Service Composition Algorithm: Student Assessment & Course Evaluation

YOUSRA CHTOUKI[1, 2], HAMID HARROUD[2] MOHAMMED KHALIDI IDRISSI[1]
SAMIR BENNANI[1]
[1]Ecole Mohammadia d'Ingénieurs
RIME Laboratory
Ave Ibn Sina, Rabat, Morocco
{s.bennani, m.khalidi}@emi.ac.ma
[2]Al Akhawayn University in Ifrane
P.O Box 104, Ave Hassan II
Ifrane 53000, Morocco
{y.chtouki, h.harroud}@aui.ma

Abstract: - Service composition is a widely addressed subject that has enforced itself on businesses due to the interchanged and mixed interests among companies and institutions. Each business is in need of collaborating with other businesses, this requires the need for using multiple services and applications from different providers. From that the need for service composition has become necessary. Education is a field where service composition can solve many problems; such as providing an educator with the desired services even if they do not exist in the LMS 'learning management system'. A new service can be composed based on existing services even if they reside in different locations and by different providers. In education for each given course a course Portfolio is an important component. For that we have chosen it to establish our proposed solution for service composition. This work defines the basics for several applications in service composition in general and in E-Learning in particular. Future work includes a translator can be added to the current system to translate the user's request from natural language also users could choose be provided with the search results and be able to pick the services he/she wants to compose.

Key-Words: - Web Service composition, policies, e-learning, course portfolio, student assessment, policy based orchestration.

1 Introduction

As Institutions both private and public all use some type of learning management system LMS. Most of them contain similar functionalities: course content, assessment, forum, and a gradebook. After evaluating and doing the state of the art, we realized that all the requirements for a given facility or even a course are varied, so most of the time we find that most existing e-Learning environments are very similar and mostly they reinvent the wheel. The challenge is that educators of different courses have very diverse needs. Therefore, a need for several other tools not provided by the available LMS like video player, file converter, flowchart or a drawing software, as well as tools specific to some subjects like math, music, programming..ect. Thus what we find most educators do is either to use these other tools independently, or do not use them because they don't have the technical background to do that. The idea to solve this problem is to be able to make use of functionalities offered by different applications (services), without necessarily having to rebuild the LMS or require any major changes to the existing LMS. The optimum solution that has been used for several years to achieve that is service composition.

The goal of our research is to use the concept of service composition to combine already existing e-learning tools and therefore create a very powerful E-Learning environment, which continues to grow with the development of new eLearning tools.

Service composition is an excellent approach that has taken a worldwide importance because it increases productivity and reduces costs accrued by application development.

2. Related Work

Here we will list the existing service composition work. There are several approaches depending on the scenario used for the study. In [7] the authors propose a semantic graph based service composition. The problem with such composition is that there is a running time environment to execute the services. Such solution will limit the possible participating services to certain services. In [8] the authors propose a UML based composition of grid service workflow without an actual implementation. In [9] the authors propose a composition with an Ontology matching, composition at type level with service matching. And the deployment is in a decentralized workflow orchestration infrastructure. In [10] the authors used the OSIRIS platform to handle a P2P (peer to peer) composition. In [11] the authors propose an enhancement to existing business process language in order to solve some of the current limitation. We can also say that BPEL (business process language) is a commonly used standard language for service composition; it is widely used by companies to build service composition solutions that provide quick solution but not the best ones. And most importantly, those solutions are not dynamic and require continuous maintenance and development. For that we opted to look for a solution that is suitable for business as well as academia, because it addresses the issues of both parties. Our solution is practical because it relies on using existing standards (WS-policy, wsdl files, web services…ect) therefore it offers interoperability. Also it offers a contribution in the field of algorithm solution for orchestration of service composition.

3. Service Composition

Service composition is a multiple step process. It tackles handling interactions between different services and providing their different functionalities to the user as one service. The different services can be combined with other homogeneous or heterogeneous services to form complex web applications [13]. The first step is to search for matching services that will be used in the composition, then those matching services can be composed into a set of possible solutions, lastly the best solution is chosen based on one or more criteria. The composition itself can be done either using choreography (in which each service handles its interactions with the group) or orchestration (in which there is an orchestrator that handles interactions among all participating services. Our solution is an orchestration algorithm to handle interactions between the services using polices.

