IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 2, Ver. II (Mar-Apr. 2014), PP 04-14
www.iosrjournals.org

Agile Web Service Composition and Messaging approach for e-Government services

Avinash Ramtohul[1], Prof. K.M.S Soyjaudah[2]
[1]*PhD Student, Faculty of Engineering, University of Mauritius*
[2]*Professor, Faculty of Engineering, University of Mauritius*

Abstract : *E-Government services are increasingly being deployed using service-based architectures. Individual web services, developed from legacy and modern firmware, are composed to achieve e-service delivery. Current web service composition approaches range from practical languages aspiring to become standards (like BPEL, WS-CDL, OWL-S and WSMO) to theoretical models (like automata, Petri nets and process algebras). In this work, current approaches were studied and a comparison of their features and weaknesses was carried out. It was observed that current composition mechanisms are static and not agile enough to cater for functional change in business needs. This paper proposes a new web service composition and messaging mechanism for electronic government services in which composition of web services is done dynamically to provide for more agility in government e-services. The proposed composition mechanism works on a software architecture which enables application software integration. This architecture has been developed to deploy end-to-end e-services to G2B, G2C and G2G users starting from legacy government software applications. Using the proposed approach, integrating back-end e-Government applications to deliver end-to-end e-services can be agile, simpler and faster. The main contribution of this paper is the novel dynamic web service composition mechanism.*

Keywords: *e-Government, SOA, Business Process Management, Software architecture*

I. INTRODUCTION

Web services (WSs) are software modules which combine and interact with each other. One of the main features of WSs is re-usability. Generally, the higher the granularity of WSs, the more is the re-usability and the number of web services. However, a higher number of web services entail a more complex composition and execution of web services. Composition rules describe how to compose web services in simple and complex environments. Fundamentally, they specify the sequence in which, and the parameters under which, Web Services may be invoked. There are two types of WS Composition mechanisms: Syntactic (XML-based) and Semantic (ontology-based). Research on web services focuses on how to formally specify them, compose them in an automated manner, expose them on a network and validate their functionality. Business process automation creates the underlying engine on which web services work. Among the most important business process automation technologies are Business Process Execution Language for Web Services, BPEL[1] and the Web Services Choreography Description Language WS-CDL[2]. These languages support mostly static WS composition due to inadequate semantic representations of WSs available on the Internet. Consequently, various solutions were proposed by the Semantic Web community and other working groups. These include the Web Ontology Language for Web Services, OWLS[3] and the Web Service Modeling Ontology WSMO[4].

Web service composition methods have been a subject of active scientific study. While some researchers focused on Web service composition and related aspects[5] [6], others explored the QoS dependencies[7]. Further work was done on pre-/post-conditions of web service composition[8-12]. The study of the importance of user constraints and preferences revealed interesting impact on web service composition patterns[13-15]. Consideration of the user context and complex dependencies between services and the transactional behavior of Web service compositions was also studied. Current research focuses on aspects such as quality of service and pre/post conditions. A SOA based architecture and a novel approach for rapid web service development and deployment of e-services on existing legacy e-Government application systems was also developed[16]. In the field of web service composition, researchers argued that the main weakness of existing methods is their inability to dynamically compose web services depending upon the business context, especially in e-Government systems.

No in-depth research seems to have been undertaken to define agile web service composition methods in the field of e-Government. Given the increasing need for agility in e-Government services, dynamic web service composition to develop and deploy e-services is more than ever important. In this paper, a new and agile web service composition and messaging method for developing and deploying e-Government services, is proposed. Based on semantic WS composition, this new method can automatically compose web services depending on the business context. Therefore, the proposed method enables business processes to be

dynamically constructed and executed using a superset of pre-defined web services, thus increasing the agility of e-Government software systems and e-services. This method uses the concept of doubly linked-lists to represent e-Government service flows on main memory. Therefore, business process execution thread reads business context parameters (contained in nodes of linked lists) and dynamically chooses web services to compose with (by traversing the linked list dynamically wherever a pointer/link exists).

II. SYNTACTIC WS COMPOSITION METHODS AND LIMITATIONS

There are two main approaches for composition of syntactic Web services. Referred to as 'WS orchestration', the first approach composes WSs and controls execution by using a central coordinator known as 'Orchestrator'. The main function of the orchestrator is to invoke and combine granular software functions and modules. The second approach is called 'WS choreography' which does not use a central coordinator. This approach uses the definition of the conversations between each pair of WS to specify the composition that should take place; the overall definition is obtained by combining the flow of peer-to-peer conversations between WSs. Though there are several proposals for orchestration languages, BPEL has emerged as the most important one. However, the choreography technique did not develop any further[17].

BPEL4WS is an XML-based language designed to combine, coordinate and execute web services. This language is based on the Web Services Description Language WSDL, an interface description language for WS providers. BPEL4WS uses a workflow approach in order to execute WSDL. The workflow expresses relationships and web service interactions using control and data flow links. BPEL4WS can be executed in centralised or distributed models using variables to carry data. Process definitions represent the main construct for modeling the flow of WSs.

WS-CDL is an XML-based specification language which works on composition of interoperable web services. Further, long running, peer-to-peer collaborations between WS participants with different roles can also be specified by WS-CDL. A choreography description defines the composition of participating Web Services. The most important element of WS-CDL is the INTERACTION activity which describes an information exchange between parties. This interaction creates message transfer from one web service to another, but focus is laid on the receiver. During any interaction, three elements are considered: Communication with the participants, Information which is being communicated and the Channel through which information is transferred. Management of exceptions and compensations is done through 'exception and finalizer work units'. Data, in form of messages, is transferred between participants and represented using variables and tokens. The XML schema or WSDL contains the definitions of the data type for these variables and tokens. Channels are used to define the way in which message exchanges take place between Web Services. These activities are synchronised as a work unit. The latter defines the pre-conditions that must be satisfied for the activity to continue.

