Research Journal of Applied Sciences, Engineering and Technology 7(22): 4723-4735, 2014
DOI:10.19026/rjaset.7.858
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: January 10, 2014 Accepted: January 25, 2014 Published: June 10, 2014

Research Article

Ontology Based Dynamic e-Learning Flow Composition of Learning Web Services

M. Farida Begam and Gopinath Ganapathy
School of Computer Science, Engineering and Applications, Bharathidasan University,
Tiruchirappalli-23, Tamilnadu, India

Abstract: Web Services has instigated it's transcend and now education has been made simple through Web Services. With the advent of Web Services, education has become far more personal, flexible and available across global borders. Workflow is a sequence of business tasks to be realized for the execution of user' request. Identifying required e-learning web services and dynamic composition and realization of those services is a challenging process. In this study we have suggested e-learning services workflow composing architecture and relevant algorithms for matching and composing e-learning flow for the learners with different learning styles. We suggested non logic based hybrid matching and composing algorithms which uses OWL-S profile and process ontologies for dynamic workflow composition of e-learning web services.

Keywords: Ontology, ontology matching, OWL, semantic web, web services, workflow

INTRODUCTION

Web 1.0 was more towards presenting content over Internet rather than providing user generated content. The content generated was unstructured, redundant and huge. According to W3C Director, it was only read only web and provided one way communication. There was very less dynamic content generated where as web 2.0 is more about collaboration, user involvement and interaction but lacks personalization and context. Web 3.0 is about the Web becoming smarter, getting to know the user better from browsing history and automatically delivering content to the user that is relevant. According to the W3C, The Semantic Web provides a common framework that allows data to be shared and reused across applications, enterprise and community boundaries. Web 3.0 is about semantic web, personalization, intelligent search and intelligent business applications. In future web search will be based on content not based on keywords. We need to move to the upcoming semantic web technologies which are sprawling to various fields. From static content generation web is moving towards providing the computing and data based web services.

Web Services provide a basis for interoperability between service providers and consumers, based on the reliable exchange of messages (Martin *et al*., 2007). Upon the users request, more than one web services are combined and serve the user. The steps involved in invoking sequence of web services to satisfy the user's request form the work flow. This web service or workflow composition involves many aspects such as:

- Selection of web services
- Control flow and data flow among the services
- Integration
- Composition of web services

Achieving all these functions manually is cumbersome and time consuming. Automated processes catch the attention of many domains and provide benefits such as generation of work flow on the fly, deriving business intelligence and also it leverages to the use of new semantic web technology such as OWL-S, WSDL, UDDI and SOAP (W3C, 2004) provide only syntactical support for discovery and integration of web services and manual intervention is required in all phases of service realization. Semantic web which is next generation web presents intriguing challenges and provides promising benefits. Automation can be achieved in all phases of work flow execution using semantic web technologies. In this study, we have concentrated on composing workflow for e-learning services. Semantic Web Service coordination aims at the coherent and efficient discovery, composition, negotiation and execution of Semantic Web Services in a given environment and application context (Klusch *et al*., 2005). Discovery involves locating web services based on functional and non functional parameters. Syntactical nature of WSDL doesn't allow the composition or orchestration of these services dynamically or in other way with user intervention.

Corresponding Author: M. Farida Begam, School of Computer Science, Engineering and Applications, Bharathidasan University, Tiruchirappalli-23, Tamilnadu, India

Discovery of candidate web services can be accomplished using two ways. First way using inference of WSDL documents. In this way, clustering parameter names of operations on a set of web services into semantically meaningful concepts and finding the similarity helps to find the candidate services. This trend of research work focus on input/output matching to find a list of operations with similar inputs (outputs) with a given inputs (outputs) of an operation and operation matching to find a list of operations similar with a given operation. Second way is defining standard description language at top level of WSDL and devising algorithms to find the capabilities of the services. This will provide more expressiveness. In general, any semantic service discovery framework needs to have the following components (Klusch, 2008).

Service description: Formal means to describe the functional and non-functional semantics of Web Services.

Service selection: Reasoning mechanisms for service matching, that is the pair wise comparison of service descriptions in terms of their semantic relevance to the query and ranking of the results based on partially or totally ordered degrees of matching and preferences.

Discovery architecture: Environmental assumptions on (centralized, decentralized) network topology, service information storage (e.g., distribution of services, ontologies, registries) and location mechanisms and functionality of agents involved (e.g., service requester, provider, middle agents).

Web Services advertisement follows structure which cannot be managed with normal RDBMS tables. We need mechanism to store, process, retrieve and manage these web services efficiently. Semantic web technologies provide solution for all these requirements. Automatic composition can be achieved with Web Ontology language OWL-S in building semantic web services. Learners with different requirements or learning styles are identified based on their behavior, actions and interactions with the system and we provide e-learning agenda or sequence of e-learning web services (e-learning flow) based on their learners profile or style. In this study we have proposed an work flow composition architecture and non logic based hybrid matching and composing technique which uses IOPE (Input, Output, Preconditions and Effect) as well as text description for matching candidate e-learning web services using on OWL-S.

