategy in the composition process model, and web services selection criteria on the basis of QoS (Quality of services), to deal with issues like throughput Capacity, Latency, Response Time, Availability, Reliability, Reputation and Execution cost[18,19]. Although different approaches are available to address this problem, we are trying to start with current problems from different approaches in this area. Finally in sections V and VI we present a frame work model which is aimed at eliminating currently faced problems during composition.

978-0-7695-3521-0/09 $25.00 © 2009 IEEE
DOI 10.1109/ICCET.2009.191

II. A MOTIVATING EXAMPLE

Web services are self contained programs that can be executed through the standard, global protocols of the internet. There are many services around the web and each one has a limited functionality. In many cases, a single web service is not sufficient to respond to the user's request and often services should be combined through a pattern of composition to achieve a specific goal [3,5,7,9,22,29]. Such a composition is carried out manually at present, which means that a user needs to execute all these services one by one and these tasks can be time and effort consuming. Due to constant changes in output/input parameters values, interfaces, networking issues, it is difficult to integrate and maintain these services.

As an example application, we consider the work by Hu, who has used the web service paradigm in an effort to make the communication between police departments and UK government agencies more effective [3, 23,29]. Each of the providers (the agencies) offers data and enquiry capabilities to the police forces (a range of services are offered as shown in Figure 1). In a typical police enquiry, whenever a police officer wants to investigate about any person he has to search through for a criminal record, fingerprints, vehicle registered in the person's name, insurance details and vehicle movement in a particular interval of time across the country. In fact, more than 50 department's services required integration in this project [29].

To date this work has achieved limited success towards the application of automated, dynamic composition of web services, because in the Police domain, only authorized persons have access to some of these services [4, 5]. So just in time integration of services is not possible at the moment, and at best a semi-automatic static composition method has to be used. For real time results we need dynamic composition. Such a type of dynamic composition is difficult because of the following factors.

- Firstly, it is very difficult to analyze services (even manually) from the web services repository (the UDDI) and integrate them to get specific required outputs.
- Secondly, web services contents are going to be changed routinely to fulfill the user requirements. At selection time before composition the system must be able to select up-to-date services. There has been considerable research aimed towards getting updated web services at the time of composition, but techniques still fall short of the requirements of dynamic composition.
- Thirdly, web service suppliers are using different conceptual models to describe their services. To enable an automated dynamic composition process we need one structure (model) of available services so that a service can easily invoke other services without any technical overhead. Figure 2 provides us with a summary overview of the development stages of web services.