III. SEMANTIC WS COMPOSITION METHODS AND LIMITATIONS

Available Web Service technologies cater only for the syntactic aspects of WSs resulting in a set of rigid WSs that are static and unable to adapt to variations in the environment. Such variations can only be handled programmatically following human intervention. Two main initiatives are discussed, namely, Ontology for Web Services Logic Semantic (OWL-S)[18], and Web Services Modeling Ontology (WSMO) developed by the WSMO working group. The former defines an ontology for the semantic markup of Web Services. This ontology aims at enabling the automation of WS development Cycle, ranging from discovery through invocation, composition, interoperation and execution monitoring by providing appropriate semantic descriptions of WSs. The WS Modeling Ontology WSMO aims at creating an ontology which will describe various aspects related to semantic WSs for solving software integration problems. OWL-S and WSMO both share the same goal of providing an industry standard for semantically describing Web Services.

OWL-S specifies web service semantics using four elements, namely, Service profile, Service model, Process model and Service grounding. The concept of SERVICE defines a starting point for the web service to have a context. The Web service is declared by creating a SERVICE instance which links the remaining three elements of a Web Service through properties such as PRESENTS, DESCRIBEDBY and SUPPORTS. The SERVICE PROFILE defines what a WS does from a high-level perspective. The functional and non-functional properties of the Web Service are described. Therefore, using the Service profile, both the web service provider and consumer can specify the functional properties of WS. The Service model uses the Process model to define the main and sub processes and describes how a WS achieves its intended functionality. Finally, Service grounding describes how WS can be invoked.

In WSMO, the conceptual design set up in the WS Modeling Framework (WSMF)[19] is used to specify Web service semantics. WSMF includes four distinct elements, namely, ontologies, web Services, goals and mediators. WSMO uses these elements to further refine the semantics. The key element is ontologies since they contain the super set of domain-specific terminologies which describe the other elements. Further, they

bridge human and machine terminologies by formal semantics. Web Services run on industry standard protocols such as http in order to exchange and combine messages and data. Properties of ontologies can be described from three different angles: behavioural, functional and non-functional. Goals specify the objectives of a client when consulting a WS, i.e. the functionalities a WS should provide from the user's perspective. Mediators, finally, aim to overcome the mismatches appearing between the different elements constituting a WSMO description. Their existence allows links among both homogeneous and heterogeneous systems.

The first limitation is that OWL-S does not separate end-user needs and the available functionality of Web Services. The name, human-readable description and contact information are not expressed in a standard metadata format. The service profile is expressed differently in WSMO which recommends acceptance of a set of agreed vocabularies such as the Dublin Core[20]. Moreover, WSMO supports the expression of non-functional properties using any WSMO element, but in OWL-S only service profile elements can be used to define non-functional properties. Further, the service model of OWL-S does not differentiate between choreography and orchestration since no formal model is available. It should be noted, however, that some work on defining the formal semantics of OWL-S processes has been completed. In summary, orchestration describes how other Web Services are composed in order to achieve the required functionality of the WS while choreography describes the external visible behavior of the WS.

IV. PROPOSED COMPOSITION AND MESSAGING APPROACH

This section discusses the novel agile web service composition and messaging method which addresses the lack of dynamic web service composition in existing methods. Based on semantic WS composition, this new method can automatically compose web services depending on the business context. Therefore, this method enables business processes to be dynamically constructed and executed using a superset of pre-defined web services. The proposed composition and messaging method uses an orchestration mechanism based on semantic web service composition. It has been developed in the context of an overall framework, called Service-based architecture for e-Government (SBA-eGOV) designed for implementing service orientation to deploy secure e-services onto an existing e-Government infrastructure shown in fig. 1.

To explain the design and workings of the proposed composition and messaging method, it is important to briefly discuss this framework. The principles of SBA-eGOV are:

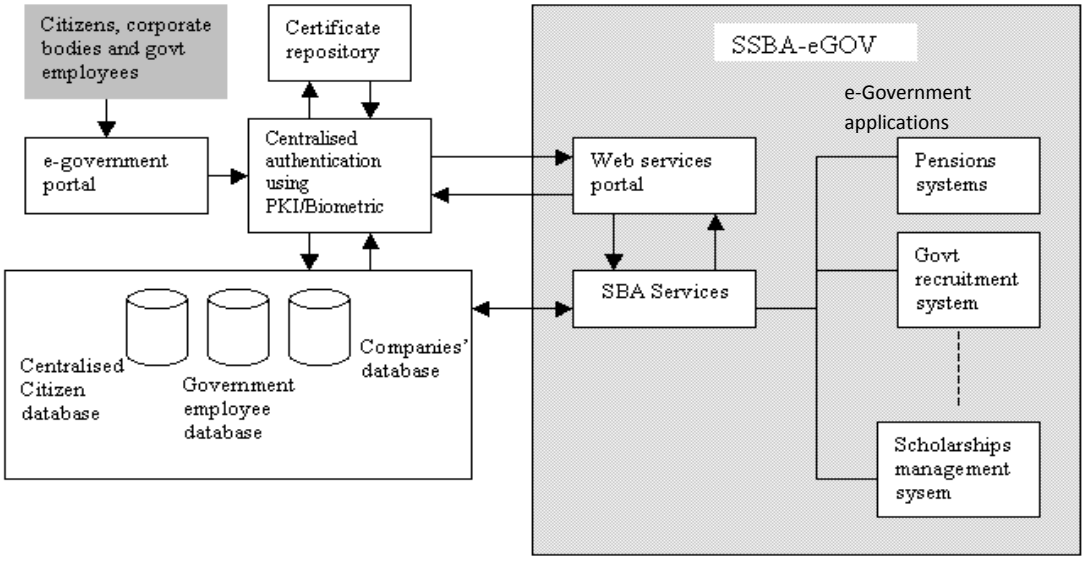

Figure1: Architectural map of e-Government using SBA-eGOV

(i) Integrate back-end legacy systems to develop/implement web services.
(ii) Use a business process driven approach to compose web services and deploy G2C, G2B and G2G e-services.
(iii) Enable messaging between web services

This paper focuses on composition and messaging principles of SBA-eGOV. Semantic composition approaches bridge machine and human terminologies by formal semantics which can be expressed as special notations and mapped onto a process flow. Generally, every G2B, G2G and G2C service has a pre-defined flow that maps onto one or more web services. For example, an e-service that allows citizens to obtain an appointment for their

vehicle fitness examination invokes the web service *appointment_for_ veh_fitness_test*, which, in turn invokes several other web services before responding to the end-user.