3.1 Service Composition Framework:

After the initial study of the situation of e-Learning management systems, we decided to break the project into several important steps from the definition of the subject to the creation of the architectural plan of the proposed solution. Fig1 shows the framework for web service composition using WS standards and WS policies. The web provides the platform for the user to submit his/her request and receives the composed service, also it serves as a continuously growing registry of the basic services that can be searched and used to create new WS. Policies are used at different layers of the process, there are service specific policies like engagement, request, and deployment policies and there are orchestration specific polices. These later ones are generated during the orchestration process by the policy generator. The other components will be further explained in this paper.

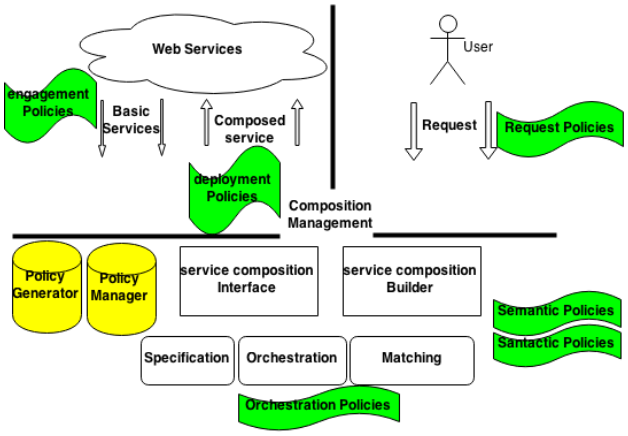

Fig1: The proposed service composition framework

Our main contribution consists of two important elements: i- The use of the composition service for creating new e-learning services. j- Study the different methods of implementing the composition of service: service composition can be static or dynamic using a model based approach or the use of different artificial intelligence techniques to generate dynamic service composition. In Fig2 we present the flowchart of the e-learning provisioning system that intends to receive a request from the user, and then search for possible solutions that can be composed using existing matching services. The

system allows recomposing other combinations of services until a satisfying solution is found.

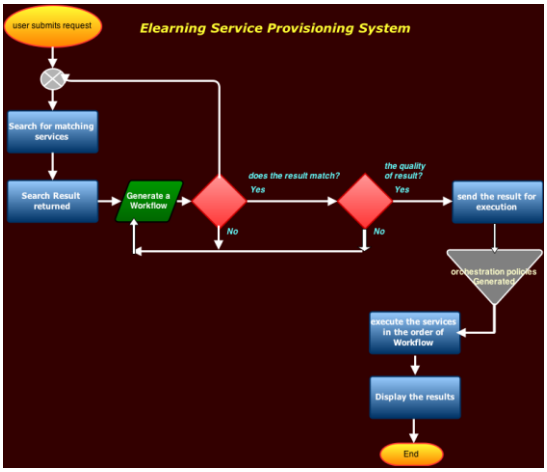

Fig2: e-learning web service provisioning

3.2 Searching for Matching Services:

After extensive research and testing we found that the approach of integrating systematic backtracking with beam search is a rewarding After extensive research and testing we found that the approach of integrating systematic backtracking with beam search is a rewarding solution to our search problem. In [12] the authors showed how to combine this approach with a memory-saving technique that uses divide-and- conquer solution reconstruction. Although beam- stack search itself is not difficult to implement, divide-and-conquer beam-stack search presents more of a challenge, but when implemented, it provides a very effective search algorithm [12] Based on the theorem Beam-stack search is guaranteed to find an optimal solution if one exists. Along with the backtracking technique the algorithm can backtrack and start from a new path. So the path to the best solution is never missed even if the wrong path is taken initially during the search. The trace-back method of tracing pointers backwards from the goal node to the start node, search algorithms that use this memory-saving technique rely on a divide-and- conquer technique of solution recovery [12]