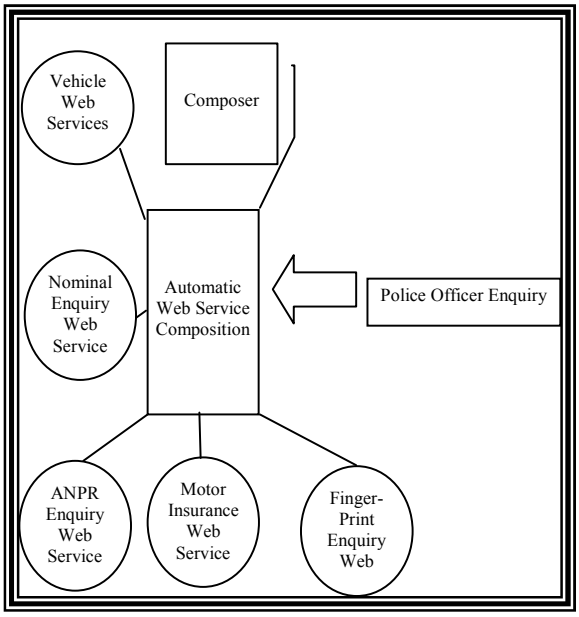

Figure 1

On the next stage of the composition process selection of services is a very important issue. There are three types of rules to consider, whether involved in static or dynamic composition, as illustrated in Figure 2. In Template Based, a specific template either needs to be created, or acquired from a repository. A user has to locate the respective template first (in static composition process) before compose services. This is time consuming process as well. In Interface Base, on the base of inputs and outputs through interfaces, user is getting services reference and these composite services after composition process provide final results. It is highly adaptable method but functionality is not guaranteed. During composition process, some time we are getting similar interfaces, but after composition undesirable outputs (final results). Mostly automated composition available tools using interface base selection concept. In Functionality Base Composition, along with pre-conditions and post-conditions, user has to provide first-order logic (formula representing the logic) into the interface information. The above mentioned individual rules are adaptable if we are interested in manual selection. In the context of our interface and functionality based (two rules combination) approach, the problem of dynamic selection can be solved as discussed in section VI.

III. WEB SERVICES AND DISTRIBUTED COMPUTING

It has been claimed that web services are reinventing the wheel because they share many characteristics with other distributed computing architecture, such as CORBA, Distributed Smalltalk, RMI or DCOM [14, 24]. Technologies from distributed computing normally have a tightly coupled relationship between client and server where the coupling between various components in a system is high.

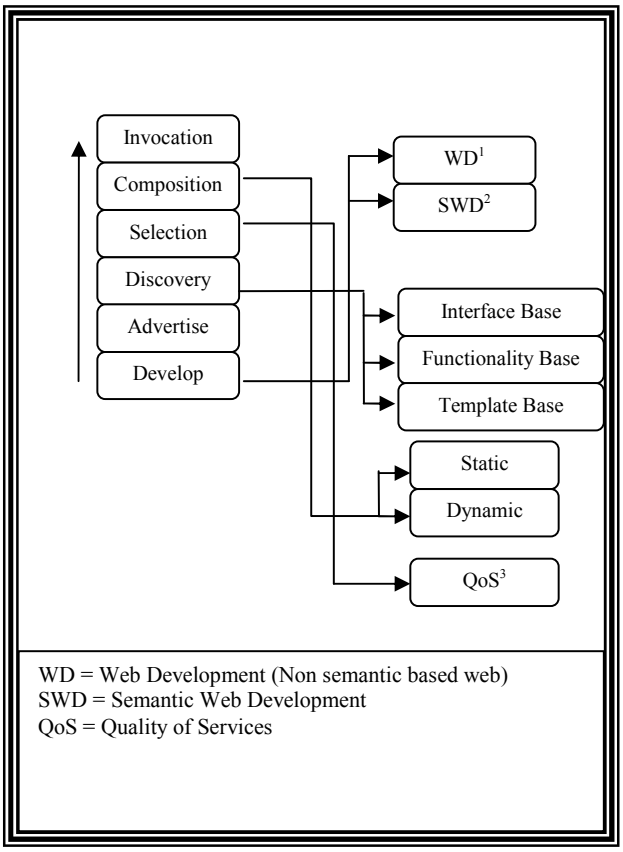

Figure 2

In RPC–oriented interaction, the service request takes the form of a method call defined by a name and a set of input and output parameters. During execution it waits for a response in a real time. In the web service-oriented interaction style, the particular web service request takes the form of a complete XML (query) document and will provide acknowledgement in the form of email or any other type of real time response. In these cases we need detailed knowledge of available services and involved overheads (physical and logical structure) to combine them.

The main question arising at this point is how can we reduce tight coupling and static binding between these components? Otherwise web service composition will give us the same capability like any distributed computing applications. The main potential advantage, therefore, of web services over RPC's is the potential for service (application) to automatically discover and compose services on demand. One technology for achieving this is AI planning, which can be used to automate the composition of semantic web services and dynamic discovery [7,10,12,13]. These techniques can potentially enable client and web services to find each other with out prior knowledge of each other. To apply these techniques for accurate results we have to introduce semantic and ontological concepts to our web service model. Hence the act of looking up capabilities of a Web services can be done at the same time as dynamically composing it with others, rather than the use of inflexible static binding between one or more services. These two previous concepts make web services stand out against distributed applications. So it will be impractical to develop real time applications (like Hu's police example explained above) by using distributed technology environments.