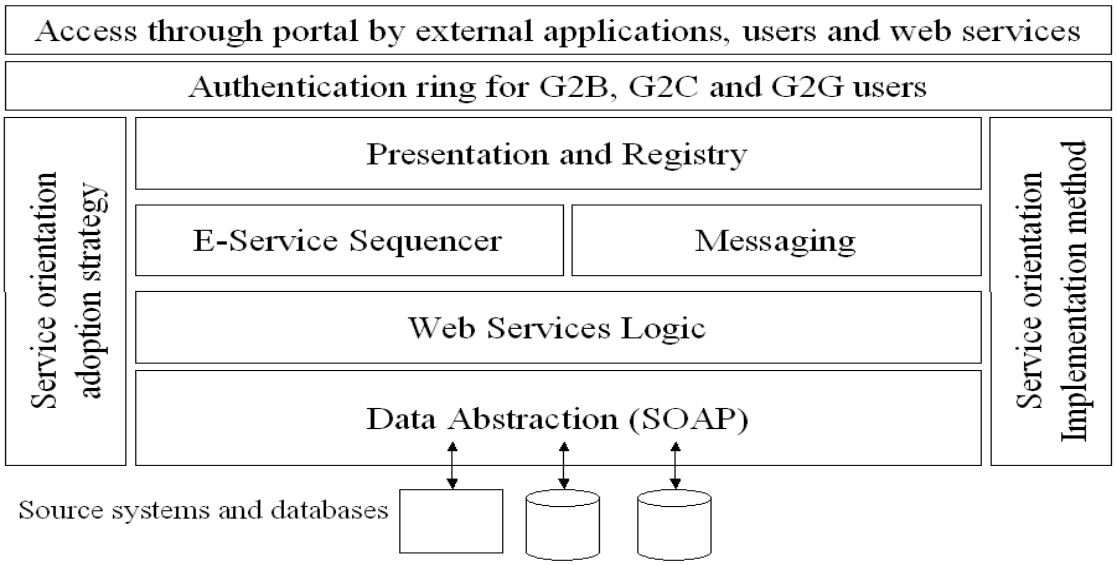

Figure 2: SBA-eGOV architectural model

The set of web services invoked also include those, which read data from the source software systems and execute built-in logic.

To create e-service from source systems (e.g. Pension systems, Recruitment system, Scholarship management system), the latter systems need to expose a set of interacting web services deployed on a portal. SBA-eGOV enables integration between source systems so that their native functionality can be deployed as simple or composite web services. Composition and messaging is done by e-service Sequencer and the Messaging layers of the SBA-eGOV software architecture. The Data Abstraction layer reads data from source systems using Simple Object Access Protocol (SOAP) and sends the data for processing to the Web Services Logic layer. The latter consists of the business logic which processes data read from source systems by the Data Abstraction layer. The software architecture presented in fig. 2 shows the various layers discussed in this section.

Execution of the web services logic is controlled using the e-service sequencer which also handles composition and messaging of web services. The presentation and registry layers maintain information on web services and information on how to call them. In order to illustrate how the proposed composition and messaging scheme works, the following web services are used.

Web_service (check_veh_details) is a web service exposed by the Vehicle Registration Authority information system, which houses a database of vehicles. It checks the registration number of the car against the National Identity Number of the owner supplied by the citizen accessing the *appointment_for_veh_fitness_test* e-service. It also checks the registered address of the vehicle and selects the closest vehicle examination centre location. ***Appointment_for_veh_fitness_test*** e-service is a service provided to citizens on the portal for booking an

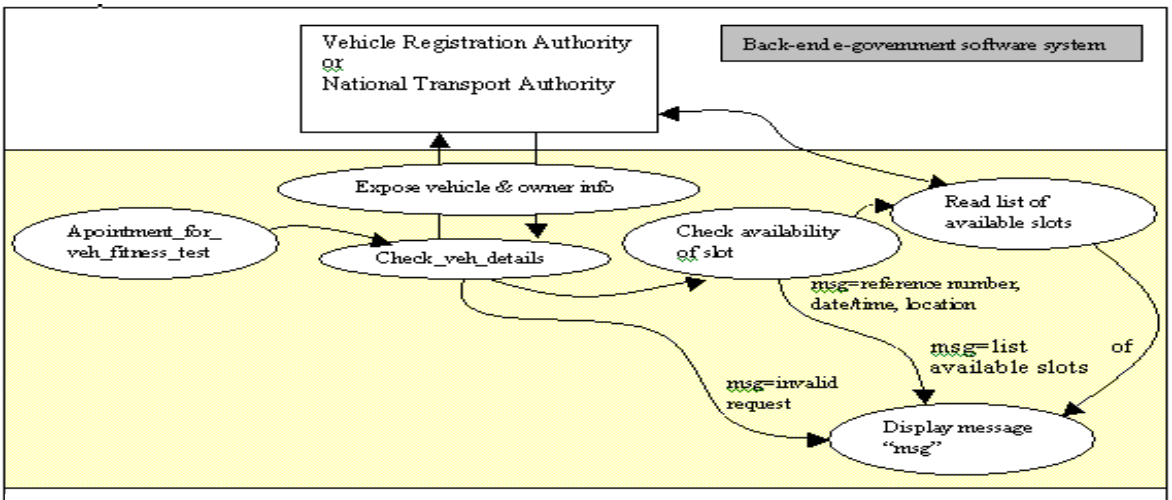

Figure 3: Web Service sequencing for e-service: Request for appointment for vehicle examination

e

appointment for vehicle examination.
Web_**service *(generate(reference number, date time, fitness test location id))*** is a web service that selects the location of the vehicle examination centre, reads the available slots for vehicle examination and generates a reference number, date and time of appointment.
Web_service (*display_available_slots*) is a web service, which displays the set of available slots. It also prompts the user to select an available slot from the list.
SBA-eGOV interacts with legacy systems to read, write and process data. The web services that are exposed by back-end e-Government applications implement abstraction of data to hide the complexity of data structures and present data as objects. For instance, in the scenario of fig. 3, the Vehicle Registration System, exposes vehicles as an object, accessed by a web service known as check_veh_details.

