Service Orchestration Algorithm for Web Services: Evaluation and Analysis

Yousra Chtouki[1, 2], Hamid Harroud[1]

[1]Al Akhawayn University in Ifrane
P.O Box 104, Ave Hassan II
Ifrane 53000, Morocco

Mohammed khalidi Idrissi[2] ,Samir Bennani[2]

[2]Ecole Mohammadia d'Ingénieur
RIME Laboratory; LRIE EMI
Ave Ibn Sina, Rabat, Morocco

Abstract

With the growing need for service oriented architecture and the rise of cloud computing, web services have become the focus of many researchers and software developers. Our area of focus is web service composition. In this paper we are presenting an evaluation and analysis of the proposed orchestration algorithm in [1]. The algorithm is intended to orchestrate interactions between different web services. Generate a workflow of a set of web services in order to generate a composed service. We present an approach that aims to optimize the process of web service composition by assigning the best service and service activities. This involves taking into consideration several parameters such as the execution time, cost of communication between services that may reside in different locations, constraints imposed by the user and the quality of service (eg: response time, cost, reliability, availability, etc..).Our work can be implemented using a set of web services standards therefore it can be integrated in different environments. We are using WSDL for web service description, PBEL for business process description and creation, SOAP, messaging protocol, and WS-Policy to describe and create rules to be applicable on web services orchestration.

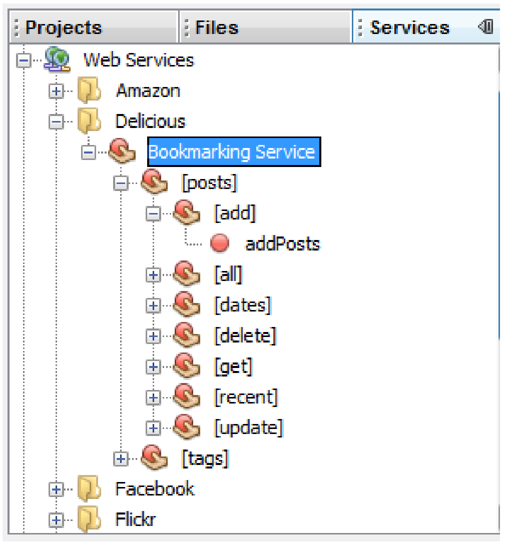
KeyWords

Web Services orchestration, web service composition, WSDL, PBEL, SOAP, WS-Policy.

1 Introduction

Web services are described as any application or part of an application that is available over the web. Web services are self-describing and self-contained network available software modules that perform concrete tasks and are deployed easily because they are based on common industry standards and existing technology such as HTML and HTTP[2]. Most web based services are compatible with all popular browsers. Therefore, the issues of software and hardware compatibility is rare if not absent. All of this has pushed software developer and users to opt for web based

services. The challenge is that most existing services are not suited for every user's needs, most of the time users may have to work on multiple services to complete a single task.
In education particularly we have surveyed a group of university professors to find out the web based applications (web services) that they use in daily basis to complete their teaching related tasks. We have found that more that 80% use at least one web service along with the existing learning management system (LMS) We have also found that most of the web services used by educators are general purpose services, most of them are free web based services, easy to use and tend to be very popular services among a large number of different users not necessarily related to education. According to the center for learning & performance technologies [3] the top 10 eLearning tools for 2012 are twitter, YouTube, Google Docs, Google Search, WordPress, Dropbox, Skype, PowerPoint, Facebook, Wikipedia. Surprisingly Moodle which is a learning management system comes as 11th in the list. This further confirms that educators are using services provided by different external web services besides their existing LMS in order to complete daily teaching related tasks. New Web 2.0 technologies and websites such as a blog, wiki or YouTube make new demands on learning, and they provide new supports to learning [4], In a previous publication we have specifically evaluated the impact of YouTube videos on the student's learning [5]. We have found that the use of YouTube videos as a support material to further illustrate concepts and enforce the students learning is very effective. We have also found that student's interest in the subject has increased. From the student's perspective using such tools as YouTube has become necessary because of the change in the nature of the students and the way they learn. They prefer random "on-demand" access to media; expect to be in constant communication with their friends and ease of access in the creation of their own new media [4]. Therefore, in any given educational environment, there is a need to use external web services to accommodate needs not offered by the existing LMS. From this comes the need for web service composition. This is the best way to benefit from different services by different providers without any major changes to existing LMS or any work environment in general.
Since a given web service offers several functionalities, therefore it contains several sub-services. Service composition allows us to use sub-services of different WS and compose them according to the user's requirements: Figures 1a and 1b show the list of web services that may participate in the composition process; we notice that each service is composed of several basic services. Therefore functionalities of existing web services can be used as a basic service. Delicious for instance contains posts and tags as subservices and each one contains other sub-services as well. The possibility to consume part of the web services relieves some of the overhead during the execution time. It also makes the composed service specific to the user requirements.



