WEB SERVICES COMPOSITION: A PRAGMATIC VIEW OF THE PRESENT AND THE FUTURE

Dhavalkumar Thakker, Taha Osman, David Al-Dabass
School of Computing & Informatics,
Nottingham Trent University,
Burton Street, Nottingham NG1 4BU, United Kingdom.
Email: dhavalkumar.thakker@students.ntu.ac.uk, taha.osman@ntu.ac.uk,david.al-dabass@ntu.ac.uk

KEY WORDS
Web Services, Business Process Management, BPEL, CDL.

ABSTRACT

Web Service is loosely coupled, highly accessible distributed computing technology that can expose applications beyond the firewall. Composition of Web Services has received much attention from the business and the research community. Composition techniques are classified as static, dynamic and semi-automatic composition, each addressing different application areas and requirements. Here, we evaluate such approaches from two perspectives: as Business Process Management (BPM) solution and the facilitation they provide to the composition participants: end users, developers and composers in their present form. In this paper, we use the present workflow-based composition standard WS-BPEL, explore its merits and limitations and analyze another static composition standard WS-CDL which has complimentary role to BPEL for the business process management. We also discuss dynamic composition as the future work and what it can offer.

1. INTRODUCTION

The last decade has witnessed an explosion of application services delivered electronically, ranging from e-commerce to information service delivered through the World Wide Web (WWW) to the services that facilitate trading between business partners, better known as Business-to-Business (B2B) relationships. Traditionally these services are facilitated by distributed technologies such as RPC, CORBA and more recently RMI. Web Services is the latest distributed computing technology. It is a form of remote procedure call like other distributed computing technology, but uses XML extensively for the messaging, discovery and description. The use of XML messaging makes Web Services platform and language neutral. Web Services use SOAP (Simple Object Access Protocol)[1] for XML messaging, which in turn uses ubiquitous HTTP for the transport mechanism. HTTP is considered as a secure protocol thus it allows the Web Services to be exposed beyond the firewall. The Web Service messages and operations with invocation details are described using a platform-independent language WSDL (Web Services Description Language)[2]. Web Services can be published and discovered using UDDI (Universal Description Discovery and Integration)[3] protocol. The Web Services architecture centred on WSDL, UDDI and SOAP is an instance of Service Oriented Architecture (SOA). Using this architecture services can be published using UDDI, with WSDL based description, and can be searched, called and bind at run time making it loosely coupled and highly accessible.

To take advantage of these features of Web Services, network applications services have to be developed as Web Services or converted into Web Service using the wrapping mechanism [4]. Moreover, multiple Web Services can be integrated either to provide a new, value-added service to the end-user or to facilitate co-operation between various business partners. This integration of Web Services is called "Web Services composition" and is feasible to achieve because of the Web Services advantages of being platform, language neutral and loosely coupled.

The logic for the composition mainly involves two activities: selection of the candidate Web Services that fulfil the requirement in accumulation and flow management. Flow management is further categorized into control and data flow, where control flow is the order in which Web Services operations are invoked, while the data flow is the order in which the messages are passed between the Web Services operations. The level of automation provided in performing selection of services and flow management classifies composition into static, semi-automatic and dynamic. Static composition involves prior hard coding of the service selection and flow management. Performing selection and flow management on the fly, in machine-readable format leads to dynamic composition. In semi-automatic composition, service composer is involved at some stage.

The focus of the discussion is to contrast these approaches keeping in mind the feasibility of their

Proceedings 19th European Conference on Modelling and Simulation
Yuri Merkuryev, Richard Zobel, Eugène Kerckhoffs © ECMS, 2005
ISBN 1-84233-112-4 (Set) / ISBN 1-84233-113-2 (CD)

implementation today and in the near future. The comparison perspective focuses on the ease of use and facilitation provided to the participating parties, to the end user who is going to use the composed service, to the service developer (service provider) and to the service composer. The later either can be software developer, the business analyst or logic based agent programmes. We will investigate Web Service composition using above mentioned criteria for the both the cases where the existing application is non-web service based or Web Service has no WSDL file or description available.

The structure of the paper is as follows: section 2 provides more detail on the Web Service Composition and discusses composition using WS-BPEL, a prominent industry standard. Section 3 provides insight on the futuristic approaches to the composition. In section 4 we conclude with our analysis of these approaches.