3.3 Graph of the matching services

We have talked in previous publications about the search for matching services to the user's request. We found that the beam stack algorithm with backtracking is what will provide us with the best solution. A matching service can partially or totally meet the user's request. Most of the time we assume partial match. Since if a user were to find his/her request in one service then there will not be a need for service composition. Once the search provides us with a resulting set of services we use them to create a graph composed of multiple nodes, each node represents a service. Fig3 shows the algorithm used to create a graph composed of a set of matching services. P stands for a process, S for a service, SS is a service set, G is the Graph, and a transition is a policy. A process is created by traversing the services at different orders. Since we are using policies to manage execution constraints and rules for each service and among services; then a policy is generated when a transition is created.

```
FindAllProcesses()

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


CreateProcess()
for each S in SS
      check against all other S in SS
        if match(S,S') then

                MakeTransition(S,S')
                Process.insert(P')

if FixedPoint(G) then
        print("reached fixed point")

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

Fig 3: the Creation of the Graph Algorithm

The next step then is to orchestrate tasks between these services, and order the execution between them. This is done by performing different traversals of the graph in order to generate a set of workflows. Fig4 shows the process of the service composition steps that we are proposing solutions to in this paper.

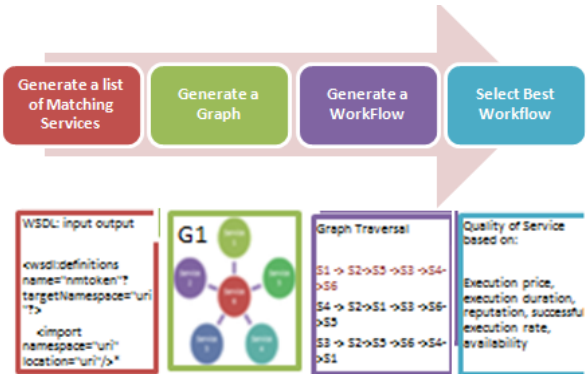

Fig4: The composition process for web services

3.4 The Traversal of the Graph

Below in fig 5, we present the program code used to allow the traversal of the nodes. Each node represents a web service; the vertexes represent the policies that link the services if the condition is met. Based on the user's input; the program traverses the nodes and generates a set of workflows. Fig 4 shows the creation of the graph module. The graph is created based on the existing matching services. So based on the users request a set of services based on the results of the search. The user specifies the number of matching services he/she wants. Each service represents one or more components needed by the user. The traversal of the graph will allow the composition of a new service based on existing services. Generate a graph where each service is a node in that graph and the connection between the nodes are transitions.