MATERIALS

Semantic web is the boon for next generation to provide the meaningful information to the Internet world. It is the vision of W3C for the web of linked data. Semantic web is an extension of available web, which will unleash the revolution of new possibilities (Berners-Lee, 2001). In many fields like medicine, bio informatics and web search and data mining, it has got widespread usage. It takes up its strength in different domains and providing many knowledge based intelligent solutions. E-learning is not an exception. Semantic web services present one more layer above existing web and allow the users to interact with the web in meaningful way. It provides information in a precise, machine interpretable form, ready for software agents to process, share and reuse it, as well as to understand what the terms describing the data mean (Fayed *et al*., 2006). In e-Learning if processes are described with semantics; it will make the applications to interact in a meaningful way, in turn serves the heterogeneous learners to acquire the knowledge according to their perception. This approach will enhance the leaner's experience. Main objective of this semantic web is to extend the current web technology to allow the development of intelligent agents, which can automatically and unambiguously process the information available on millions of web pages (Berners-Lee, 2001). These software based agents provide appropriate infrastructure and help in achieving development of semantic evolution of e-Learning systems. Conceptualization, Ontologies and data interchange formats RDF, XML, XTM (XML Topic Maps), OWL-Web Ontology Language, OWL-S, RDF Schema, Rule ML (Rule Mark up Initiative) and SPARQL (W3C) are the basic technologies to develop formal description of concepts, terms and relationships within a given knowledge domain. Using OWL and OWL-S, the web resources are converted to proper knowledge representation such as semantic annotations. These annotations can be used to support the user in composing work flows for various business processes including e-learning processes. Moreover, Semantic web services provide a number of different educational activities by transforming a static collection of information into distributed way on the basis of Semantic Web technology making content within the WWW machine-processable and machine-interpretable (Ibert *et al*., 2008).

Semantic web architecture: Semantic web is the next generation of the WWW and it extends the current Web by giving information well-defined meaning, better enabling computers and people to collaborate. In Semantic Web information has machine-processed and machine-understandable semantics that enhancing the machine readability of web content. Figure 1 illustrates the various technologies used in different layers of abstraction in the Semantic Web.

Unicode and URI: This is an international standard for encoding the text. This includes all scripts in active use today, many scripts known only by scholars and

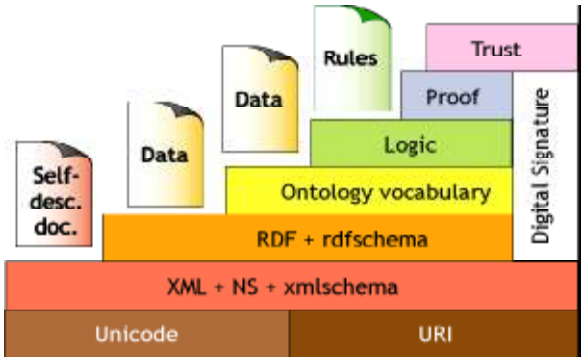


Fig. 1: Layers of the semantic web (www.w3.org/2001/12/semweb-fin/swlevels.png)

symbols which do not strictly represent scripts, like mathematical and linguistic symbols. URI-Uniform Resource Identifier is used to identify any resource such web page, e-book, e-Learning material such as slide, presentation and etc.

XML+NS+rdfschema: XML (eXtensible Markup Language) XML Schema ensures that there is common syntax used in Semantic web. XML Name Spaces allow specifying different markup vocabularies in one document. XML Schema serves for expressing schema of a particular set of XML documents.

RDF+rdfschema: RDF (Resource Description Graph) is core data representation format for semantic web. It is based on triples subject-predicate-object that form graph of data. RDF Schema is used to define the vocabulary of RDF model.

Ontology vocabulary: Ontology is a formal specification of conceptualization. It describes the knowledge base of the particular domain. It has set of components such as Classes, slots, facets and instances. Concepts of the domain are represented as classes, properties of concepts called slots describing various features and attributes of the concepts and restrictions on slots are called facets. There are several higher level languages for generating Ontologies such as OIL (Ontology Inference Layer) and DAML (DARPA Agent Markup Language) +OIL and OWL. Wide usage and adoption of OWL makes this language as standard for Ontology representation language for Semantic web. It has set of XML elements and attributes which defines the terms and relationship among these terms in well structured manner. It also extends the elements of RDF and RDFS. OWL includes the concepts like union, intersection and complement of Ontology. Cardinality constraints on properties can be imposed on properties. Properties can also have characteristics in OWL. On the whole, OWL provides better way of representing knowledge base which is required in semantic web services. It helps us to obtain the shared understanding of the key concepts, ensures reusability of knowledge base and makes distributed large scale machine processing automatic.

Ontologies are the constructs which should be well defined, with machine readable computational semantics, commonly acceptable understanding that describes a particular domain. It is one of the ways of knowledge representation. Knowledge of particular domain is captured, stored and retrieved as ontologies. Ontologies are involved in extracting knowledge and making intelligent decision. Many intelligent agents use the ontology, derive the intelligence and passes this to other intelligent agents.