IV. THE EXECUTE ABILITY PROBLEM

Web services are sometimes portrayed as "silver-bullet" solutions to integrated web applications, because they have the potential to replace the role of the original web and relational database-related technologies [24]. Web service technology enables application-to-application interactions over the web, since any interaction with a web service involves sending and receiving messages [15,28]. One way to describe a particular service is in terms of its preconditions on input parameters values, precondition on prior operations invoked, and its output conditions and effects. In the web services composition process, the two terms choreography and orchestration are very important.

Web services choreography is to do with the interactions of services with clients/users, and determines the specification of operations, states and conditions, which control how the interaction occurs [27]. Following the temporal constraints output by the choreography process should result in the completion of a useful function [23]. As stated by Micheal Hu, Web service choreography permits the description of how web services can be composed, how rules and association in the web services can be established, and how the state, if any, of composed services is to be managed [29]. The World Wide Web Consortium introduced the Web Services Choreography Description Language (WS-CDL) which captures the interaction within the participating services. The choreography model also helps to determine control-flow dependencies, message correlation, time constraints and transactional dependencies.

On the other hand an orchestration defines the sequence and condition in which one web service invokes other services in order to carry out any specific task, i.e an orchestration is the pattern of interaction that a web service planner must follow in order to achieve a goal [23,29]. On the basis of the above discussion we can say the dynamic composition model requires four additional layers Semantic, Ontological, Choreography description and Orchestration concepts. The Choreography Model and Orchestration Model provide us with a comprehensive solution for basic issues like precondition, effect and post condition.

V. DATA DISTRIBUTION AND QUALITY OF SERVICES

Service-Oriented Architecture (SOA) is recently defined paradigm for organizational models of systems, aimed at simplifying large business operations using existing services. SOA's main manifestation is in the area of web services. Although there is plenty of controversy about how SOA will manifest itself in the context of web services technologies, the issue of the quality of services will always be central to the argument [2, 24]. Businesses will have to have secure web services and will have to be able to guarantee that

messages arrive at their intended destination and are processed reliably [4, 7, 23]. During execution of a composed web service process we also require output variable values from different services (data servers). If some parameters are missing or due to any reason not available for the next service then the process will fail. Web service standards and technologies are composed of two major types of application interaction patterns on the basis of their database interaction access: Centralized Dataflow and Decentralized Dataflow. If our focus is towards dynamic service composition then in both approaches there are some limitations [3,8,9,11,17,19]. In Centralized Dataflow, data between component services is passed through the composite services and in that situation bottleneck problems occur, affecting throughput and response time. On the other end in Decentralized Dataflow components, services exchange data directly with various data base servers. The result is that the distribution of network traffic among all the services involved improves loading characteristics on the composite service, improving in particular throughput and response time. Both of the models have their own advantages in distributed computing environment [7,13,21,23,25]. The Decentralized Dataflow seems to be very efficient for dynamic services composition but in some situations (for example in the Hu's Police case study described above) it will affect QoS factors like latency, execution cost and capacity. For automatic web services composition we can use the centralized data model by adding a middleware extension support to avoid tight coupling between services. This type of extension will be automatically added in our model if we use UDDI (Universal Description Discovery and Integration) or WS-Coordination and WS-Transaction. The proposed frame work model will result in performance improvement, lower time response and higher throughput maintainability. The web services QoS requirement refers to both functional as well as non-functional quality aspects of web services [2,16]. The overall performance of web services depends on the complexity of the application, as well as the network, messaging and transport protocols (SOAP.HTTP).