```
create_graph()
{
int i,max_edges,origin,destin;
printf("\tCreating a graph of solutions \n");
printf("Enter number of solutions : ");
scanf("%d",&n);
max_edges=n*(n-1);
for(i=1;i<=max_edges;i++)
{
printf("Enter edge %d( 0 0 to quit ) : ",i);
scanf("%d %d",&origin,&destin);
if((origin==0) && (destin==0)) break;
if( origin > n || destin > n || origin<=0 || destin<=0)
{ printf("Invalid edge!\n");i--; }
else {adj[origin][destin]=1;}
}/*End of for*/
}/*End of create_graph()*/
display()
{
```

Fig 5: The creation of the Graph based on matching services

The traversal of the graph will generate a set of workflows. The selection of the best workflow using quality of service properties, which are: Execution price, execution duration, reputation, successful execution rate, availability. The number of services is chosen by the user.

3.5 The execution of the orchestration

The next step is to compare two different methods of execution of distributed services Centralized verses Decentralized Execution of Service Orchestration in order to decide whether to opt for a centralized or decentralized execution of the composition process, we did a comparative study and found that decentralized execution is more efficient and has more advantages.

3.5.1 Centralized Execution:

In the fig6a below web service providers, and the orchestrator requests services from them. The orchestrator is responsible for generating a workflow process and executing the composed service as a single service. The user sends a request to the central system which is then responsible for communication, interaction and execution of data and information gathered from different service providers.

Fig6a: centralized execution of the service orchestration

3.5.2 Decentralized Execution:

The concept of decentralized execution is a natural step in the evolution of the fusion of distributed systems, cloud computing, service-oriented architectures and the growth of web services and standards, and mobile environments. The main advantage of decentralized execution is that it is easier to build and relatively requires little time to implement. It also allows automating the current way of working without any major modifications to the existing framework. All of these have pushed the need for a partially or completely decentralized computing. Execution is a major step in the composition of services because it involves services residing in different locations. Naturally, it would be easy to think that having each service executed by the supplier will ensure its proper implementation as

well as reduce the cost of communication and data transfer between nodes (service providers). One way of achieving decentralized execution is to partition the execution code between multiple servers or all participating nodes. In Fig6b, each system has a service provider; it has its own control manager managing the needs for execution of its services. Control Manager contains policies and specific aspects to the performance of services provided by the service provider. Therefore, a given service is executed by the service provider in which it resides. One or more service providers may receive a request directly from the user then the control manager communicates with other service providers to complete the request of the user. The data and control dependencies between components can be analyzed and the code can be divided into smaller components that run in distributed locations.

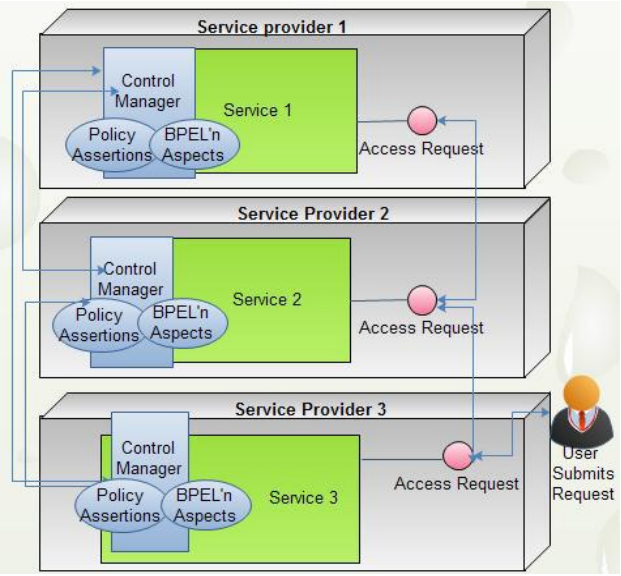

Fig6b: centralized execution of the service orchestration

4. Polices based orchestration:

We opted for decentralized execution of the service composition orchestration. This choice is driven by all the benefits listed in the previous section. However, as we have said, many questions arise in decentralized execution; such as security and privacy, among others outlined in the previous sections. Our goal is to solve these problems using policies.

Policies have been widely used in software development and especially in web services. There are several languages and existing standards to support the creation and implementation of policies. Since our work is focused on web services, our aim is to rely on using standards so that we can provide interoperability and not have a compatibility problem. WS-Policy is the standard language for writing policies for web services. A composition policy is a set of rules on a set of authentication communication and security policies can be made to obtain a set of composite services. A policy declaratively specifies the behavior of these composite services that are automatically generated. Polices used in service composition are adaptive as well as allow the benefit of maximum optimization. One challenge is to ensure that the error handlers keep their semantics, regardless of the node (server) that runs it. If the error handler includes sending a message to another component in the compound web service (which runs on a different node because of decentralization) then changes must be made accordingly.

Policies are selection rules in which we can specify what action should occur when one or more conditions exist. WS-Policy is a well-known standard. It defines a basic set of structures that can be used and extended by other Web services to express and publish about their own policies. WS-Policy, defines a set of statements, each specifying different requirements of authentication protocol and messaging and privacy policies, and QoS features. After outlining some of the problems caused by the decentralized execution of the orchestration, we now see how the policies can provide a solution to control the execution flow of each Web service involved in the process of service composition, so they offer a very flexible environment to address all types of restrictions to ensure total security and resolve the error management problem. Business BPEL language provides BPEL'n'Aspects which is a non -intrusive mechanism for adapting the BPEL process control flow. It is based on Web service standards to process activities created as aspects of the BPEL processes.

BPEL'n'Aspects is based on existing technologies and standards. WS-Policy is used to specify aspects. Fig7 shows the structure of aspects such as WS-Policy Assertion. Association of aspects of BPEL process is done with WSPolicyAttachment. This mechanism allows setting aspects of the processes during the execution without changing the definition of the target process.

Aspects also allow to specify where, when and in what part of the code in which we must execute a specific instruction. Pointcut specifies the methods in the code that we want to insert an instruction in the part before (), we specify when to perform, in

between braces just after before (), we specify the instruction we want to add. For example, it could be an authentication request if the user is not present in the list of a given organization staff.

```
public aspect AddUser{
pointcut CreateMethods(): call(public * *.bar(..)) ;
before() : CreateMethods()
{
System.out.println("adding an instruction" );
}
}
```

Fig7: Example of an Aspect in BPELnAspects

The integration of aspects in the implementation of the basic functionality is called Weaving. In AOP static approaches, such as in AspectJ during the compile-time/load-time the pointcuts are mapped to locations in the program code whose execution could give a junction point during execution. These are defined to add calls to inform and also finally dynamic controls that the places identified in the code actually give a junction point during execution. To allow more of a dynamic interaction between services and less interference with the user, with aspect-oriented programming languages, aspects can be deployed during the execution of the application, whose behavior can be adapted dynamically.

4.1 The proposed orchestration Algorithm:

Here, we present the algorithm to perform the orchestration between the web services that will participate in the composition. We use fig8 algorithm to generate all the possible workflows based on the results of our service selection (step 1 in Fig4). S is a set of service participants. G is the graph that contains all the services that match the request. Pn represents a set of generated processes. The generated process is a potential solution to the user request. Each process is represented using a graph traversal where the vertices are the policies that will control the execution flow between services. In the orchestration process the integrated policies in the WSDL of each participant services are used to generate new policies to manage the execution flow of the compound service. These policies are saved as vertices of the graph G.