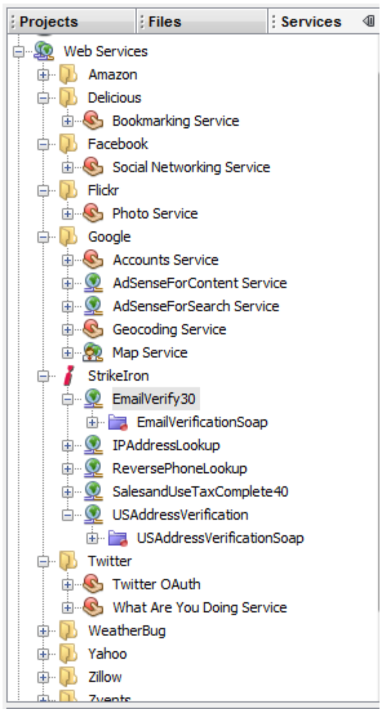


Figure 1a

Figure 1b

2 Motivational Example:

We have established that there is an obvious need to integrate multiple web services in order to accomplish daily eLearning tasks. Based on the list of the most popular eLearning tools we opted to pick some of the most popular web services and generate a motivational example based on our teaching practices and needs. YouTube is increasingly being used by educators as a pedagogic resource for everything from newsworthy events from around the world to “slice-of-life” videos used to teach students within an ESL (English as a Second Language) course. From instructional videos to an online space to share student authored content [4]. Thus, the goal is to use YouTube as a source of videos to serve as a learning content and deliver it through the currently used LMS. This requires several steps, first, search for a video by topic then select one, download the selected video this can be done using RealOne downloader plugin. Next, upload the video on the LMS. To complete these steps we need three service providers: YouTube, RealOne, and the LMS. Obviously, each of the participating services has its own rules, restrictions and limitations that have to be taken in consideration during the composition process. This will ensure that the interactions between these services will be smooth and invisible to the end user.

Figure 2 outlines the video request process, in which the user submits the request and the goal is to make use of all three services LMS, YouTube and RealOne. The user submits the request to search for a video and upload it to the LMS, the request is submitted through the orchestrator plugin that can be plugged to any web application. In this case the request is submitted through the LMS. Since the LMS doesn’t have a video search the search for video is sent to YouTube (the video WS). YouTube executes the video search request, returns a list of matches to the user, the user makes a selection. Then the YouTube service checks for the RealOne Downloader plugin, if it's found the download of the selected video is executed and the file path is saved and returned as output of YouTube WS.