Semantic web services: Web Services are modular, self-describing, self-contained distributed applications that are accessible over the Internet. These loosely coupled and contracted components that communicate via XML based interface and encapsulate discrete functionality. They are programmatically accessible over standard Internet protocols which are SOAP, WSDL and UDDI and they define standards for service discovery, description and messaging protocols. They provide syntactic interconnection between heterogeneous systems. Web Services facilitate many businesses with the availability of applications through platform independent technology. At the core, Web Services faster the way the business and people interact with each other. In the field of education, Web Services has instigated it's transcend and now education has been made simple through Web Services.

The mainstream XML standards for interoperation of Web Services specify only syntactic interoperability, not the semantic meaning of messages. Web services based Service Oriented Architecture provides heterogeneity and interoperability in realizing the service request. However, there are certain deficiencies in normal web services which use only syntactical information descriptions, syntactic support for discovery, composition and execution, web Service usability, usage and integration needs to be inspected manually. There is no semantically marked up content/services. It does not support the Semantic Web. Above all, dynamically combining services using WSDL is very difficult to achieve as WSDL handles input and output as strings rather than the necessary concept for combining them.

Keeping ontologies as data model, Semantic web services increases the effectiveness of the output obtained from the WWW. It helps to improve machine supported interpretation. Semantic web services are realized using OWL-S Web Ontology Language for Web Services. It defines complete description frameworks for describing Web Services and related aspects. It supports ontologies as underlying data model to allow machine supported data interpretation and

define semantically driven technologies for automation of the discovery, integration and composition of web services and their usage process. With the advent of Semantic Web Services, education has become far more personal, flexible and available across global borders.

OWL-S web ontology language-services: OWL-S is W3C standard for annotating web services. It is written using OWL and its aim to provide general terms and properties to describe web services. Adding semantics to web services changes the way the web applications work. These annotations with OWL-S help to achieve high automation in discovering the service, invoking and integrating/composing the service from different other services. The OWL-S ontology is still evolving and making connections to other development efforts, such as those building ontologies of time and resources. OWL-S ontology is referred as a language for describing services, reflecting the fact that it provides a standard vocabulary that can be used together with the other aspects of the OWL description language to create service descriptions (W3C, 2004). OWL-S is not intended to replace existing Web Services or Semantic Web standards. It remains compliant with existing standards and adding explicit semantics that is operational and understandable to computer programs. Web service community needs this which helps to encompass comfort with descriptions of entities outside current Web Service specifications. OWL-S enables or motivates us to approach web services with different scenarios of automation processes discussed below.

OWL-S is upper ontology for services i.e., it is a general ontology not meant for any particular ontology. It is also called foundation ontology. OWL defines the web service on three important aspects given as what does the service do?, How does the service work? and How is the service invoked? OWL-S lays its basis on a process ontology and benefits from developments in workflow modeling as well as the agent technology (McIlraith *et al*., 2001). OWL-S describes services by three components namely, Service Profile, Service Model and Service Grounding.

The first aspect is defined by OWL-S sub-ontology called Profile ontology which describes the classes and properties to answer this type of question. This is to advertise the service. Equivalent to WSDL document in normal web services architecture. The second aspect is defined by the sub ontology Service Model/Process ontology which defines all the terms need to describe how the service works and the third aspect is defined by the sub ontology Grounding ontology which is meant for providing the terms used to describe how the service can be accessed technically. These three ontologies are connected through a high level ontology called service ontology. This describes how these three ontologies can work together to describe the service. The above three ontologies are used in achieving the goals of OWL-S such as automation, machine interpretation and machine processing. Automatic discovery can be achieved through Profile ontology, automatic invocation is achieved through Grounding ontology and automatic composition and monitoring is achieved by Process ontology.

Survey on semantic web service composition: This section briefly explains about various workflow composition techniques used in real business applications. Web service integration is the process of finding a web service that can deliver a particular service and composing several services in order to achieve a particular goal, However, semantic web service descriptions do not necessarily reference the same ontology. Henceforth, both for finding the adequate service and for interfacing services it is necessary to establish the correspondences between the terms of the descriptions of the web services.

Kim *et al*. (2004) specify AI planning techniques in which rich knowledge base is used for automatic composition. They have implemented an approach to interactive workflow composition that incorporates:

- Knowledge-rich descriptions of the individual components and their constraints.
- A formal algorithmic understanding of partial workflows, based on AI planning techniques.

Using this approach, a system can analyze a partial workflow composed by the user, notify the user of issues to be resolved in the current workflow and suggest to the user what actions could be taken next.

Cardoso and Amit (2003) discussed about QoS model for work flow composition. They have brought a model that will allow for the discovery of Web services and for the composition of workflows based on operational requirements such as which includes their timeliness, quality of products delivered, cost of service and reliability. They have recommended that a good management of quality leads to the creation of quality products and services, which in turn fulfills customer expectations and achieves customer satisfaction.