2. OVERVIEW OF THE WEB SERVICES COMPOSITION

Traditional techniques approach Web Services composition as the Business Process Management (BPM) solution. Business process can be considered as the group of activities to carry out business goals [5]. Business applications represent such activities in the business processes, for example a customer order fulfilment process will include individual applications for the activities: customer placing an order, checking account status, verifying order and despatch. BPM deals with achieving the integration of these individual applications to achieve business process view.

Business process can have scope within inter and intra organization relations. EAI (Enterprise Application Integration) is the BPM solution to achieve intra-organization business applications integration, while B2B integration software (e.g. Electronic Data Interchange) addresses the problem for inter organization business application integration. Traditional EAI and B2B integration solutions are very complex, proprietary and presumes many details about the participating applications making them tightly coupled. As the business applications are now being developed using the Web Services, the BPM problems (EAI, B2B) are being addressed with the composition of Web Services, mainly to utilize SOA based Web Service features.

Main industrial standards to achieve such composition of Web Services as BPM solution are WS-BPEL (Web Services Business Process Execution Language) [6], WS-CDL (Web Services Choreography Description Language) [7] and BPML (Business Process Modelling Language) [8]. These approaches use WSDL extensively and build on top of it. WSDL based operations and messages with the data types are the main details based on which the flow management and other essential requirements for composition can be built upon. But to achieve business process view of the composition, composition specification needs to be built on the explicit process model [9]. This process model addresses requirements for describing flow management in composition, handling business transaction with roll back facility, state management for business interaction support, and also handling exception and errors. The process model and the extent to which these features provided, differentiates these standards.

2.1 Composing services using BPEL

WS-BPEL (also called BPEL) specification - enhances and replaces existing standards XLANG from Microsoft and WSFL from IBM. Apart from being based on WSDL, it uses workflow management as process model to achieve the formalization for control and data flow. WS-BPEL facilitates static composition, as the selection of services and decision on flow management is done priori. All the participant services in BPEL are modeled as partners. Partners contribute to the total processing capability of the BPEL process. BPEL process also has its own processing capability for dataflow, control flow, data manipulation, fault and event handling and state management. The significance of BPEL architecture is that the process itself is published as a Web Service. This composed BPEL service can be treated as a single Web Service and can be used for further composition hence facilitating recursive composition.

BPEL standard aims to be the de facto standard for inner-organization (EAI) and inert-organization (B2B) Web Services integration. Hence, BPEL is categorized into executable BPEL (BPELe) and abstract BPEL (BPELa), where BPELe maps well to the requirements of inner-organization based Web Services integration and BPELa is aimed to cover the requirements for achieving integration in cross-organizational domain. BPELe represents the orchestration feature of BPEL, where the Web Service operations are orchestrated from a single party view. BPELa uses same language constructs and semantics as the executable but rather presents the aspects that are more suitable for Business-to-Business interaction.

We describe a Web Service composition scenario implemented using IBM BPEL runtime, BPWS4J [10]. The scenario is based on a travel agent service, which manages the reservation of airline and hotel for the customer trip. Travel agent is implemented as BPEL process, which is the composition of four Web Services: AirFrance service, AirUSA service, HotelRating service and HotelService service. Process logic for the travel

agent is (fig 1): to check the availability of flight service from two competing airlines AirFrance and AirUSA, make flight reservation, and then retrieve hotel ratings from the HotelRating service at the destination city and make the reservation using HotelService Web Service at the selected hotel. The reservations are made based on customer preference (Air Line price preference: cheapest, medium or high and hotel rating preference: A, B or C).

2.1.1 Travel Agent Example

As BPEL is built on top of WSDL, WSDL file of partner business services are required for the composition process. This fact is described in BPEL using partnerLinkType. The portType of such a Web Service defines the role of partner in the composition. Listing 1 shows AirFrance and AirUSA web services as partners and the role they play in the composition using portTypes (i.e. fr: is the unique identifier for the AirFrance WSDL file). Figure 1 is the sequence diagram for the travel agent service. 1.1.a and 1.1.b are two activities for checking the flight between source and destination city is available or not, performed in parallel. The BPEL syntax for this using <flow> to achieve parallel execution is shown in the listing 2. Both invocations are executed in parallel.