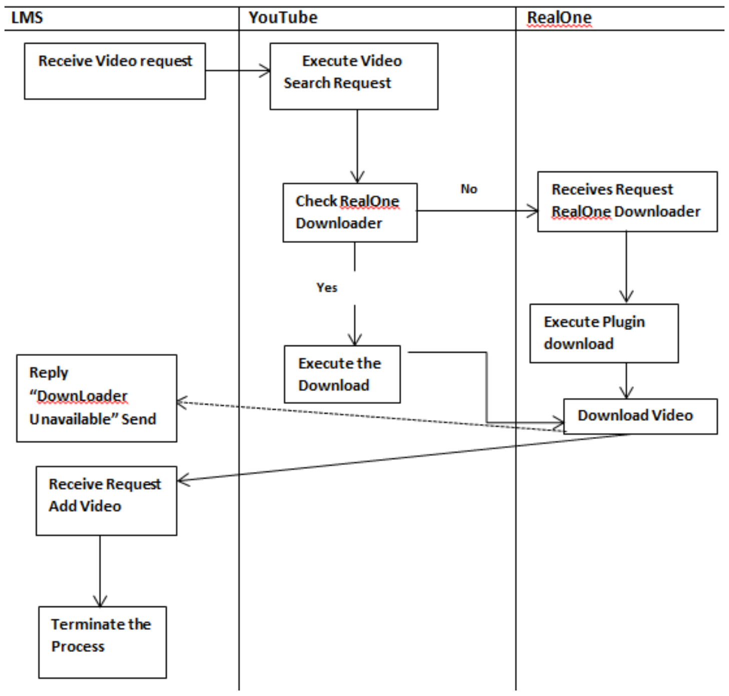


Figure2: Outline of a video request process

3 Service Composition:

Service composition is a very complex process, it includes several levels: The basic services, their description and how they will be integrated and triggered by the system. The orchestration and composition process, which includes finding the right set of services; generate a workflow and determine the interactions between the participating services; finally, the composed service and how it will be presented, stored and reused for future compositions.

The goal of our work is to find an orchestration algorithm that will provide optimum results in terms of managing interactions between the different participating services. Next, ensure that the rules specific to each service are applied whenever that service is involved in a composition process. Then, enforce rules specific to the interactions between the participating services. Since web based applications can be developed using different languages and have different functionalities. One of the key attributes of Internet *standards* is that they focus on protocols and not on implementations. thus standards allow for services to be described in a common language web service description language(WSDL), to use a common messaging protocol to communicate (SOAP) and finally the integration of these web services and the creation of newer ones based on some features of existing ones (PBEL). The standards are a collection of specifications, rules, and guidelines formulated and accepted by the leading market participants and are independent of implementation details. Standards establish a base for commonality and enable wide acceptance through interoperability [6]. We are also using BPMN Business Process Modeling Notation to express the workflow of our scenario. Figure 3 illustrates the workflow diagram of our scenario. There are three participating web services: the LMS, YouTube, and RealOne downloader. The user submits the request for a video through the LMS. The figure3 shows the process to process coordination and communication, the composer application describes how a video request is executed by YouTube WS via the RealOne downloader. Several activities are executed by each of the participating WSs. YouTube WS has to check if RealOne downloader plugin is available (not included in the figure) if so it will submit the request to download it then submit the request to download the video selected by the user through the LMS. The RealOne downloader has to execute the request submitted to it by the YouTube WS. In the remainder of this paper we describe how the orchestrator coordinates the interactions between these different services and explain how it uses integrated policies to apply orchestration rules.