Fei and Lu (2012) uses data flow driven approach for composing scientific workflow. In this study they have suggested a dataflow-based scientific workflow model that separates the declaration of the workflow interface from the definition of its functional body; a set of workflow constructs, including Map, Reduce, Tree, Loop and Conditional which are fully compositional one with another, a dataflow based exception handling approach to support hierarchical exception propagation and user-defined exception handling are the unique features of this study.

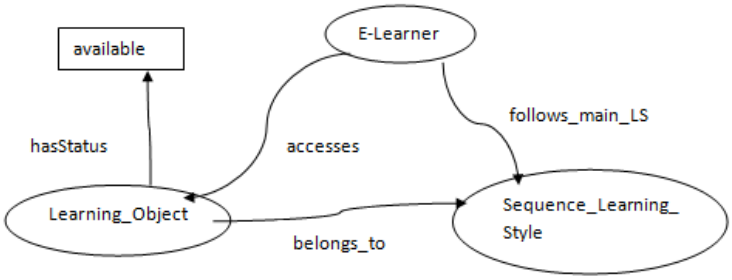
Table 1: Various semantic web service composition techniques

Approach	Based on functional/QoS based	Technique used	Description
Chinthaka *et al*. (2009)	Non QoS based	Case based reasoning	Composition of workflows based on the characteristics of the inputs and the outputs of the reusable workflow components, facilitating user exploitation of existing services and workflows during workflow composition
Chiu *et al*. (2009)	QoS based	Ontology based AI planning	Defining cost models on service completion times and error propagation, composes service workflows which can adapt to user's QoS preferences
Sohrabi and McIlraith (2010) approach	Non QoS based	Situation calculus and first order logic	The composition system uses the user preferences to generate preferred solutions
Gil *et al*. (2009)	QoS based	Ontology based AI technique	It assumes a distributed architecture where data and component catalogs are separate from the workflow system, queries to external catalogs and therefore any reasoning regarding data or component properties is not assumed to occur within the workflow system
Lécué *et al*. (2008) approach	Non QoS based	AI and DL reasoning	DL reasoning between the input, output parameters (DL based description) of service in order to infer semantic matchmaking between parameters along with the situation calculus
Aydın *et al*. (2008) approach	Non QoS based	Abduction theorem and event calculus	Abduction theorem generates a series of events as well as a set of temporal ordering predicates, giving partial ordering of events which are more suitable for web service composition
Lécué *et al*. (2008) approach	QoS based	Casual Link Matrix(CLM) -for functional parameters CLM++ for non functional parameters	Semantic connections between services are stored on CLM+ which can be used to compute the composition that represents the possible service composition that matches the service request
Zeng *et al*. (2008) approach	QoS based	Forward chaining and backward chaining used	Goal-directed service composition and optimization framework is presented. The goal directed problem takes three inputs, the domain specific composition rules, the description of the business objectives and the description of the business assumptions
Ardagna and Barbara (2007) approach	QoS based	Processes with Adaptive Web Services (PAWS), BPEL used	SLA based constraints are used
Groenmo and Jaeger (2005) approach	QoS based	Semantic web languages	Four phases: • For constructing composition model • We service handling based on semantic matching • Obtaining composition model • Composition model used
Klusch *et al*. (2005)	QoS based	Fast forward-planner with a HTN planning	OWL-S service composition planner, called OWLS-Xplan is used to convert OWL-S 1.1 services to equivalent problem and domain descriptions that are specified in the planning domain description language PDDL 2.1 and invokes an efficient AI planner Xplan to generate a service composition plan sequence that satisfies a given goal.
Majithia *et al*. (2004)	Non QOS based	Semantic web technology	The framework consists of 2 core and 5 supporting services
Kim *et al*. (2004)	Non QoS based	Semantic technology with planning technique	Constructs partial work flow combines knowledge bases (that have rich representations of components) together with planning techniques (that can track the relations and constraints among individual steps). And implemented system called CAT (Composition Analysis Tool) that analyzes workflows and generates error messages and suggestions in order to help users compose complete and consistent workflows
Cardoso and Amit (2003)	QoS, functional and semantic based matching	Uses OWL-S	Provide techniques for various composition techniques of all types

Laukkanen and Heikki (2004) designed workflow composer agent which performs matching operation on IOPEs description of semantic web services. Various semantic web service composition techniques adopted by different approaches have been listed out in the Table 1 with description. From the information collated in table we can ensure that OWL-S based e-learning composition framework has not yet been addressed elaborately.