A. Presentation and Registry

While all other layers of SBA-eGOV are transparent to the end-user, the presentation layer consists of a mechanism to expose web services to users. An overview of web services and their input/output parameter specifications are displayed to users and to other web services wishing to consume web services from this provider.

Web services exposed on this layer can be composite or simple. A composite web service is a sequenced collection of several web services for deploying an e-service completely or partially. For example *Apointment_for_veh_fitness_test* is a composite web service consisting of simple web services such as *Check_veh_details* and *Check_availability_of_slot*. Simple web services are re-usable components within SBA-eGOV. *Check_veh_details* can be shared with another web service which can also be accessed by the Law enforcement division for investigation purposes. SBA-eGOV implements Web Services Registry using a database table. Table 1 holds information about all web services covering all events in their life cycle, ranging from creation to deletion.

Table 1: Web services registry implementation using database table

Web Service ID	**Web service name**	**Web service description**	**Web service input parameter name and type**	**Web service output parameter name and type -** Data structure
WS001	*Check vehicles details implemented as Check_veh_details*	Checks the details of vehicles including registration mark	Vehicle registration mark, NID	Make, model, engine rating
WS002	*Check availability of slot implemented as Check_availability_of_slot*	Verifies availability of the requested slot for appointment	Date/time	1: Available 0: Not available

B. E-service Sequencer, messaging and web services logic

Composing, sequencing and messaging play a major role in any service-based architecture and can be of type orchestration or choreography. This section discusses the proposed method, based on orchestration, for achieving these major functions. The process of calling and executing several web services in a given chronology is referred to as composing web services. Exchange of information between web services is called messaging.

Using the sequencing feature of the proposed web service composition method, a complete business flow can be represented in form of web services. Thus the sequencer becomes aware of business processes. The e-service sequencer therefore replaces the Orchestration layer found in traditional SOA framework.

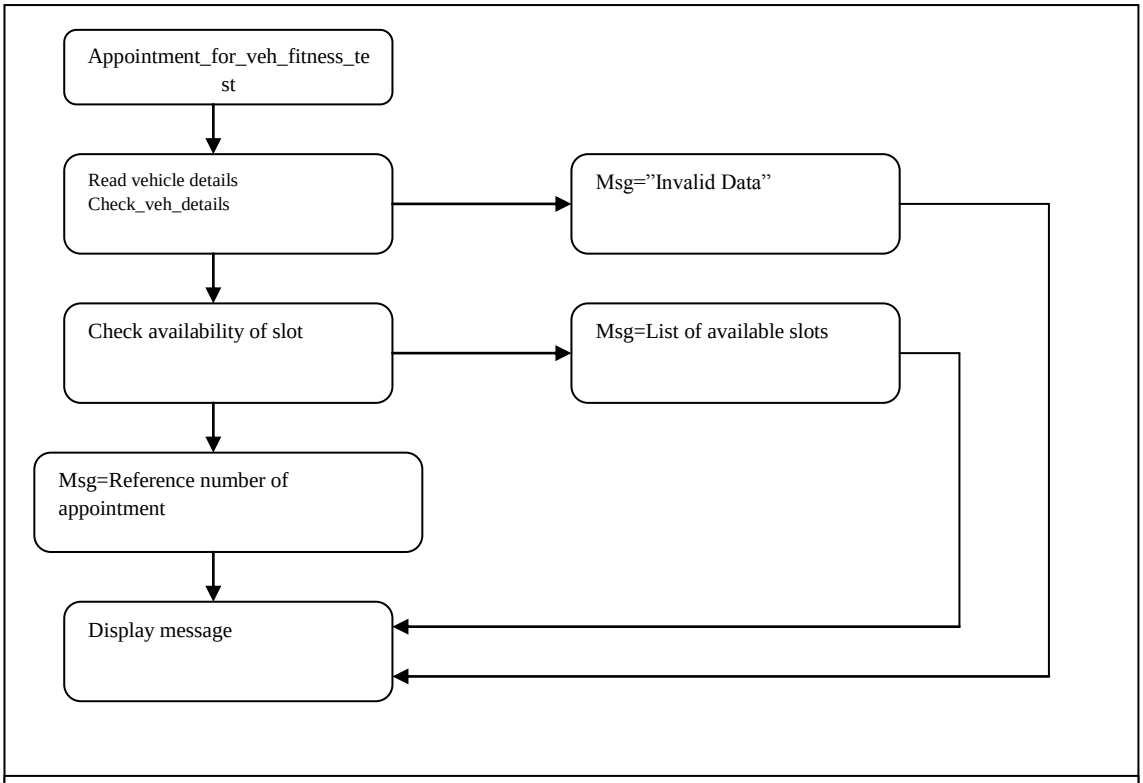

Figure 4: High-level representation of a typical business process

The sequencer

(i) executes pre-defined business flows in a sequence
(ii) makes calls to web services based on a business process flow
(iii) sends and receives messages in form of parameters to and from web services

1) Translating business flows into web services

A business process can be broken down into sub-processes or modules and hence in form of sequenced process flow, consisting of sequenced sub-processes. A business process can be sequenced manually/programmatically or using a process-modeling tool. Fig. 4 shows the business flow of an e-service, Appointment_for_veh_fitness_test, represented in form of sub-processes. Since the flow is sequenced, the sub-processes are also sequenced.

In traditional business process management notification methods (BPMN), these processes can be stored in form of XML files. The proposed method represents business process flows in terms of doubly linked lists. A linked list is a data structure for storing relational information in active memory. Linked lists consist of a collection of nodes (memory addresses) pointing in a single direction (singly linked lists) or in both directions (doubly linked lists). In the process flow of fig. 4, each process box is a node of a doubly linked list. Each node represents a call to a web service and sends the relevant parameters in form of a data structure to the next node (right, top or bottom). Every node of the linked list contains a data structure that stores sequencing information and data messages. This enables the current node to know which node it should point to and what parameters the next node will require.