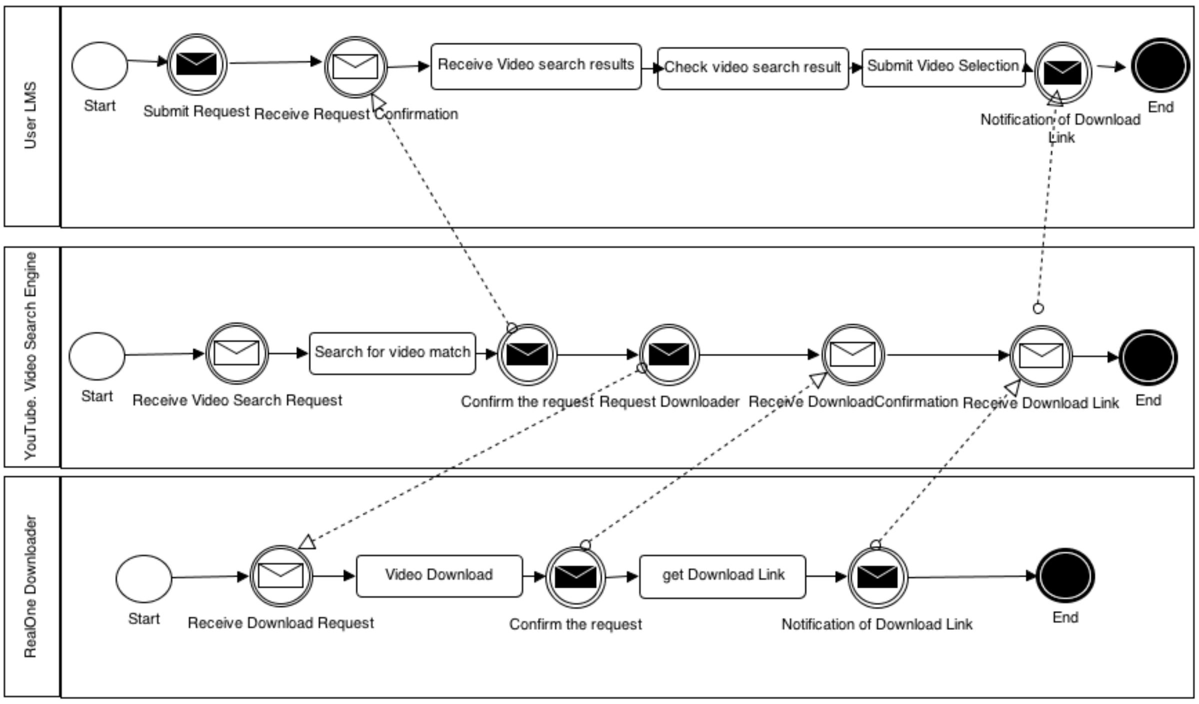


Figure3: YouTube download Request flow diagram

a) Finding a matching service:

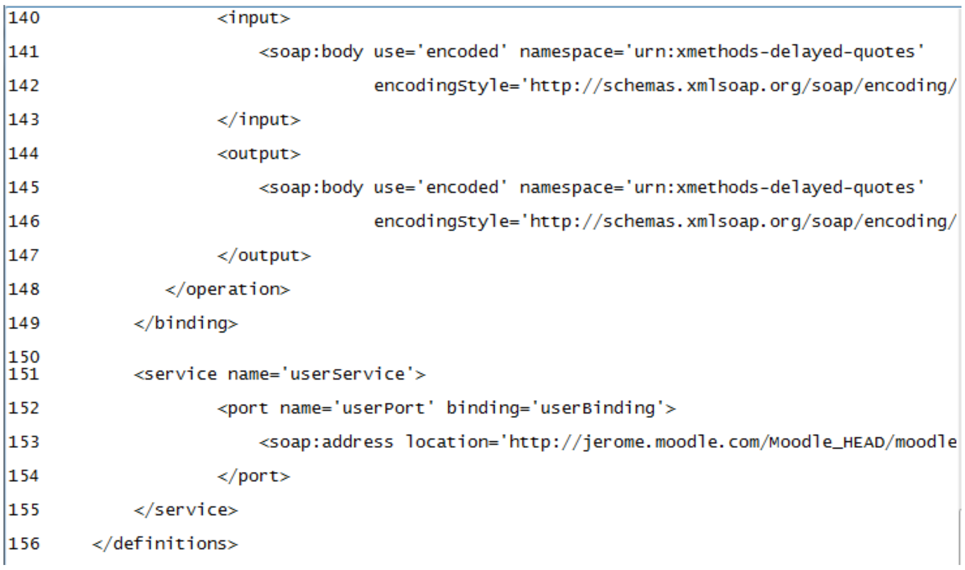
Each participating service in the composition process is described using a WSDL (web service description language). WSDL is based on XML, some of the advantages of XML is that is retains the data structure in transit, it is also very flexible which solves application heterogeneity and middleware problems [6]. The WSDL file contains information about the service, how to access it and its input and output. WSDL is a specification defining how to describe web services in a common XML grammar. WSDL describes four main important pieces of data:

- Address location of the specified service
- Information about the interface to describe all service functions
- Information about the Data type for all message requests and message responses
- Information about binding specifications for the transport protocol used

We are using the service related data stored in the WSDL to search and locate a set of services using the beam stack algorithm. It is guaranteed that a match is found if one exists. The participating services are both language and platform independent. Services by different providers running in different platforms and written in different languages can interact and communicate easily using this specification. We are using the WSDL data to allow for a given service to be invoked by clients and to use all or some of its functionalities.