METHODOLOGY AND DISCUSSION

In our e-learning flow composition we consider the atomic data services. Many universities now provide



Web Service Name:	Sequence_Learning_Style Service
Description:	E_Learning_Material-Returns Java Introduction page-belongs_to Sequence_Learning_Style
Precondition:	Learning_Object hasStatus available, E-Learner follows_main_LS Sequence_Learning_Style, Learning_Object belongs_to Sequence_Learning_Style
Input:	Learners' Learning_Style
Post Condition/Effect:	E-Learner accesses Learning_Object
Output:	ConfirmationMessage, OutputURL

Fig. 2: A simple sequential learning style learning object/e-learning service black box description

their e-learning materials or Learning Objects as web services. A e-learning service or learning object has the following minimum service description in terms of profile and process ontologies.
E_Learning service:

Input : Learning_Style
Output : OutputURL
Precondition : E-Learner follows_main_LS Sequence_ Learning_Style
Effect : E-Learner Accesses the Learning_ Object

Learning objects which are to be distributed as web services should be annotated with proper text description, service name and input and output precondition and Effects are the better parameters for matching the service with user request. E-Learner's learning style, set of keywords related to learning style, can be represented in text description of an template ontology for matching. This ontology is given as input and matched with e-learning service profile as well as process ontology, compared and degree of matching is considered to select the candidate e-learning service for composition. Algorithm in learning style detection can be used here for finding the matching candidate service.

Likewise many e-learning semantic services available can be searched for e-learning services and can be included in e-learning flow and this sequence is given to execution module for execution. E-learning Object description should contain which learning style that learning object belongs to. This can be computed/identified from searching the profile ontology details such as text description, service name, input and output as well as service category parameters and process ontology details input, output, preconditions and effect. Figure 2 depicts the sample e-learning Service which is actually a Learning Object with IOPEs listed in table.

The following listing gives an idea of how the e-learning service or learning object can be annotated using profile and process ontologies. Precondition and Effect are described using KIF (Knowledge Interchange Format).

Semantic service matching finds the similarity between the annotations or semantics of desired service with that of advertised service. This is the main part of any matching algorithm. Our discovery and matching algorithm use text description of profile ontology and precondition and effects of process ontology of semantic web service searched against the E-Learner's ontology parameters such as interests, his learning style he follows. Matching process can be non logic, logic or hybrid. Logic based semantic matching is based on logic inferences and description logic rules and Non logic based match makers widely available rather than logic one. There are variety of non logic matching techniques such as direct, indirect, logic based, similarity based and Graph matching. In Logic based matching we have different types of matching such as exact, plug-in, subsumption and fail matching. If two services S1 and S2 preconditions and Effects exactly same this is exact match. S1 can be substituted by S2. If Effects of S1 matches with preconditions of S2 then S1 is prerequisite or to be finished first and then S2 to be executed next. Effect of S1 is stricter than Effect of S2 then S1 subsumes S2. If S1's condition partially satisfies S2 condition then it is considered as plus match. No such mentioned match is existing then it is fail match. There are two non logic based matching

possible such as subsumed by and nearest neighbor matching. In hybrid subsumed by matching selects services in such a way that requested input is sub set of service and output data of service is slightly more general than requested, hence, in this sense, request is subsumed by the service. In nearest neighbor matching as well as hybrid subsumed by matching check the degree of text similarity between the input If matching is based only on profile it is called black box matching and matching is based on only process it is called glass box matching. In the following algorithm, hybrid of black and glass box matching is introduced to e-learning services as we rely on learning style detection to achieve personalization in e-learning and compute e-learning flow. Only functional parameters such IOPEs not sufficient to find out whether the service is to be included in e-learning-workflow. Learning objects which are to be distributed as web services should be annotated with proper text description, service name and Input, Output, Precondition and Effects and these are the better parameters for matching the service with user request. We consider the e-learning data services.

Matching algorithm is also based on the text description which contains important keywords relevant to the learning style of the learners, For example for Sequential learning style based web service/learning object text description may contain keywords like Introduction, Theory, Basic, Fundamental. For Visual learning style based web service/learning object the text description could contain keywords like Animation, Video, Multimedia, etc. E-Learner's learning style identified in module second, set of keywords related to his/her learning style, can be represented as an ontology which we call it as Learner Service Template LST Ontology. This is service request ontology and it is given as input and matched with e-learning service which we consider as Learning Service Object LSO profile and as well as process ontology, compared. Degree of Matching (DoM) is considered to select the candidate e-learning service for composition. If it is computing service, input and output parameter are the phenomena for matching operation. In e-learning domain, the suitable e-learning services or learning objects to be found for composing the e-learning-workflow. Hence the keywords in text description and preconditions and post conditions should also be matched against the advertised services.