2) Defining business flows using linked lists

Linked lists are capable of holding data structures of complex nature. Every node has storage capacity for storing information and can also be linked to others nodes. Generally, a linked list is identified and accessed using a handle, known as pointer, which points to a memory address. Once identified, the first pointer is used to access all the nodes of a list. Using this mechanism, the linked list can be accessed sequentially from one end to the other. Linked lists can also be extended to point to as many nodes as required. This property provides the possibility of using linked lists to model business processes by representing processes as nodes and process branches as pointers, illustrated in fig. 5.

As described earlier, a process or sub-process box represents a call to a web service. Since a process can be modeled using a linked list, every node therefore, represents a call to a web service. Hence, using the

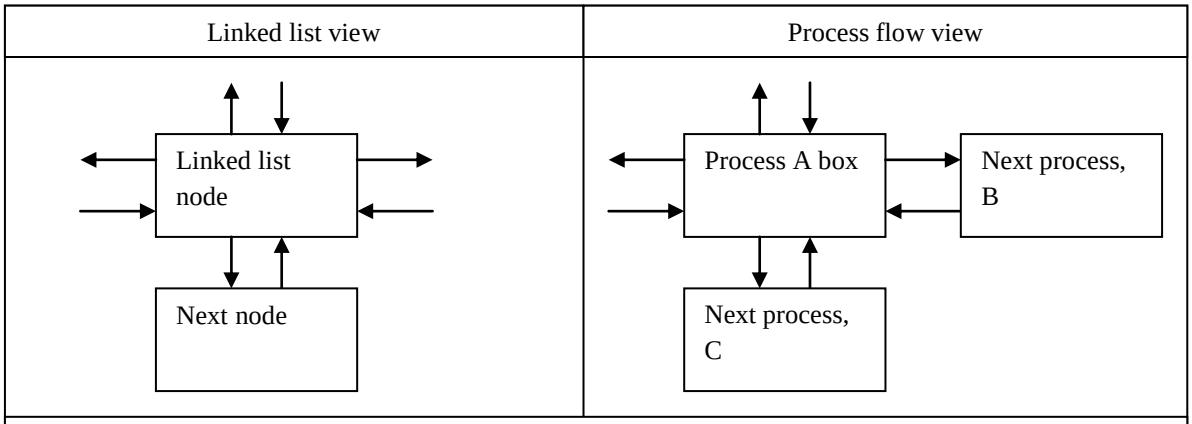

Figure 5: Using linked lists to represent process flows

example of *appointment_for_veh_fitness_test* e-service, the process flow of fig. 4 can be mapped onto a linked list to represent a complete business flow, shown in fig. 6. Thus, nodes can be referred to as web services and linked lists referred to as processes. All the nodes have the same data structure that stores information about the current node, the next node and the web-service(s) to be called. Information on parameters and their values are exchanged among web services (nodes) through the pointers which point to each other (left, right, top, bottom) in the linked list. The process flow could have been represented by a singly linked list; however, parameter values required by the current node are always stored in the previous node. Hence, it is important for a node to be able to traverse back to the previous node to read the parameter values. Web service execution requires parameter values that are computed in the node that precedes the current node. Therefore, the current node has to use the back or top pointer to read data of the preceding node. Using this mechanism, SBA-eGOV implements exchange of data between web services. Every node can hold 50 parameters. It is also possible to extend the list to accommodate more parameters. Below is the algorithm to create the data structure for the linked list.