Figure 4 is a sample of the (LMS) Moodle Wsdl file. Moodle is one of the participating services in the service composition process. As stated previously, each participating service can contain several sub-services. For Youtube as a web service it contains: VideoSearch. unregistered users can watch most videos on the site; registered users have the ability to upload an unlimited number of videos; Flag – ability to indicate a video that has inappropriate content; Title - main title of the video; Tags – keywords specified by the person who has uploaded the video; Channels – relating to groupings of content; Related videos - determined by the title and tags, appear to the right of the video; Subscribe: registered users can subscribe to content feeds for a particular user or users[4]. We use VideoSearch which is a subservice as a participating service in the composition process.

Our Moodle WSDL defines the interface which comprises: the abstract interface description containing the supported operations, the operation parameters and their types; also it contains binding and implementation description containing a binding of the abstract description to a concrete transport protocol, message format, and network address. Moreover the WSDL will also include the policies specific to each service in order to enforce integrity and allow the service to be executed by different consumers without sacrificing security. We will talk about these policies in the next section 'service specific policies'



```
                <input>
                    <soap:body use='encoded' namespace='urn:xmethods-delayed-quotes'
                               encodingStyle='http://schemas.xmlsoap.org/soap/encoding/
                </input>
                <output>
                    <soap:body use='encoded' namespace='urn:xmethods-delayed-quotes'
                               encodingStyle='http://schemas.xmlsoap.org/soap/encoding/
                </output>
            </operation>
        </binding>

        <service name='userService'>
                <port name='userPort' binding='userBinding'>
                    <soap:address location='http://jerome.moodle.com/Moodle_HEAD/moodle
                </port>
        </service>
    </definitions>
```

Figure 4: the moodle wsdl file

i) Service Specific Policies:

One of the challenges in service composition is how to ensure that rules that are supposed to be applied to a specific service will still take place even if it is executed by outside sources. Especially, authentication and security rules. As a solution to this problem we express these rules as policies. These policies are included in the wsdl description of each service. This guarantees efficiency. An example of a service specific rule or policy is a restriction that a gradebook service has which doesn't allow modification of the gradebook values after the end of the term. This ensures that once a given term is completed, the grades in the gradebook are final and may not be modified. The orchestration platform gets the policy directly from the WSDL, given that the policy is defined in the WSDL. In our scenario, the videos that we search for on YouTube can be added to the LMS only for current courses, once the teaching term is finished all current courses are listed as past courses. Therefore the wsdl file for the LMS includes a policy that learning content can only be added for current courses (the status of a course is current)

The policy is directly loaded from the service description file, which is in our classpath, and will apply all policies in runtime, provided that this resides in the application server and doesn't have any conflicting dependencies in the project. Metro is one that allows such specifications. The service specific policies are sent via SOAP messages as a SOAP header.

b) Orchestration Algorithm

Web services seek to create interoperability among information systems. Web service orchestration enables the coordination of activities [7]. Orchestration is one approach to accomplish service composition in which there is a central server that controls and manages all the interactions between the participating services. The orchestration code is located in the central server. Once a user submits a request for a service that doesn't exist, the call to compose a service comes into play. The central server searches for a set of services that match at least part of the user's request. Based on the set of basic services, the central server then must order and manage the execution as well as the interactions between the participating services; next from a set of possible combinations that meet the user's request, the orchestrator must select the best one then execute it so that the user can use it as a single composed service. The execution of the composed service requires retrieval and exchange of data and code with each basic service provider. This execution can be centralized or distributed among multiple servers. There are several existing tools and standards related to service orchestration such as BPEL business process language. Although a BPEL program invokes services distributed over several servers, the orchestration of these services is typically under centralized control [8]. To perform a decentralized orchestration using BPEL, the orchestration code must be divided among multiple servers. When the BPEL program is partitioned we get multiple independent sub-programs. Those subprograms do not need centralized control to interact. There are also partitioning algorithms and programs like Zenflow and the one proposed in [8].