Listing 1: profile and process ontology OWL listing:

```
<?xml version = "1.0" encoding = "UTF-8"?>

<!DOCTYPE uridef [
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf syntax-ns">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd "http://www.w3.org/2001/XML Schema">
  <!ENTITY service "http://www.daml.org/services/owl-s/1.2/Service.owl">
  <!ENTITY profile "http://www.daml.org/services/owl-s/1.2/Profile.owl">
  <!ENTITY process "http://www.daml.org/services/owl-s/1.2/Process.owl">
  <!ENTITY grounding "http://www.daml.org/services/owl-s/1.2/Grounding.owl">
  <!ENTITY javaGrounding "http://bai-hu.ethz.ch/owlet/ont/OWLSExtensions.owl">
  <!ENTITY expr "http://www.daml.org/services/owl-s/1.2/generic/Expression.owl">
  <!ENTITY swrl "http://www.w3.org/2003/11/swrl">
  <!ENTITY myonto: "http://www.semanticweb.org/lenovo/ontologies/2013/8/E_Learning_Onto#">
]>
<rdf:RDF
        xmlns:rdf="&rdf;#"
        xmlns:rdfs="&rdfs;#"
        xmlns:owl="&owl;#"
        xmlns:xsd="&xsd;#"
        xmlns:service="&service;#"
        xmlns:profile="&profile;#"
        xmlns:process="&process;#"
        xmlns:grounding="&grounding;#"
        xmlns:expr="&expr;#"
        xmlns:swrl="&swrl;#"
        xmlns:javaGrounding="&javaGrounding;#"
        xmlns="&DEFAULT;#"
        xml:base="&DEFAULT;" >
 <owl:Ontology rdf:about="">
        <owl:imports rdf:resource="&service;"/>
        <owl:imports rdf:resource="&profile;"/>
        <owl:imports rdf:resource="&process;"/>
        <owl:imports rdf:resource="&grounding;"/>
 </owl:Ontology>
 <!-- Service description -->
 <service:Service rdf:ID="E_learning_Service">
  <service:presents
 rdf:resource="#E_Learning_Profile"/>
  <service:describedBy
 rdf:resource="#E_Learning_Process"/>
  <service:supports
 rdf:resource="#E_Learning_Grounding"/>
 </service:Service>
  <!-- Profile description -->
 <profile:Profile rdf:ID="E_Learning_Profile">
 <service:isPresentedBy
 rdf:resource="#E_Learning_Service"/>
 <profile:serviceName
 xml:lang="en">E_Learning_Service</profile:service
 Name>
 <profile:textDescription
 xml:lang="en">E_Learning_Material-Returns Java
 Introduction page-belongs to Sequence_Learning_
 Style</profile:textDescription>
 <profile:hasInput
```

```
   rdf:resource="&myonto;#Sequence_Learning_Style"
   />
    <profile:hasOutput rdf:resource="#OutputURL"/>
  </profile:Profile>
  <process:AtomicProcess
  rdf:ID="E_Learning_Java_Introduction">
  <service:describes
  rdf:resource="#E_Learning_Service"/>
    <process:hasInput
   rdf:resource="#Sequence_Learning_style"/>
    <process:hasOutput rdf:resource="#OutputURL"/>
</process:AtomicProcess>
<process:AtomicProcess
rdf:ID="E_Learning_Introduction">
  <process:hasInput>
     <process:Input
    rdf:ID="Sequence_Learning_Style"/>
          <rdfs:subClassOf
          rdf:resource="&myonto;#Learning_Style">
         <process:parameterType rdf:resource =
         "&myonto;#Sequence_Learning_Style">
  </process:hasInput>
<process:hasOutput>
     <process:Output rdf:ID="ConfirmationMessage"/>
        <process:parameterType
        rdf:datatype="&xsd;#String">
       &xsd;#anyURI</process:parameterType>
        <rdfs:label>Confirmation Message</rdfs:label>
             </process:hasOutput>
<process:hasOutput>
  <process:Output rdf:ID="OutputURL"/>
           <process:parameterType
           rdf:datatype="&xsd;#anyURI">
           &xsd;#anyURI</process:parameterType>
  <rdfs:label>Sequence_Learning_Material</rdfs:label>
 </process:hasOutput>
<process:hasResult>
  <process:Result>
   <process:inCondition>
    <expr:KIF-Condition>
      <expr:expressionBody>
           (exists (?E1 ?LS1)
           (and (instance ?E1 E-Learner)
               (instance ?LS1 Sequence_Learning_Style)
               (follows_main_LS ?E1 ?LS1)))
      </expr:expressionBody>
   </expr:KIF-Condition>
   </process:inCondition>
<process:inCondition>
   <expr:KIF-Condition>
     <expr:expressionBody>
          (hasStatus Learning_Object available)
     </expr:expressionBody>
   </expr:KIF-Condition>
   </process:inCondition>
   <process:inCondition>
    <expr:KIF-Condition>
      <expr:expressionBody>
          (exists (?LO1 ?LS1)
          (and (instance ?LO1 Learning_Object)
              (instance ?LS1 Sequence_Learning_Style)
              (belongs_to ?LO1 ?LS1)))
      </expr:expressionBody>
   </expr:KIF-Condition>
   </process:inCondition>
    <process:hasEffect>
      <expr:KIF-Condition>
       <expr:expressionBody>
        (exists (?E1 ?LO1)
        (and (instance ?E1 E-Learner)
            (instance ?LO1 Sequence_Learning_Style)
            (accesses ?E1 ?LO1)))
       </expr:expressionBody>
      </expr:KIF-Condition>
  </process:hasEffect>
</process:Result>
   </process:hasResult>
</process:AtomicProcess>
```

Our approach uses text description, as well as IOPEs for composing our e-learning flow. This kind of matching considered here is non logic based subsumed by or nearest neighbor flexible approach based on similarity checking and information retrieval. Input is learning style of the learner and output usually a web site or link to download the Learning Object. Learning Objects can be reading material, puzzle, assignment, YouTube lecture, problem to be solved or multimedia based material (Animation, Audio/video). Figure 3 illustrates the e-learning flow composition architecture which includes matching, discovery and composition modules.