```
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
```

Fig8: orchestration algorithm

4.2 The feasibility of the solution

The next step is to validate the feasibility of using the proposed solution in real scenarios. We wanted to find the impact of YouTube videos on the students' learning. We have assessed the need for e-learning and found a potential use of the proposed algorithm to compose services. We found that students lack interest in the subjects taught. Therefore, we evaluated the impact of the use of illustrative videos from an external web service (YouTube). We found that there was an increase in the student's interest. For this we have selected the use and integration of YouTube videos with existing LMS.

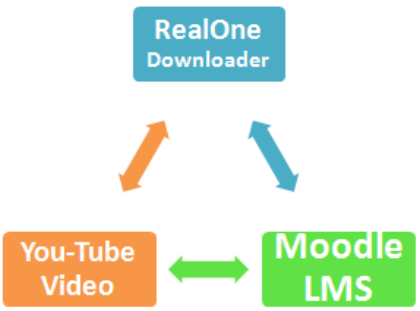

Fig9: The composed service based on 3 WSs

For a successful learning experience it is necessary that students be interested in the subject and that knowledge is presented intuitively. Due to the decreased interest in reading, educators must use other methods to ensure that learning is ongoing. We have presented the results of a study on an introductory computer course. Since the goal of every educational institution is to increase productivity and the level of student's success. Therefore, the study presented one of the many existing web solutions that can enhance the student learning experience fig9.

Since YouTube videos were used for only two groups out of a total number of 150 students. We also want to explore the possibility of automatic

search and YouTube videos filter and its integration into the existing system of Learning Management. Online services for storing, sharing and automatic categorization of videos is important for indexing and searching. Research in this area opens the door to a broader set of opportunities and potential for the future in terms of the use of technology in education.

Fig10 shows the application process a video with a specific educational scenario, in which the user submits the request and the goal is to make use of three web services: LMS, YouTube and RealOne. The user submits the request to search a video, download and add it to the LMS; the application is submitted by the orchestrator plugin that can be connected to any web application. In this case, the application is submitted by the LMS. Since LMS has no video search this research is sent to be executed by YouTube (SW video). YouTube Video executes search query returns a list of results to the user, the user makes a selection. Then YouTube service controls the RealOne Downloader plugin, if you are downloading the selected video then a file path is saved and delivered as output from YouTube web service.

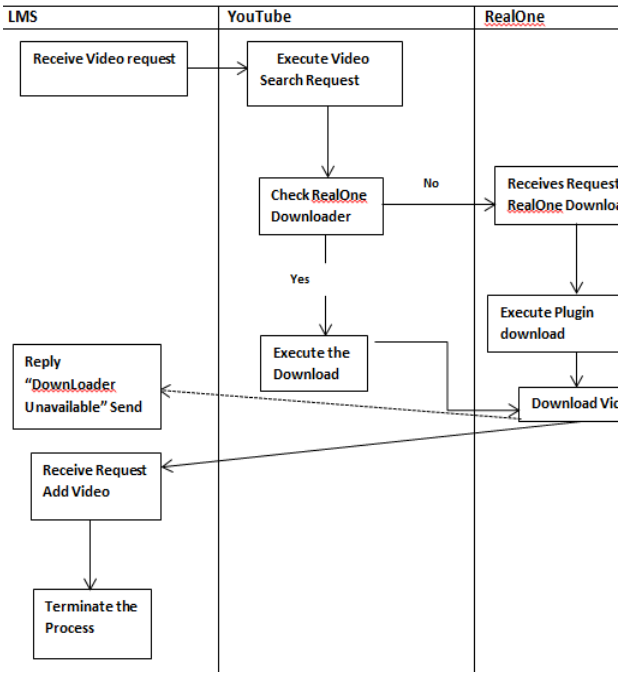

Fig10: Outline of a video request process

4.3 The testing and validation of the results

The last step is to analyze and evaluate the proposed algorithm for composing web services. In this step, we present an assessment and analysis of the orchestration algorithm proposed. The algorithm is designed to orchestrate interactions between Web services, in order to generate a workflow from a set of WSs in order to create a compound service. We present an approach that aims at optimizing the Web service composition process by selecting a close to best solution. This is to take into account several parameters such as the execution time, the cost of communication between services that can reside in different places, the constraints imposed by the user and service quality (eg response time, cost, reliability, availability, etc.). Our work can be implemented using a set of web service standards therefore; it can be incorporated in different environments. We use WSDL to describe Web services, BPEL for describing business processes and creation, SOAP messaging protocol, and WS-Policy to describe and create rules to be applicable to the web service orchestration. Like we have stated before, WS-policy is used to describe both service specific policies and orchestration specific policies. The service specific policies are part of the WSDL file of each web service. While the orchestration specific policies are generated during the orchestration process and executed at execution time.

The search for all possible services corresponding to the user request is performed using the search algorithm of the beamstack search; Beam search is an approximate search algorithm widely used. By focusing the search efforts on the most promising ways through a research space, the search can find a solution in relatively limited time and limited cost and memory use - even for problems with huge search spaces. However Beamstack the algorithm is not considered complete because it will not necessarily find the optimal solution even if it exists. This usually happens if the wrong path is taken during the search so the algorithm doesn't have a way to start again with a different path which means that although a solution exists it may not be found. This means that in order to ensure that the optimal solution is found, we opt to use the beamstack search with backtracking.so that the algorithm can always backtrack and take new search path to find a solution. Thus, if the algorithm returns no solutions it's because one doesn't exist. If there are no services to meet the request of the user, the process stops and informs the user of the result. The orchestration algorithm generates a set of possible workflows based on all possible crossings of the graph. The algorithm uses path graph whose orchestration policies are the vertices of the graph.