1: Define structure for the linked list nodes
2: Initiate variables

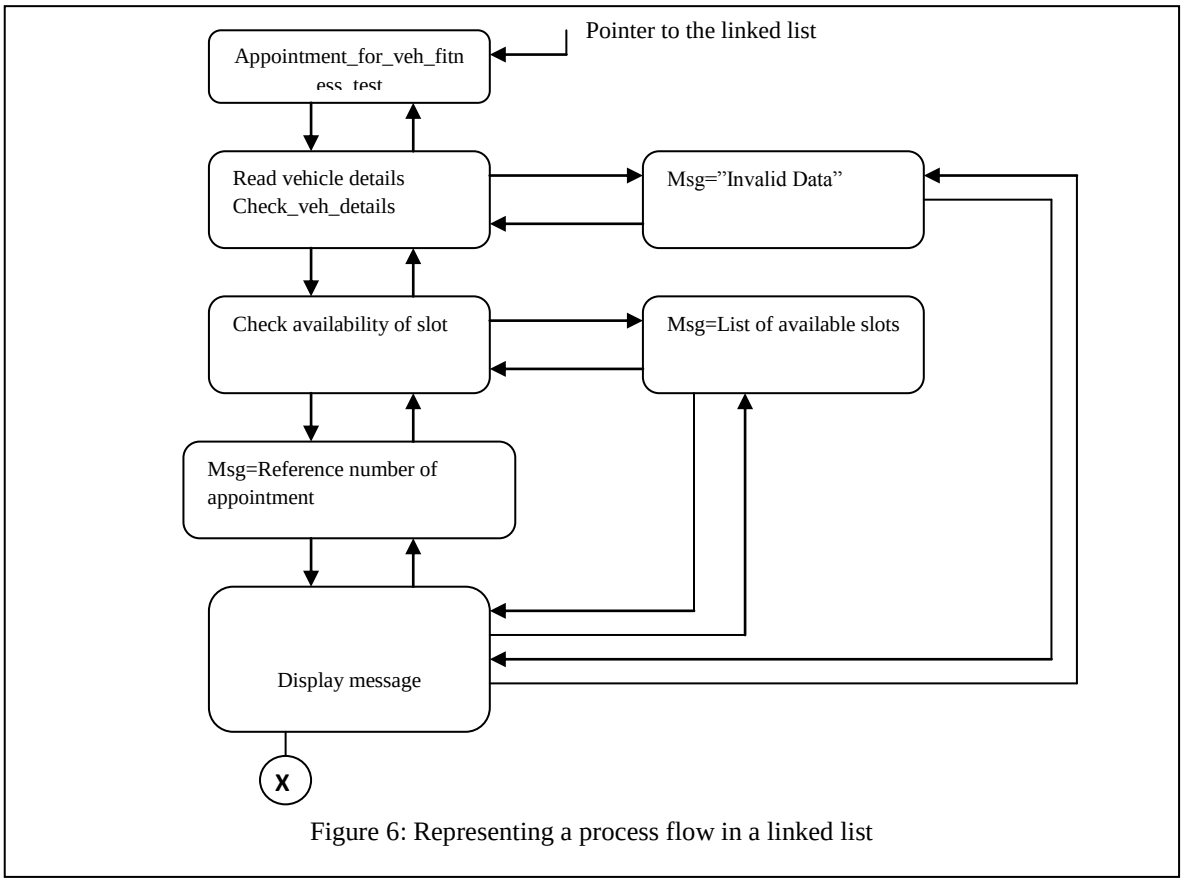

Figure 6: Representing a process flow in a linked list

3: Define variables for storing web_service_id, web_service_name, web_service_url
4: Define array to store parameter names and parameter values
5: Define pointers which point to right, left, up and down
This structure defines the contents of every node of the linked list and contains meta-data regarding the web service, metadata of parameters and links the node to other nodes as per the process definition. Further, the data structure also creates the necessary links to other nodes in order to represent the business process.

3) Messaging: Data exchange among web services

The use of doubly linked lists provides the possibility of accessing the previous and the next node from the current node. Any node can also read parameter values from any other node in the list. This feature is critical in the proposed messaging mechanism as it allows moving data from one web service another. Thus data can be transferred from one web service to any other linked web service in the business process. The next algorithm shows how the linked list is built and how data is exchanged between web services.

1: Allocate memory to new node having the structure defined above
2: Read the web service id from the Registry layer when user clicks on an e-service
3:Name linked list as 'Appointment_for_veh_ fitness_test'. This is also the e-service name
4: Write value of the web_service_url onto a variable 'http://www.mauritius.gov/webservices /Appointment_for_veh_fitness_test?'
5: Let web_service_id= '101', Obtained from the web-services registry
6: Let web_service_name= 'Appointment_for_veh_ fitness_test'
7: Build the web service call URL by using the parameter values read from the user
8: Set values web_service_msg_param_name to veh_registration_mark value is 'AC 22'
9: Set web_service_msg_param_name to veh_owner_NID and value to 'R1510198523255'
10: Set web_service_msg_param_name to 'Appt_date' and value to '5-Jan-2012'
11: Point node to null

4) Sequencing web services

The vehicle details shown above are entered by citizens on the portal, captured by an HTML interface and sent to the web service *appointment_for_veh_fitness_test* using POST method. This web service generates the linked list shown in fig. 7. The main pointer is called 'Head' and is the first node in the list. A second node, Node 2, is then created. This node calls the web service *check_veh_details* by using the vehicle parameters that are located in the first node. These two nodes are shown in fig. 8. *Check_veh_details* validates the vehicle registration mark against the owner's national identity number. The result of this validation check is then stored in the current node, i.e. Node 2. This composition and messaging method also has the capability of handling 'if…then' by branching the linked list right, left, top or bottom in order to create or access the next node. Thus, decision boxes appearing in process flows are also implemented using linked lists. Inherently, the branching can be dynamic depending upon the parameter values and the outcome of the validation checks. The next algorithm creates the second node, Node 2, which has double links to the first node.

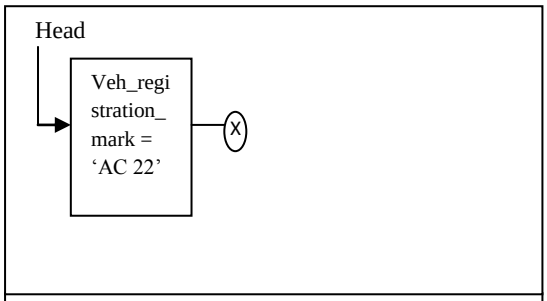

Figure 7: Building up the linked list for this web service

1: Start second node and allocate memory
2: Set the web_service_id to '102' which is an identifier of the next web service which is composed with web service 101 to deliver the e-service invoked by the end-user
3: Set web_service_name to 'check_vehicle_details' read from the web services Registry using identifier of Step 2 , i.e. web service '101'2.
4: Set the web_service_url to the web service name: 'http://www..gov.com/webservices/check_vehicle_details?' and value to the output returned by web service 102.
6: Store parameter names and values in the linked list as obtained from the user.
7: Link this node with the previous one using 2 links
8: Build the URL with these values and point to the next node. The URL contains the host name, port, next web service name and parameter values
9: Point node to null.

Node 2 calls the web service *check_veh_details* by using the vehicle parameters that are located in the first node, shown in fig. 8. This check validates the vehicle registration mark against the owner's national identity number. The result of the call to the web service *check_veh_details* is then stored in the current node, i.e. Node 2. This composition and messaging method also has the capability of handling 'if…then' by branching the linked list right, left or bottom to create the next node. Decision boxes appearing in process flows, can be implemented using linked list. Inherently, the branching can be dynamic depending upon the parameter value and the check conditions.

if the parameter value = '1'
then branch to the right node and create a link back to the previous node
else
branch to the bottom node
create a link back to the top node

If the result is 'OK' the next node is created and makes a call to another web service for checking the availability on the requested date. An appropriate error message is recorded in the web service node 'display