E-learning services which are ready for execution and realization are called Grounded Learning Services (GLS). Assume zero or more GLSs are already realized. Due to nature of e-learning materials, we considered atomic web services. When the designer wishes to add a e-learning service to the e-learning workflow, system starts by creating a Learner Service Template (LST) ontology. LST contains:

- Input, Output, Precondition and Effect. Input the name of the learning style the Learner follows
- Set of Keywords related to Learning Style be matched with e-learning service text description Input is the type of Learning style the learner follows. Output is an URL of the Learning Object to be accessed. The Fig. 4 illustrates the construction of LST from Keywords and IOPEs and Keywords

Structure of GLS, LST and LSO: The components used in discovery and integration of e-learning services have some structure or format defined. We present the structure of all these to be used in composition process.

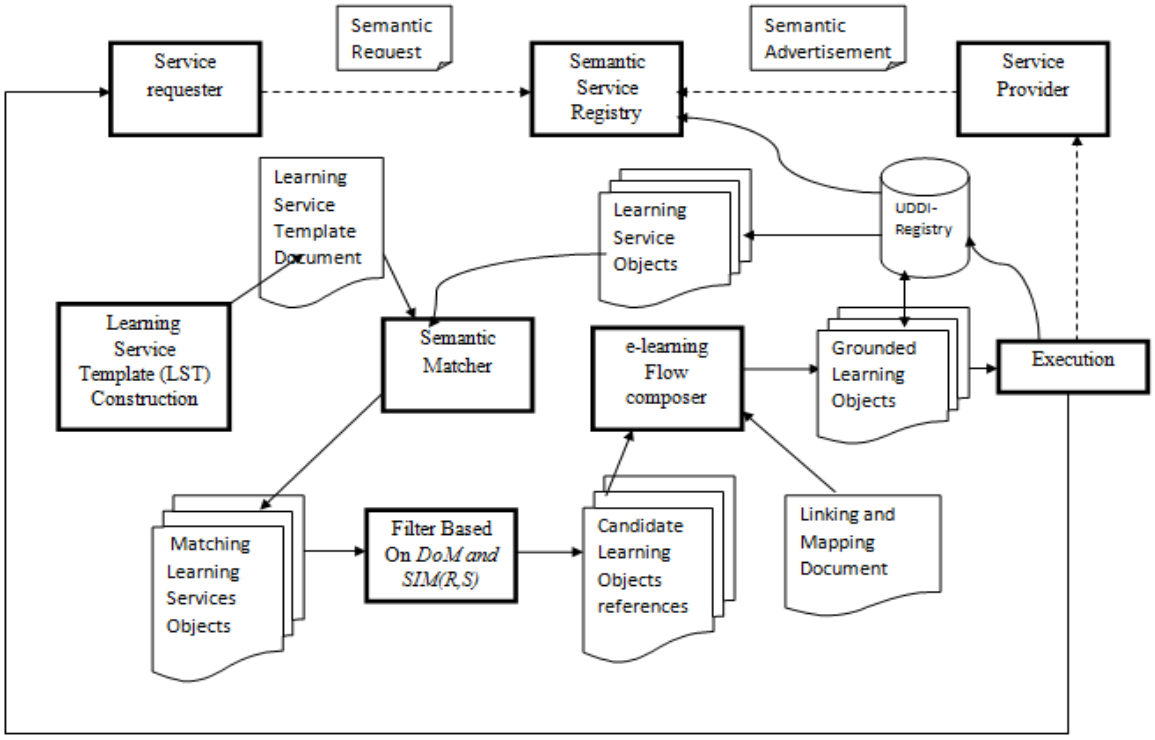


Fig. 3: Architecture of e-learning flow composition

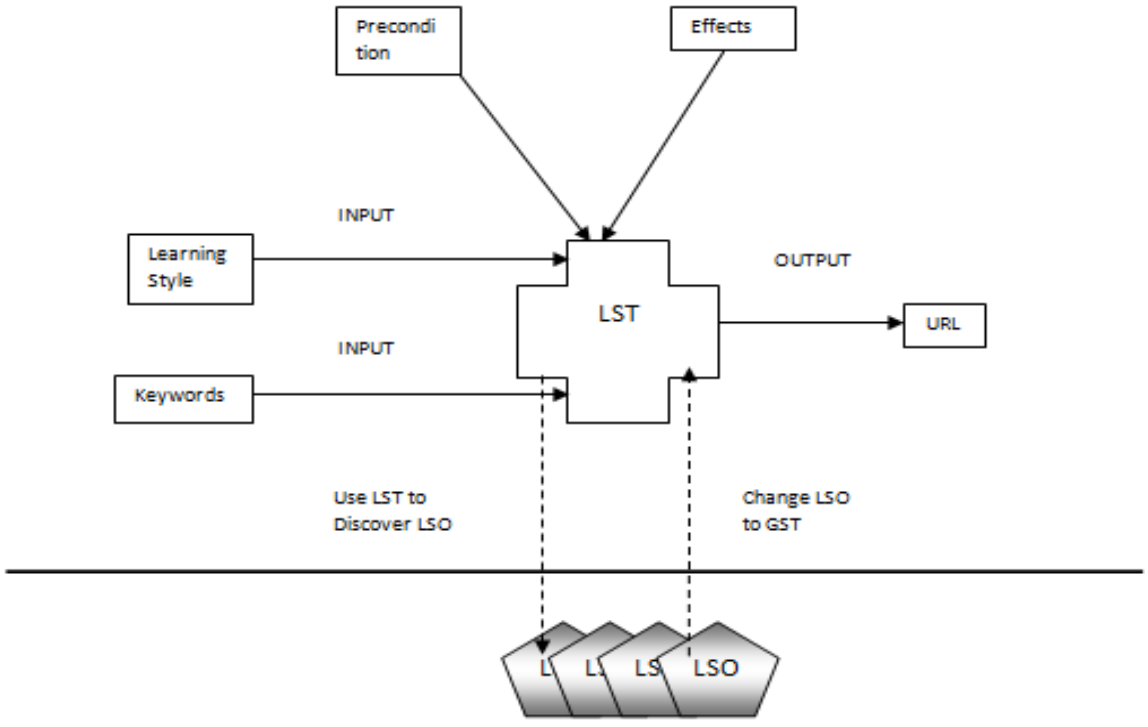


Fig. 4: LST construction

Grounded learning service: Grounded Learning Services are to be realized and used to construct the e-learning flow which is our main aim of the system designed. It can be defined as tuple of:

$$GLS(n) = <TD, Is, Os, PC, E>$$

where,

n = Name of the service

TD = Text description
Is = SET of input parameters
Os = Set of output parameters
PC = Precondition
E = Set of effects respectively

The entire GLS specification is to be used by algorithms to synthesize the workflows based on matching of profile and process ontology. We may give example with the following structure:

GLS (*"JavaIntroduction"*) = <{"E_Learning _Material-Returns Java Introduction page-belongs to Sequence_Learning_Style}, {"Sequential LearningStyle"}, {"http://www.JavaIntro.com/Intro.html"}, {myonto: E-Learner myonto: follows_main_LS myonto: Sequential Learning Style}, {myonto:E-learner myonto: accesses myonto: Learning_ Object}>

The inputs, outputs, precondition and Effects in this example are associated with ontological concepts described in Learning Style detection section.

Learning service template: Learning Service Template is used to search for candidate e-learning service. LST contains the information that is used to define the characteristics of the e-learning service to be found for e-learning flow.
A LST is specified as:

$LST = <lsn, lsd, Os, Is, PC, E>$

Six fields exist: *lsn*, *lsd*, *Os*, *Is*, *PC* and *E*.