This traversal will enable the generation of a process, even if we have a set of services that partially meet the user's request. In case there are errors in the execution of any of the services, the system can continue with the next process in the list of generated processes (different traversal). The processes are sorted in the order of priority based on a criterion (composed QoS). QoS of the composed service (process) is based on how closely it corresponds to the user's request. It should also be mentioned that the use of policies allows greater flexibility in responding to different participating services and providers differently. It is suited for each of the WS as its needs require. With this kind of flexibility, WSs can then be chosen to provide different quality of Service (QoS), such as: service accuracy, granularity, speed and range, depending on the end customers. WS can also be differentiated based on the technical quality and QoS information such as response time, bandwidth usage performance and reliability.

5. Composed Service: Course Portfolio

In fig11 below we show the workflow for the CoursePortfolio process. As we have mentioned the educator for any given course uses the existing LMS along with other web services (WS). The textbook publisher's website is used for different forms of assessment such as exercises and quizzes. Such content can be loaded into the coursework section of the LMS so that they can be graded and tracked on dues dates by the teacher. Then Youtube is used for lecture videos and illustration videos that can also be loaded into the LMS to create the course content and handouts. These videos can also be shared on Facebook which is used for discussions to share illustrative videos, articles, and images and to allow students participation outside the classroom. GoogleDoc is used to create documents such as the syllabus and the lectures slides. These are loaded into the LMS in the syllabus and course handouts section. When we need to create a CoursePortfolio which is composed of the syllabus, the course content (slides, videos, articles..), the assessment, the grade book..ect all of this content doesn't necessarily in the LMS but thanks to web service composition, each of these services has a wsdl file that can be used to request it for execution, send input and get output from it. Since the CoursePortfolio is needed as a pdf file, we chose to use an online PDFconverted a WS to convert the content retrieved from the LMS and get it as a final output as a PDF file the CoursePortfolio.

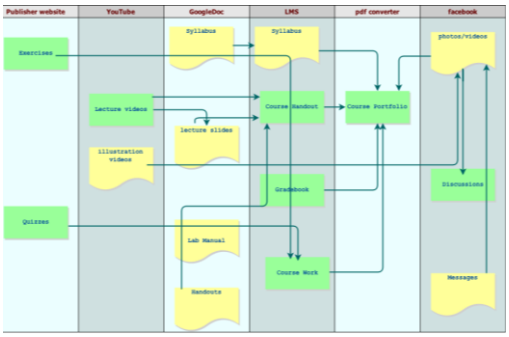

Fig 11: CoursePortfolio Request flow diagram

5.1 Student Assessment:

For every course there are a set of ILOs. The success of an institution is evaluated by how well the students met those ILOs. Every educator illustrates his/her achievement by providing evidence that at least 70% of the students met the ILOs 'the ABET criteria for giving accreditation to institutions'. The problem is that the students' assessment on ILOs is usually diverse and cannot be easily collected and computed to an accurate number. Because a given assessment like a quiz may cover multiple questions that cover fully or partially multiple ILOs. Thus, the personal evaluation of the educator remains the only method of evaluating how much the student met the ILOs. In our institution this assessment is done manually by the professor. The accuracy of such method is questionable. Since this method is very inaccurate we decided that this should be computed automatically. For that we have developed a program to automatically generate this evaluation. The student assessment is intended to evaluate whether the student has achieved the ILOs.

It is intended to assess a student in two different courses with several ILOs each.

Figure 12, 13 and 14 show the three main components of the student's assessment. The student component is handling the student information. Because this program will compute the individual student's achievements and that can be used for all the students taking a given class. Therefore, we can have an evaluation of the course by collecting the assessment of all the students. The course component handles the ILOs of a given course. This can be used for different courses with different ILOs. Last is the evidence component which handles the different evidences linked to each ILO. This structure allows the educator to link different ILOs with different ways of assessments

and still be able to collect evidences for success for each student.