VI. CURRENT TECHNICAL AND FUTURE WORK

In this short paper our discussion is around distributed technologies, execute ability issues, data distribution, QOS issues and how to avoid problems with execute ability issues. So this basic path provides us with a way leading to a real time integrated setup for execution of these composed services, minimizing dependencies and faults as mentioned in [11, 14, 19, 23]. In the proposed model as shown in Figure 3 we try to add a solution to the main problems which we discussed above. The activity of this process starts when a new service being firstly registered in a service repository. We use a translator if any type of language conversion is required. The service composition request arrives at a web server, which will try to locate in its own services database. If already such a type of interface base composition exists then an integrated result will send to client. Otherwise the server will try to search through a web services database. The web server will find desired services through matching engine from the web service database. An evaluator will evaluate these services on the basis of interface base search in the first round. On the basis of results during the first round, the evaluator again will apply functionality-based rules in the second round. The composer will compose these selected services and return the services address to the web server. The composed solution results will be sent to the composition requestor. The copy of this services integration will also be saved in a service repository for future use.

For the practical implementation of such a desired framework we are using Eclipse version 3.4.0 with GEF (Graphical Editing Framework) plug-in, Apache tomcat server 6.0, and a UDDI server based integrated platform. The above mentioned path actually leads us towards a semi-automatic type of composition. If any of servers is not responding or input/output issues arise then the composition process will fail. Secondly, during experimentation we also observe that the composition process is not getting any advantage from the new uploaded services. To include new services into our composition process we have to re instantiates both servers. By this integrated environment we are trying to fill the gap between distributed network application development and web services as discussed in section III. In the SOA paradigm, if we introduce communication links between distributed technologies and web services then automatically we will find an autonomous environment. Where applications can connect without prior knowledge of platform, human interaction and technical details, then we can achieve loosely-coupled, platform-agnostic application properties.

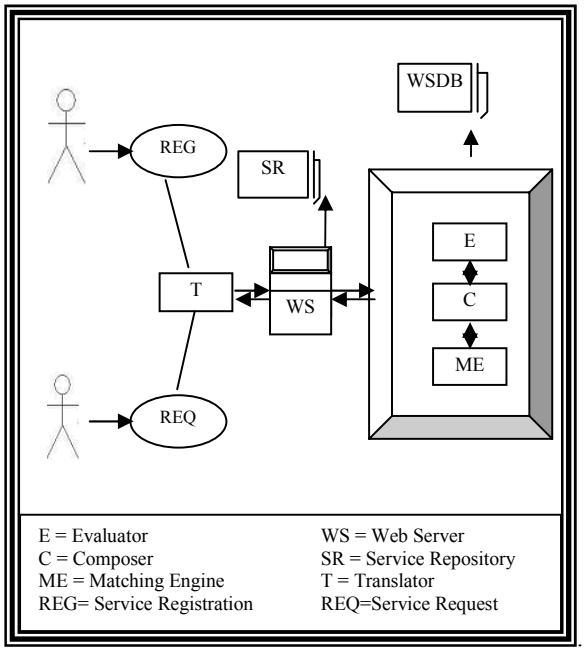

Figure 3

To overcome this issue we have to introduce a plan which itself produces required outputs on the basis of input values as discussed in [7, 13]. So our next step is to view

dynamic web composition process as an automated planning problem. We can consider composition tasks process in different states i.e. initial state, final state and goal (composition final result) of a planner.

VII. CONCLUSIONS

At this stage, automated dynamic web service composition development process is still under development, although some automated tools and proposals are available. The full automation of this dynamic process is still an ongoing research activity. In this paper we have outlined the main challenges faced by web service composition, such as execute ability, data distribution and its effect on QoS. We also tried to elaborate the main differences and advantages of web services over distributed application development. Based on an analysis of current problems, we have introduced a model of dynamic services. In the proposed model we try to fix current issues for dynamic composition. In our future work we intend to elaborate the phases of the proposed model, and develop searching algorithms, leading to a robust solution to the dynamic composition task.