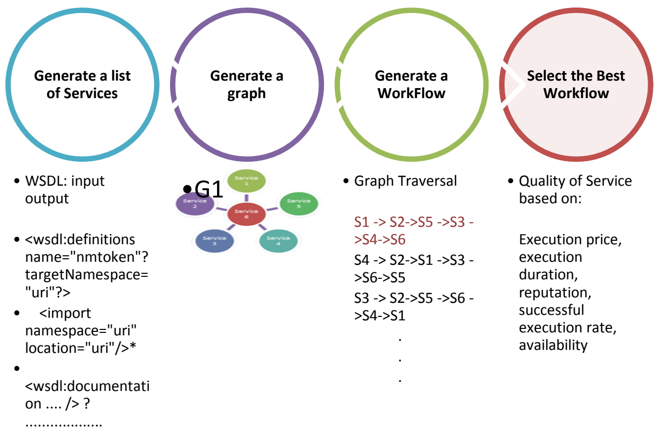
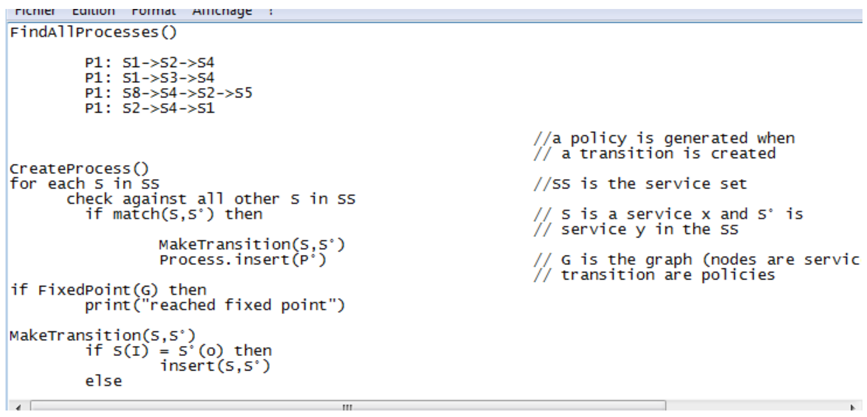

Generate a list of Services
Generate a graph
Generate a WorkFlow
Select the Best Workflow
• WSDL: input output
• <wsdl:definitions name="nmtoken"? targetNamespace= "uri"?>
• <import namespace="uri" location="uri"/>*
• <wsdl:documentation .... /> ?
....
•G1
• Graph Traversal
S1 -> S2->S5 ->S3 ->S4->S6
S4 -> S2->S1 ->S3 ->S6->S5
S3 -> S2->S5 ->S6 ->S4->S1
• Quality of Service based on:
Execution price, execution duration, reputation, successful execution rate, availability

Figure 5: Service Composition Process using the proposed algorithm

Here we will illustrate how we get from step 2 of figure 5 to step 3 of the same figure. How we use the algorithm in figure6 to generate a set of possible workflows based on the results of our service selection (step1 of figure4). S is a set or participating services. G is the graph that contains all the services that match the request. Pn represents a set of processes generated. Each generated process represents a potential solution to the user's request. Each process is represented using a traversal of the graph where the vertexes are the policies that will control the flow of execution between services. In the orchestration process the built in policies within the WSDL of each of the participating services will be used to generate new policies to manage the flow of execution of the composed service. Those policies will be stored as vertexes of the graph.