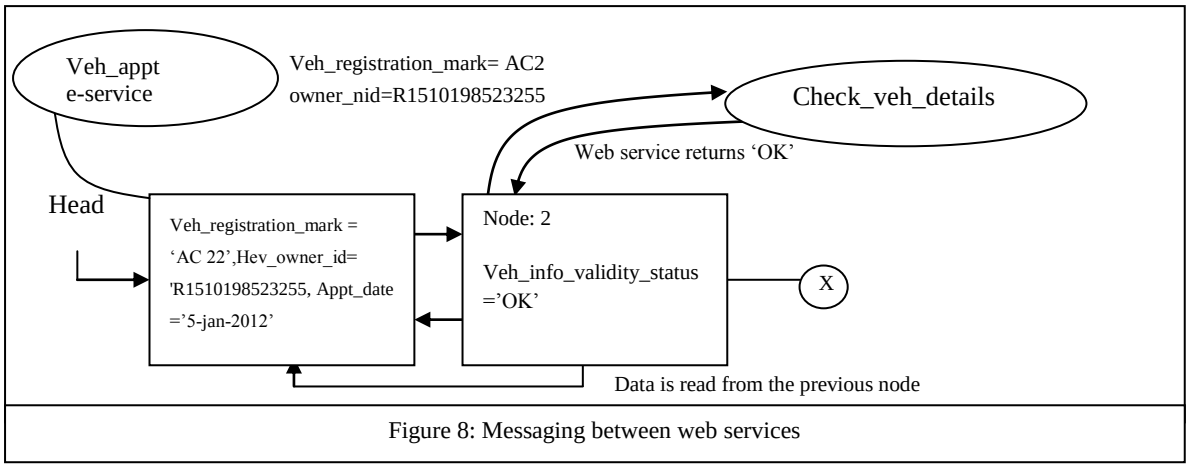

Figure 8: Messaging between web services

message' for all the possible cases as shown in the fig. 9.

Using doubly linked lists, web services can be composed in order to execute business processes and hence e-services. Fig.s 7 and 8 show how the proposed method can be used to address dynamic web service composition and messaging. Using the same approach, the complete business process is constructed as shown in fig. 9.

V. RESULTS

This paper has successfully shown how an e-service can be developed and deployed using the proposed agile web service composition and messaging approach. A typical e-service, appointment for vehicle was used as an example to illustrate the method and its working. The service was broken down into modular business processes and web services. Further, these web services were logically sequenced and messages were passed from one web service to another so that information can be exchanged. Using the proposed method for web service composition and messaging, the appointment_for_veh_ fitness_test e-service was successfully implemented. The screen shot is shown in fig. 10.

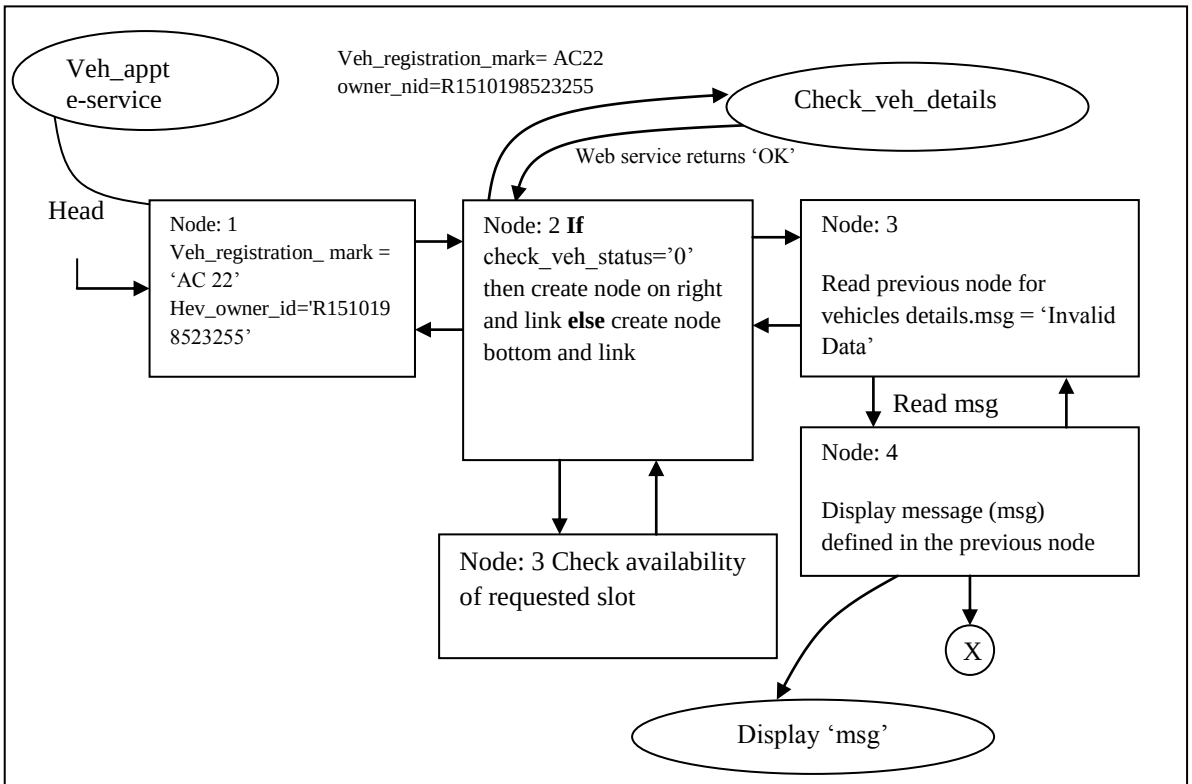

Figure 9: Messaging between web services

VI. CONCLUSION

This work analysed syntactic and semantic web composition approaches and discussed the strengths and weaknesses of available methods. Further, the paper proposed a novel approach for agile web services composition, sequencing and messaging mechanism as part of an overall framework called SBA-eGOV. An example of an e-service has been used to successfully test and illustrate how the proposed method can be used to deploy the e-service. One of the main weaknesses of the existing methods is their inability to dynamically compose web services. The proposed method has the ability to compose web services based on the evolution of parameters as processes are executed, hence dynamically. The example used has successfully shown how existing web services can be dynamically composed.