REFERENCES

[1] Jinghai Rao and Xiaomeng Su. "A survey of automated web services composition Methods" 2004.

[2] Natallia Kokash. "A Service Selection Model to improve Composition Reliability" 2006.

[3] Volha Bryl and Fabio Massacci. "Designing Security Requirements Models through planning" pages 33-47,2006.

[4] Evern Sirin, James Hendler and Bijan Parsia. "Semi-automatic Composition of Web Services using Semantic Descriptions" 2002.

[5] Biplav Srivastava and Jana Koehler. "Web Service Composition – Current Solutions and Open problems" pages 28-35, 2003.

[6] Dan Wu, Evren Sirin james Hendler and Dana Nau. "Automatic Web Service Composition Using SHOP2" 2003.

[7] Mark Carman, Luciano Serafini and Paolo Traverso. "Web service Composition as Planning"2004.

[8] Daniela Berardi , Giuseppe De Giacomo and Massimo Mecella. "Automatic Web Service Composition: Service- tailored vs. Client-tailored Approaches" 2006.

[9] Rosanna Bova, Salima Hassas and Salima Benbernou. "An Immune System-Inspired Approach for Composite Web Service Reuse,2006.

[10] Konard Pfadenhauer, Schahram Dustdar and Burkhard Kittl. "Challenges and Solutions for Model Driven Web Service Composition"2005.

[11] Cheng Yushi, Lee Eng Wah and Dilip Kumar Limbu. " Web Services Composition- An Overview of Standards" Pages 137-149, 2005.

[12] Axel Polleres presented topic " AI Planning for Semantic Web Service Compositon.

[13] Freddy Lecue and Alain Leger. "Causal link matrix and AI planning: A model for Web Service Composition".

[14] Biplav Srivastava, Joseph P. Bigus and Donald A. Schlosnagle. "Using ABLE to bring Planning to Business Application" 2004.

[15] Yilan Gu and Mikhail Soutchanski. " A Logic For Decidable Reasoning About Services" 2006.

[16] Federico Chesani, Paola Mello and Marco Montali. "Abduction for Specifying and verifying Web Service and Choreographies".

[17] Annapaola Marconi, Marco Pistore and Paolo Traverso. "Implicit vs. Explicit Data-Flow Requirements in Web Services Composition Goals".

[18] John Gekas and Maria Fasli. "Automatic Web service Composition using web connectivity analysis techniques" 2005.

[19] Shuping Ran. "A Model for web services discovery with QoS".

[20] Mikio Aoyama, Sanjiva Weerawarana, Hiroshi, Clemens Szyperski, Kevin Sullivan, Doug Lea. "Web Services Engineering: Promises and Challenges".

[21] Lucian-Mircea Patcas. "Data Distribution in Web services Composition". Presented in ("The 15th PhDOOS workshop ECOOP, Glasgow).

[22] Marco Pistore, Jose Luis Ambite, Biplav Srivastava. "Workshop on Knowledge level Automated Software Engineering" (August 28, 2006).

[23] Michael Hu, Howard Foster "Using a Rigorous Approach for Engineering Web Services Compositions: A Case Study".

[24] Eric Newcomer "Understanding Web Services, XML, WSDL, SOAP and UDDI".

[25] H. M. Deitel, P. J. Deitel "Web Services A Technical Introduction" 2002.

[26] Rajesh Sumra, Arulazi D "Quality of service for web services-demystication, limitations and best practices".

[27] Chris Peltz (Hewlett-Packard Company) "Web Services Orchestration and Choreography".

[28] Francisco Curbera, William A. Nagy, Sanjiva Weerawarana "Web services: why and how".

[29] Michael Hu "Web Services Composition, partition, and quality of service in distributed system integration and re-engineering" 2004.

[30] Daniela Barreirs clars "selecting web services for optimal composition" 2006.

[31] Karl Gothchalk (IBM services architecture team) "web services architecture overview".