Similarly other operations for checking the possibility of reservation are performed on AirFrance and AirUSA, and reservation is made after comparing the price (activities 1.2a, 1.2b, 1.3a, 1.3b in figure 1). The payment details are omitted to keep the example simple. Listing 3 shows the code where the user has specified the cheapest flight reservation in his preference.

```
<plnk:partnerLinkType        name="airFrancePLT">
  <plnk:role name="AFcheckServices">
     <plnk:portType name="fr:AirFrance"/>
 </plnk:role>
</plnk:partnerLinkType>
<plnk:partnerLinkType   name="airUSAPLT">
 <plnk:role name="AUcheckServices">
   <plnk:portType name="usa:AirUSA"/>
  </plnk:role>
</plnk:partnerLinkType>
```

Listing 1. Partners in BPEL process

```
<flow>
<invoke name= “invokeAirFrancecheckServices”
partnerLink = “AFcheckServicesPL”
portType="fr:AirFrance"…….>
<invoke name= “invokeAirUSAcheckServices”
partnerLink = “AUcheckServicesPL”
portType="fr:AirUSA"…….>
</flow>
```

Listing 2. Concurrency using <flow>

```
<switch name="comparePrices">
<case condition="bpws:getVariableData
('compInfo','PriceAirUSA') &lt;
bpws:getVariableData('compInfo','PriceAirFrance')
">
<invoke name= “AUinvokegetReservation” </case>
<otherwise>
<invoke name= “AFinvokegetReservation”
<partnerLink= “AFgetReservationPL” ……..>
</otherwise>
</switch>
```

Listing 3. Selecting the cheapest AirLine<switch>

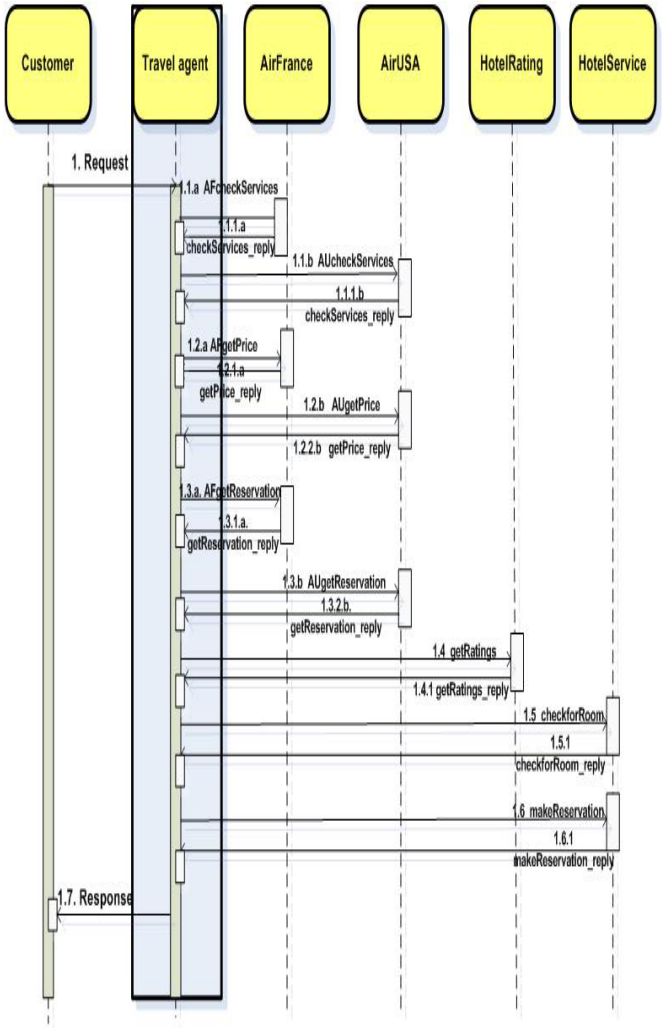

Fig 1. Sequence diagram for the composition [Travel Agent viewpoint]

Implementation of travel agent example shows the expressiveness of BPEL as composition language. In the section 2.2 we will consider the merits and limitations of BPEL architecture when applied to the composition problems from inter or intra organization domain. We will use our travel agent example for the discussion.