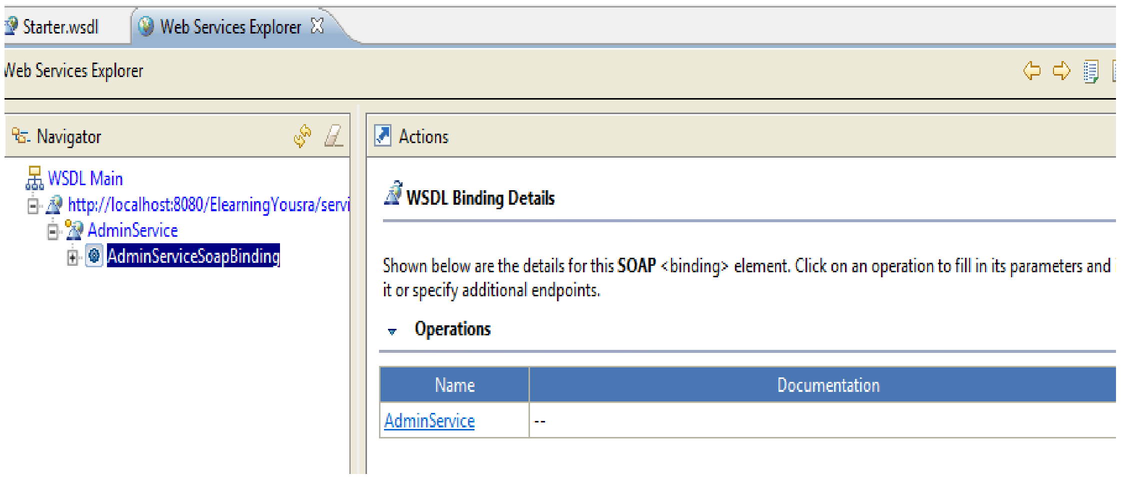


```
FindAllProcesses()

        P1: S1->S2->S4
        P1: S1->S3->S4
        P1: S8->S4->S2->S5
        P1: S2->S4->S1

                                                        //a policy is generated when
                                                        // a transition is created
CreateProcess()
for each s in SS                                        //SS is the service set
      check against all other s in SS
        if match(S,S') then                             // S is a service x and S' is
                                                        // service y in the SS
                  MakeTransition(S,S')
                  Process.insert(P')                    // G is the graph (nodes are servic
                                                        // transition are policies
if FixedPoint(G) then
        print("reached fixed point")

MakeTransition(S,S')
        if S(I) = S'(O) then
                insert(S,S')
        else
```

Figure 6: The orchestration algorithm

i) Communication between Services

Since we are supporting web service composition then communication between the participating services is done using the standard SOAP. SOAP stands for simple object access control. It is a standard to allow different services by different providers to communicate and exchange data. SOAP protocol is intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols [10]. The framework is intended to be independent of any specific programming model and other implementation particular semantics. Figure7a and Figure7b show the soap binding used to create communication links between the web services.



Shown below are the details for this SOAP <binding> element. Click on an operation to fill in its parameters and it or specify additional endpoints.

Name	Documentation
AdminService	--

Figure7a: Example of messaging protocol is used to define binding between the basic services

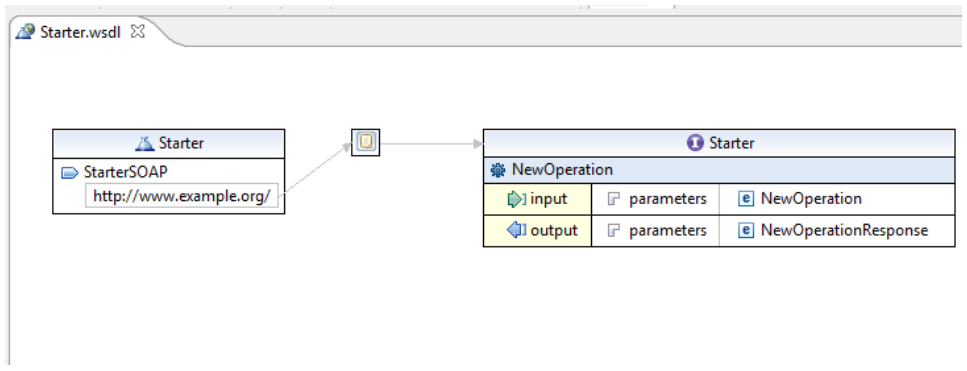


Figure 7b: Example of input/output definition within SOAP operations

ii) Orchestration specific policies:

These are rules and restrictions specific to the management of the orchestration process. Each policy will specify how a given service may be composed with another. Our services are described as nodes and the policies are the vertexes of that graph. Each policy may contain several alternatives.

Alternatives within a policy may differ significantly in terms of the behaviors they indicate. Conversely, alternatives within a policy may be very similar. In either case, the value or suitability of an alternative is generally a function of the semantics of assertions within the alternative [9] the orchestration algorithm allows generating policies that will handle the execution of the composed service. Those policies are derived from policies of basic participating services. Since WS-Policy offers a framework through which we can extend the description features already provided through the WSDL. WS-MetadataExchange protocol allows to dynamically exchange WS-Policy between two participating service endpoints. Using the WS-MetadataExchange protocol, service endpoints can exchange policies at run-time to bootstrap their interaction with information about the settings and protocols that apply [2]

iii) Validation of the Orchestration Algorithm:

The search for all possible matching services to the users request is performed using the beam stack search algorithm with backtracking; Beam search is a widely-used approximate search algorithm. By focusing its search effort on the most promising paths through a search space, beam search can find a solution within practical time and memory limits – even for problems with huge search spaces [11]. However the beamstack algorithm is not considered comprehensive because it won't necessarily find the optimal solution. This means that in order to guarantee that the optimal solution is to be found we opt to use beamstack search with backtracking. If there are no services that match the user's request; the process halts and notifies the user of the result. The orchestration algorithm generates a set of possible workflows based on all possible traversals of the graph. The algorithm is using graph traversal in which the orchestration policies are the vertices of the graph. This traversal will allow generating processes even when we have a set of services that match partially the user's request. In case there is a fault in the execution of one of the services, the system will continue with the next possible process in the list of generated processes (different traversals). The processes are all ordered by priority order based on one criteria (quality of composed service). Quality of composed service (process) is based on how close is the process to match the user's request. We should also mention that the use of policies allows more flexibility dealing with different participating WSs differently. It is customized for each of the WS as needed. With this kind of flexibility, WSs can then be designed to offer differentiated QoS (quality of service); eg: service precision, granularity, timeliness and scope, depending on the end customers. WS can also be differentiated based on technical quality and QoS details such as response time, performance bandwidth used, and reliability [2]

4 Conclusion and future work

In this paper we have presented an evaluation of a service orchestration algorithm for web service composition. Web Service composition has being addresses using different method and techniques one of those methods are rule based techniques. Although our method is rule based as well we are making use of WsPolicy as a web service rule standard. Also our algorithm allows the web service composition to be dynamic to a certain level by generating newer policies that will address the execution of the composed service. This resolves several issues including that only minimal changes are needed to the existing environment. Newer web services may be added dynamically if they are described using the same WS standard used. This will allow the system to expand continually and dynamically. We have tested our proposed algorithm using a simple E-learning scenario which is to search for learning content (a video from YouTube), download it then upload it to the LMS. Our method preserves the restrictions and conditions for execution of each participating web service. For future work a translator can be added to the current system in order to translate the user's request from natural language or any other service user language users choose to input their request in. The translator will then translate it into the service specification language used WSDL and therefore the system will be user-friendly.

References

[1] Y.Chtouki, H.Harroud, M.khalidi, S.Bennani ; 'Policy Based Orchestration in Service Composition' ;IJAEST-Vol-No-11-Issue-No-2; ISSN: 2230-7818; http://www.ijaest.iserp.org/ijaest-archieves-vol11.2.html ; 2011.

[2] M.Papazoglou ; 'Web Services & SOA : Principles and technology', Pearsoned ; 2012 ; Print

[3] The center for Learning & Performance Technologies; 'Top 100 Tools for Learning 2012'; http://c4lpt.co.uk/top-100-tools-2012/

[4] Duffy, P. "Engaging the YouTube Google-Eyed Generation: Strategies for Using Web 2.0 in Teaching and Learning." The Electronic Journal of e-Learning Volume 6 Issue 2, pp 119 - 130, available online at www.ejel.org

[5] Y.Chtouki, H.Harroud, M.Khalidi, S.Bennani; "The impact of YouTube Videos on the Student's Learning"; Proceeding of ITHET 2012;Turkey; 2012.

[6] W.Roshen; 'SOA-Based Enterprise Integration';McGraw-Hill;2009;Print

[7] Janssen, Marijn, Kuk, George,' Comparing Coordination Arrangements Enabled by Web Services and Web Service Orchestration Technology'

[8] M.Nanda, S.Chandra, V.Sarkar; 'Decentralized Execution of Composite Web Services';
OOPSLA '04 Proceedings of the 19th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications; Pages 170-187; USA; 2004.

[9] S.Bajaj et al; 'Web Services Policy 1.2 - Framework (WS-Policy)';W3C Member Submission; April 2006

[10] M. Gudgin, et al, 'SOAP Version 1.2 Part 1: Messaging Framework'; June 2003.

[11] R. Zhou, E. Hansen, Beam-stack search: Integrating backtracking with beam search, in: Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS-05), Monterey, CA, 2005, pp. 90–98.