```
le_s* fill_student()

      node_s* stdptr;
      node_c *crsptr1,*crsptr2;
      stdptr=(node_s*)malloc(sizeof(node_s));
      crsptr1=(node_c*)malloc(sizeof(node_c));
      crsptr2=(node_c*)malloc(sizeof(node_c));
      printf("student name:");gets(stdptr->s_name);
      printf("student ID:");scanf("%d", &stdptr->ID);
      fprintf(fptr,"student name:%s ID:%d\n",stdptr->s_name,stdptr->ID);
      printf("course1 name:");scanf("%s",crsptr1->c_name);
      fprintf(fptr,"course1 name:%s\n",crsptr1->c_name);
      stdptr->sc1=crsptr1;
      fill_course(crsptr1);
      printf("\n\ncourse2 name:");scanf("%s",crsptr1->c_name);
      fprintf(fptr,"course2 name:%s\n",crsptr2->c_name);
      stdptr->sc2=crsptr2;
      fill_course(crsptr2);
```

Fig 12: The Student Component

```
i fill_course(node_c* crsptr)

      node_o *obj1,*obj2,*obj3,*obj4;
      obj1=(node_o*)malloc(sizeof(node_o));
      obj2=(node_o*)malloc(sizeof(node_o));
      obj3=(node_o*)malloc(sizeof(node_o));
      obj4=(node_o*)malloc(sizeof(node_o));
      printf("object1 name:");
      getchar();gets(obj1->o_name);crsptr->co1=obj1;
      fill_evid(obj1);
      fprintf(fptr,"obj1 met at %d%%\n\n",obj1->met);
      printf("object2 name:");gets(obj2->o_name);
      crsptr->co2=obj2;
      fill_evid(obj2);
      fprintf(fptr,"obj2 met at %d%%\n\n",obj2->met);
      printf("object3 name:");gets(obj3->o_name);
      crsptr->co3=obj3;
      fill_evid(obj3);
      fprintf(fptr,"obj3 met at %d%%\n\n",obj3->met);
      printf("object4 name:");gets(obj4->o_name);
      crsptr->co4=obj4;
      fill_evid(obj4);
      fprintf(fptr,"obj4 met at %d%%\n\n",obj4->met);
```

Fig13: The Course Component

```
i fill_evid(node_o* obj)