2.2 Discussion on BPEL facilitated composition

We will first consider BPELe for the enterprise application integration and Business-to-Business Integration relations. The architecture of the BPELe process model assumes the selection of the services to be done manually, for this reason the interpretation of the requirement expected from the service is according to the understanding of the composer. This assumption is easy to achieve for the inner-organization relations, where the composer has access to all the internal Web Services details, which makes the achieved solution tightly coupled. This is sufficient for the enterprise application integration, as the change in the process logic and addition or removal of services (Business Process Reengineering) can be done in-house, in private domain.

This approach has serious limitations when it comes to B2B integration, where candidate services can be from public, external or cross organizational domain. To illustrate this in our travel agent example, which is a B2B relation, lets consider a scenario, where the new business (AirUK) wants to join this composition. It is really difficult to add new businesses that are not part of the old mix, because it is required that they as a minimum implement the following operations: 1) an operation to check whether flight service is available between two cities 2) an operation to check the trip expense between these two cities.3) an operation for making reservation. Hence, the addition can be made successful if the travel agent business publishes this information as an agreement i.e WSDL file or text description and the AirUK business has or newly implements the service with the above operations and then makes WSDL file available for the composition.

Apart from being tightly coupled, BPELe assumes the B2B integration from the single party viewpoint as the requirement specified above is from the travel agent business logic viewpoint. Real world B2B integrations are peer-to-peer in place of being centralized. It is more like contract in terms of performing responsibility in the collaborative work, as a result such integration requires a notion to specify " I will provide this functionality to achieve this from all my partners (other businesses)".

BPEL specification claims to facilitate the B2B integration using abstract BPEL, where each party describes their B2B participant functionality using BPELa file. Hence all such businesses can publish their own BPELa files and can work together. But again if these BPEL files are not developed in collaborative way at first place, then the consumer-producer relationship cannot be achieved using BPELa. For this reason, the bottom-up approach i.e. implementation first and then description adopted by BPELa is not adequate for the B2B relations. Therefore, the top-down approach that describes functionality first and considers implementation at later stage is better suited for such B2B domain Web Services composition. Overall, BPELa provides notion to specify only "this is the functionality I provide" in place of required collaborative notion "I will provide functionality to achieve this from all my partners (other businesses)" making BPELa inadequate for B2B integration.

To consider the facilitation provided to the composition participants, in case of service provider if provider wants to make their service available for composition then they need to provide minimum functionality required by the business logic from the single party perspective in the integration. Considering new AirUK service for travel agent composition, AirUK to be part of the composition, options for the service provider of the new AirUK service will be:

a) If the AirUK has web service but does not implement required functions, then the service needs to be modified to accommodate the required functionalities. WSDL file for this service can be made available to the travel agent via UDDI. This WSDL file can contain only those operations useful to the agent service.

b) If the AirUK has a non-web service application, with the required functionality already built-in, then just a WSDL file is required to be created without modifying the existing non-web service application. As such Non-Web Services, which have WSDL, can still be composed in BPEL. BPEL execution engine uses WSIF (Web Services Invocation Framework)[12] for the Invocation of such non web-services [13].

Considering the case of service composer, they mainly encounter problems in parameter mismatch during the flow management, as the output of one service operation has different format from the input of next service operation in the flow logic. The other case might be the bit of missing functionality, which probably is already available as legacy code. To address both the case, BPEL implementations [11] provides a mechanism where the higher language code(C#, Java) can be inline in the BPEL process. BPEL-J [14] is another such an industrial effort to combine BPEL and Java, where BPEL is for defining Business processes and Java to provide general programming language expressiveness, allowing each language to do what it does best.

To illustrate the case of end-user, in our travel agent example, the BPEL agent process is published using JSP technology. This service can be retrieved using this simple web page or can be retrieved from the public UDDI registry, where the access point in the business service binding information points to this JSP page. In

such B2C interactions, it is totally transparent from the end-user that the service is a Web Service, composition of multiple Web Services, could be implemented on heterogeneous platforms or using different programming languages. As Semantic Web [15] introduced a scenario, where the intelligent software programmes -agents work on behalf of the end user, and can use composition to satisfy user requirement or task however BPEL, in its' present form has nothing to offer to facilitate this approach. The similar way it has little to do with automatic service composition.