This work focused on agile composition of web services and did not dwell in details concerning security of messages, an area of further research. For instance, in the proposed approach, data messages are exchanged between web services in form of parameters. These messages are transmitted in clear text, hence inducing a security flaw in the system. The proposed model should therefore be extended so that data messages are encrypted before transmission. The encryption key should not be included in the data message. This mechanism will secure data messages. A second area of further research pertains to access control of web services. Not every user or web service should be allowed to access every other web service. Therefore a central Role-Based-Access-Control (RBAC) specification should be implemented on top of the proposed model for agile web service composition. Implementation of these two security measures will result in a fully implementable agile web service composition method.

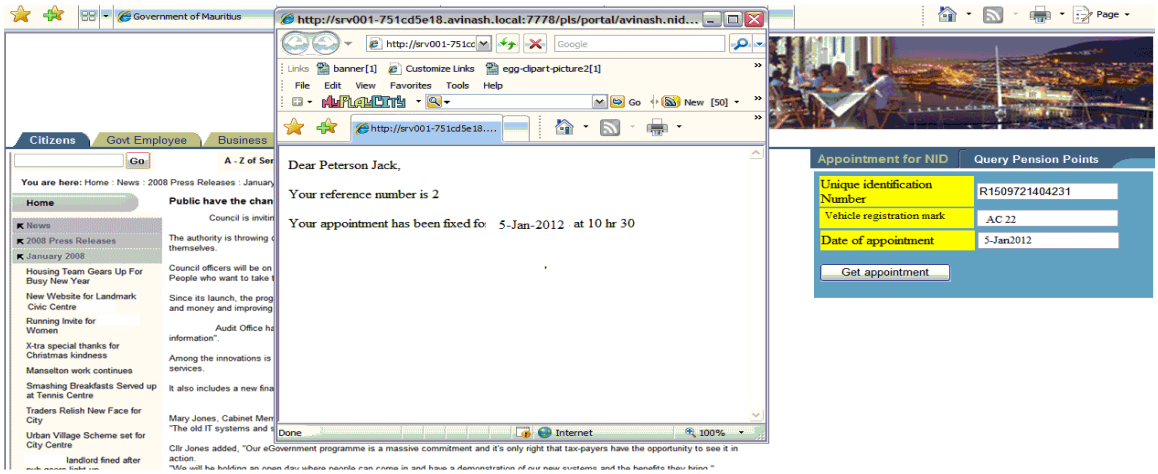

Figure 10: Screenshot showing the results of the web service *appointment_for_veh_ fitness_test*

References

[1] IBM 2007. Business Process Execution Language for Web Services version 1.1 BPEL 1.1. Available: http://ibm.com/developerworks/library/ws-bpel, Accessed Dec 2011

[2] Kavantzas, D. Burdett, and G. Ritzinger 2004. WSCDL v1.0. Available: http://www.w3.org/TR/2004/WD-ws-cdl-10-20040427/

[3] Ankolekar et al., 2007, DAML-S: Web Service Description for the Semantic Web in Proc.

[4] WSMO working group, 2008. Available: http://www.wsmo.org. Accessed Sep 2011

[5] Bartalos et al. 2011. Effective Automatic Dynamic Semantic Web Service Composition. Information Sciences and Technologies Bulletin of the ACM Slovakia, Vol. 3, 2011, No. 1, pp. 61–72.

[6] Dustdar 2008 et al.. M. P.: Services and Service Composition – An Introduction (Services and Service Composition – Eine Einf¨uhrung). IT – Information Technology, Vol. 50, 2008, No. 2, pp. 86–92

[7] Agarwal et al 2005. User Preference Based Automated Selection of Web Service Compositions. In ICSOC Workshop on Dynamic Web Processes, 2005,pp. 1–12.

[8] Zh.—Jiang et al. 2009. Effective Pruning Algorithm for QoS-Aware Service Composition. In Int. Conf. on E-Commerce Technology 2009, IEEE CS 2009, pp. 519–522.

[9] Huang et al, 2009. Effective Pruning

[10] Bellur et al., 2009 H.: On Extending Semantic Matchmaking to Include Preconditions and Effects. In ICWS '08: Proceedings of the 2008 IEEE International Conference on Web Services, Washington, DC (USA), 2008, pp. 120–128.

[11] Kirci Ozorhan et al. 2010. Automated Composition of Web Services With the Abductive Event Calculus. Information Sciences, Vol. 180, 2010, pp. 3589–3613, ISSN 0020-0255.

[12] Urbieta et al. 2008. Analysis of Effects- and Preconditions-Based Service Representation in Ubiquitous Computing Environments. International Conference on Semantic Computing, Los Alamitos, CA (USA), IEEE Computer Society 2008, pp. 378–385

[13] Gamha et al. (2008). A Framework for the Semantic Composition of Web Services Handling User Constraints. In ICWS '08: Proc. of the 2008 IEEE Int. Conf. on Web Services, IEEE CS 2008, pp. 228–237.

[14] Karakoc 2009. Composing Semantic Web Services Under Constraints. Expert Syst. Appl., Vol. 36, 2009, No. 8, pp. 11021–11029.

[15] Lin et al. 2008. Web Service Composition With User Preferences. In European Semantic Web Conference 2008, Vol. 5021 of LNCS, Springer, 2008, pp. 629–643

[16] Avinash Ramtohul, University of Mauritius. Mphil/PhD Transfer report, Service Based Architecture for enabling effective e-Government in Mauritius 2011. Pp 22-24

[17] Maurice ter Beek, Antonio Bucchianrone, Sfefania Gnesi 2007. Web Service Composition Approaches: From Industrial to Formal Methods. Second International Conference on Internet and Web Applications and Services, ICIW 2007, May 2007, Mauritius.

[18] McGuinness and F. van Harmelen. 2009. OWL Web Ontology Language Overview. Available: http://www.w3.org/TR/owl-features/. Accessed on Jan 2011.

[19] Fensel and C. Bussler, 2009. "The Web Service Modeling Framework WSMF," Electr. Commerce Res. Apps., vol. 1, no. 2, pp. 113–137.

[20] Weibel et al. 1998 Dublin Core Metadata for Resource Discovery.IETF 2413. Available: http://www.ietf.org/rfc/rfc2413.txt, Accessed on Oct 2013.