      node_e *ev1,*ev2,*ev3;
      ev1=(node_e*)malloc(sizeof(node_e));
      ev2=(node_e*)malloc(sizeof(node_e));
      ev3=(node_e*)malloc(sizeof(node_e));
      printf("evidence1 name:");gets(ev1->e_name);
      printf("Earned grade:");scanf("%lf",&ev1->ern_pt);
      printf("Max points:");scanf("%lf",&ev1->max_pt);
 obj->oe1=ev1;
      printf("evidence2 name:");getchar();gets(ev2->e_name);
      printf("Earned grade:");scanf("%lf",&ev2->ern_pt);
      printf("Max points:");scanf("%lf",&ev2->max_pt);
      obj->oe2=ev2;
      printf("evidence3 name:");getchar();gets(ev3->e_name);
      printf("Earned grade:");scanf("%lf",&ev3->ern_pt);
      printf("Max points:");scanf("%lf",&ev3->max_pt);
      obj->oe3=ev3;getchar();
      obj->met=((ev1->ern_pt/ev1->max_pt)+(ev2->ern_pt/ev2->max_pt)+
                        (ev3->ern_pt/ev3->max_pt))*100/3;
      printf("objective:%s met by %d%%\n",obj->o_name,obj->met);
```

Fig 14: The evidence Component

The assessment program was written in C and doesn't exist as a web service yet, it wasn't included in the course portfolio process in fig11. We chose to talk about it in this paper because it's an important component of the course portfolio. Once implemented, the assessment program can easily be added as a participating WS to be part of new composed services. ABET the accrediting agency for academic programs in applied science, computing, engineering, and engineering technology requires an assessment of the course to be provided with each course portfolio.

6. Conclusion

We presented a summary of the work that was done over more than 6 years of research in composing web services. The solution is intended to allow the user to use multiple services and create a new service based on functionalities of the existing services. This process handles web services only and uses policies to allow interactions between different WSs that are provided by different providers. Each provider may have his own restrictions in order to allow a given service to be executed such as security checks. The use of policies allows preserving the integrity of each service while it is interacting with other services. Our initial proposition included a complete framework that handles everything from receiving the user's request through a suitable interface and a possibility to allow the user to pick a set of services to compose; all the way to the presentation of the composed services. However after doing extensive research and due to the limitation of resources we found that the main challenge in service composition is to find a good combination of suitable languages, tools..etc. to perform the composition process. We had to make few changes to our initial plans. One is that we planned on handling both WS and non WS, then we decided to focus on WSs only because the world is going towards more standardization, compatibility, web-based tools, cloud..ect. Therefore, standalone applications are fading away. Second, we were planning to implement and test a complete platform with all the components proposed in the first figure, however we found that it's best to focus our work in the area that is in need of new contributions which is the search and the orchestration algorithm. Web service composition was discussed using different methods and different techniques some of these methods are based on rules. Our method is based on the rules and we use WS-Policy as standard Web service rule. In addition, our algorithm allows the composition of Web services in a dynamic way at some level by generating new policies that will address the performance of the composed service. This solves several problems, including the fact that only minimal changes are required to the existing environment. More WSs can continue to be added in a dynamic way if they are described using the same standard used by WS, if not then a WSDL file can easily be generated manually. This will allow the system to expand continuously and dynamically. We tested our proposed algorithm using an e-learning simple scenario. This scenario involves the search of learning content (YouTube Video), download it then integrate it into the LMS. Our

method preserves the restrictions and conditions of implementation of each participating Web service. For future work, a translator can be added to the current system to translate the user's request from natural language also users could choose be provided with the search results and be able to pick the services he/she wants to compose

References:

[1] Y.Chtouki, H.Harroud, M.Elkhalidi, S.Bennani, '*A Service Composition Framework for Providing e-Learning Services'* ; Proceceedings of the ITHET2010 Conf, IEEE explore, 2010.
[2] Y.Chtouki, H.Harroud, M.khalidi, S.Bennani, "*Decentralized verses Centralized Execution of Service Orchestration*" Proceeding ISTEC2011, Turkey. 2011
[3]. Y.Chtouki, H.Harroud, M.khalidi, S.Bennani, *'The Impact of YouTube Videos on the Students's Learning'*,Proceedings of the ITHET2012 Conference, 2012, Turkey. IEEE explore
[4]. Y.Chtouki *'Service Orchestration for Web Services '*, Proceedings of JDRIME 2012, Rabat Morocco ,2012
[5] Y.Chtouki, H.Harroud, M.khalidi, S.Bennani, '*Service Orchestration Algorithm for Web Services: Evaluation and Analysis'* (ijaest) international journal of advanced engineering sciences and technologies, Vol No. 11, Issue No. 2, 332 – 341, 2011.
[6] Y.Chtouki, H.Harroud, M.Khalidi, S.Bennani; "*Policy Based Orchestration in Service Composition"*; International Journal of Computer Science issues; IJCSI Volume 10, Issue 5, Sep 2013
[7] K.Fujii and T.Suda, 'Dynamic Service Composition Using Semantic Information', ACM, 2004.
[8] The European Commission for Higher Education http://ec.europa.eu/
[9] V.Agarwal, K.Dasgupta, N.Karnik and A.Kumar, '*A Service Creation Environment Based on End to End Composition of Web Services'*, 2005.
[10] T.Moller and H.Schuldt,'A Platform to Support Decentralized & Dynamically Distributed P2P Composite OWL-S Serv Execution', ACM, Switzerland, 2007.
[11] '*Modeling Service Orchestration with a Rule-enhanced Business Process Language',* M.Milanović,D.Gasević,G.Wagner,V.Devedzié 2009.
[12] R.Zhou and al, *'Beam-Stack Search: Integrating Backtracking with Beam Search'*, 15th International Conference on Automated Planning and Scheduling, Monterey, CA, 2005.
[13] Sandeep Kumar, Nikos E. Mastorakis , *'Novel Models for Multi-Agent Negotiation based Semantic Web Service Composition'*, WSEAS TRANSACTIONS on COMPUTERS, ISSN: 1109-2750, Issue 4, Volume 9, April 2010.