To conclude, BPELe is the best candidate for composing private, inner-organization Web Services making suitable for EAI. Business analysts and developers can work together and can compose such Web Services manually using BPELe. The composition is hard coded and the developers should have the explicit knowledge of all the details of participating business services. The control and data flow logic also should be known in advance. BPELe can also facilitate tightly coupled B2B integration; in contrast BPELa is poor candidate for describing peer-to-peer B2B collaborations.

3. WEB SERVICES COMPOSITION APPROACHES IN THE NEAR AND THE DISTANT FUTURE

3.1 Composition using WS-CDL

During our discussion over BPELa as B2B integration language, we highlighted two main requirements adequate for B2B integration. First, a description language is required to facilitate top-down, using which collaborative functionality in B2B integration is described, agreed first and the respective parts can be implemented by each partners. Second, the description should be from peer-to-peer point of view, since natural B2B integrations are peer-to-peer collaborative relationships, not governed by a single party. Recently promoted working draft WS-CDL [6] from W3C Web Services choreography working group confirmed our conclusions that more work on BPEL is required to make it adoptable for B2B integration by addressing these requirements and adopting approach specific to the B2B relationship.

WS-CDL is a description language, using which the B2B integration partners can first describe the collaborative functionality. This description document is considered as a contract and each party can implement their own part. The WS-CDL document describes common and complementary behaviour of all the parties involved, making the viewpoint global, peer-to-peer [6]. Travel agent B2B integration previously illustrated in figure 1 transforms into figure 2 under WS-CDL architecture, where travel agent is no longer the controller of the integration, in view of the fact that the respective functionality and the ordering of the activities performed is from the perspective of all the parties involved and agreed by all in the CDL document.

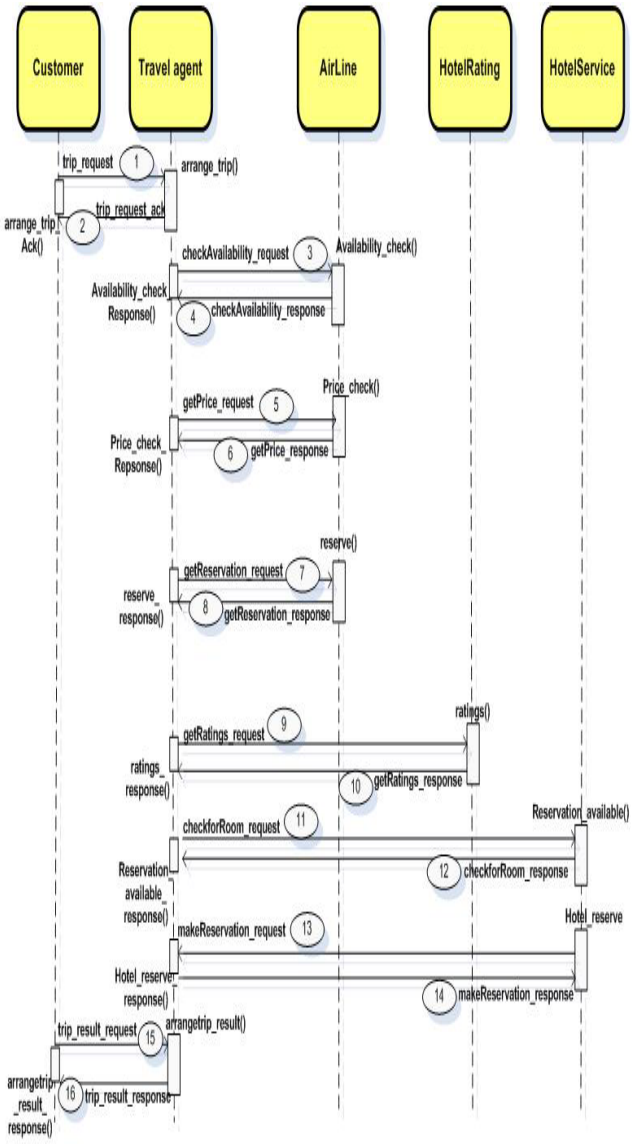

Fig 2. Travel Agent B2B integration scenario using CDL [global viewpoint]

CDL document describing this scenario becomes contract between participating parties in terms of the functionalities they agree to provide. For example, considering the interaction between businesses Travel Agent (TA) and AirLine (AL) Services in activity (5,6): TA interacts with AL service for checking the price (activity 5) for the required flight; for that reason AL provides a web service operation Price_check, So that TA can send getPrice_request. And TA has the Price_check response operation, which AL uses to send the response message.

Listings 4 and 5 describe this interaction in CDL. CDL has the notion of role and relationship for the participating parties in particular interaction. The interaction PriceCheckRequestInteract in figure 9, documents the fact that it is the fifth activity in sequence, the participant in the interaction are those who have TravelAgentAirLineBinding relationship, operation affected is the AirLine Web Service operation Price_check and getPrice_requestAtTravelAgent (Unique Identifier for getPrice_request) message exchanged from the role TravelAgent to AirLine.

```
<roleType name="AirLine">
   <behavior name="airline"
            interface="AirLinePT"/>
</roleType>

<roleType name="TravelAgent">
   <behavior name="TravelAgentForAirLine"
           interface=" TravelAgentForAirLinePT"/>
                              ……
</roleType>

<relationshipType
            name="TravelAgentAirLineBinding">
           <role type="AirLine"/>
           <role type="TravelAgent"  behavior
            ="TravelAgentForAirLine " >
                              …..…….
</relationshipType>
```

Listing 4. Roles and Relationships

```
<interaction name="PriceCheckRequestInteract"
    operation="Price_check" …..……. >

<participate  relationship="Travel
  AgentAirLineBinding"
  fromRole="TravelAgent"
   toRole="AirLine"/>
   <exchange name="request"
                    action="request">
    <send variable="getPrice_requestAt
                  TravelAgent"/>
    <receive variable="getPrice_reqeustAtAirLine"/>
   </exchange>

</interaction>
```

Listing 5. Interaction

The other aspect of CDL architecture is that the internal business logic of each party remains hidden from the business partners. i.e. in this travel agent application, after getting price from airlines, travel agent can have internal application to implement the business logic for the air line selection based on certain criteria, while the external detail described in CDL document is just an operation to make reservation at particular airline. This internal logic can be an EAI application composed using BPEL.

If considering the facilitation provided to the participants in CDL, service composer designs the global interface CDL and as the other parties follow the interface, composer does not have to deal with individual service providers and can easily accommodate them once providers follow the global interface. Having a global interface also liberate composer from problems of functionality and technicality mismatch. CDL is still descriptive language but can play the role like WSDL, to create stub files so that each party service provider can have blue print of what they are supposed to implement [16]. Consider the benefit of having such descriptive language with top-down approach when the integration takes place between large numbers of Web Services.

Overall, CDL is designed to address the requirements for B2B integration and compliments BPEL, which is a better candidate for EAI. Consequently, CDL and BPEL together address the problem of BPM by facilitating static composition.

3.2 Dynamic Composition

Commercial Institutions are focusing their efforts on standardizing the static composition techniques in preparation for their wider adoption amongst the business community. In contrast, research community foresee that there is a better futuristic potential in the semi-automatic and automatic or dynamic composition of Web Services. Dynamic composition achieved can serve a great range of business domains. In such kind of composition participating services can be external and public. User can specify parameters for the successful composition and the composition is performed at the run-time. The solution addresses the problems of identifying candidate services, composing them, and verifying closely that they satisfy the request.

Dynamic Web Services composition is the topic of our ongoing research. As per our definition of composition automation, a semantic based language specifying the capability of Web services is required so that services can be selected on the fly for the composition according to user parameters. Semantic web based OWL-S [17][18] can be utilized to achieve this. A layer on top of OWL-S is required for: automating flow management, interpreting the semantic web service based capability, and to manipulate data before invoking operations and to carry out execution according to the user requirements. Artificial intelligence planning, workflow management, and intelligent agents etc. are the available options.

Using dynamic solution, users will get the maximum flexibility, as composed services will be an optimum mix based on the user specified input parameters. The service providers will be able to participate in the composition to their benefit with minimal effort, as the human developer will be taken out of the loop.

4. CONCLUSIONS

Web Services composition approaches are characterized as static, semi-automatic and dynamic. Well-known industrial standards like BPEL and CDL facilitate composition as the BPM solution. BPEL categorized into BPELe and BPELa is designed to address intra and inter organization BPM integration problem. BPELe architecture makes it better suited for enterprise level integrations while BPELa is a poor candidate for B2B integrations. CDL is an effort to overcome the limitations of BPELa and covers from where the BPEL has left. These standards satisfy current business requirements by adopting static composition.

In this paper, we also briefly explored the research efforts into dynamic composition of Web Services, and noted that future composition solutions based on OWL-S can be applied to wider range of business applications to facilitate machine-readable, agent based automatic Web Services composition scenario.

REFERENCES

Gudgin M et al. 2003. “Simple Object Access Protocol(SOAP) Version 1.2”, W3C recommendation, 2003, http://www.w3.org/TR/soap12-part1/

Christensen E et al. 2001. “Web Services Description Language(WSDL) version 1.1”, W3C recommendation, http://www.w3.org/TR/wsdl/

Hately A et al. 2003. “Universal Description Discovery and Integration (UDDI), Version 2.0” OASIS Specification, http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm#uddiv2

Osman T et al, 2005. An Integrative Framework for Traffic Telematics Web Services, to be appeared in the *Parallel and Distributed Computing Networks Conference(PDCN 2005).*

Leymann F at al. 2005. “Web Services and business process management”. *IBM Systems Journal*, Volume 41-2, 2002, 198-211.

Andrews T et al, 2003. “Business Process Execution Language for Web Services, Version 1.1”, http://www-128.ibm.com/developerworks/library/wsbpel/

Kavantzas N et al, 2004. Web Services Choreography Description Language (WS-CDL) Version 1.0, http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217/

Arkin A, 2003. “Business Process Modelling Language Version 1.0”, http://www.bpmi.org/bpml-spec.htm.

Staab S et al, “Web services: been there, done that?” *IEEE Intelligent Systems*, Vol. 18, Issue. 1, Jan 2003, 72- 85.

BPWS4J, “Business Process Execution Language for Web Services Java Run Time”. http://www.alphaworks.ibm.com/tech/bpws4j.

Oracle BPEL Process Manager (PM), http://www.oracle.com/technology/products/ias/bpel/index.html.

Web Services Invocation Framework, Apache Software Foundation Web Services project, http://ws.apache.org/wsif/

Afshar M et al, 2004. “WSIF and JSR-208: Flexible binding frameworks for today and tomorrow”, *Web Services Journal*, http://www.sys-con.com/story/?storyid=46558&DE=1

Blow M et al, 2004 “BPELJ: BPEL for Java”, White paper from BEA and IBM, http://www-106.ibm.com/developerworks/webservices/library/ws-bpelj/

Lee B at al, 2001. “The Semantic Web”, *Scientific American*

Austin D et al, 2004. “Web Services Choreography Requirements”, *W3C working draft*

Martin D et al, 2004. “Semantic Mark-up for Web Services, OWL-S version 1.1”

Martin D et al, 2004, "Bringing Semantics to Web Services: The OWL-S Approach", *Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*.

AUTHOR BIOGRAPHIES

DHAVALKUMAR THAKKER took his Masters degree in Data Communication Systems at Brunel University, London-UK. He also received his bachelors degree in engineering from India. He is a PhD student of Nottingham Trent University, School of computing and Informatics. His current research interests are distributed computing technologies, Web Services composition, Semantic Web and Ontologies. His email address is dhavalkumar.thakker@students.ntu.ac.uk.

TAHA OSMAN is senior lecturer at the Nottingham Trent University, UK. He received a B.Sc honours degree in Computing from Donetsk Polytechnical Institute, Ukraine in 1992. He joined the Nottingham Trent University in 1993 where he received an MSc in Real-time Systems in 1994 and a PhD in 1998. His current research concerned investigation of Fault-Tolerance in Open Distributed Systems. His email address is taha.Osman@ntu.ac.uk.

DAVID AL-DABASS is professor at the Nottingham Trent University, UK. His research work explores algorithm and architecture for machine intelligence. His email address is david.al-dabass@ntu.ac.uk and web page is http://ducati.doc.ntu.ac.uk/uksim/dad/webpage.htm.