lsn is the name of the Web service to be found. The name specified does not have to syntactically match exactly with the name of the Web services to be discovered. The *lsd*, *Os* and *Is* fields correspond to a textual description and a set of output and input parameters, Post Conditions and effects respectively, of the Web service to be matched. *LSO* are also in the same structure as that of GLS.

Semantic matching LST with existing LSOs: Ontology operations such as ontology alignment, ontology mapping and ontology matching and Ontology integration are very much useful to unfold the research in the fields like data integration, peer to peer systems, e-commerce, information retrieval and query answering, as well as in social networks. Among these ontology operation we have considered ontology matching to find the similarity between two service profile ontologies. The name of the service, inputs and outputs of LST need not match exactly with available learning object. In matching process, considering only syntactic matching of inputs and outputs will not result better e-learning flow as we all know that authors/providers of e-learning objects use synonyms of the same word for their web service names and description. Semantic match should be considered for selecting the candidate services. For example the learner searches for e-learning service on ontology example diagrams, then the keywords such as diagram, image, illustration can be considered the same and searched. We need identify the set of keywords to be used with each learning style. We assume that service provider, requester and matchmaker share a basic minimal vocabulary of primitive components together with a set of mapping rules such as synonym relations in the thesaurus WordNet. The final e-learning workflow consists of all services which are semantically matching with input, output, preconditions and effects as well as keywords specified Textual description in Learning Service Template. The Learning service object is a structure that holds the description of a real Web service. As stated earlier, we specified Web services semantically. In matching process the template ontology LST is matched against list of Learning Service Objects LSO ontology.

In our Ontology matching computation, these are possible the possible results, equality, subsumption, consequence or disjointness. We need to find the correspondence between LSTs IOPEs and text description to that of LSOs i.e., find the mapping between *LST.I*, with *LSO.I*, *LST.O* with *LSO.O*, *LST.PC* with *LSO.PC*, *LST.E* with *LSO.E*. The textual description of LST is divided into set of tokens and matched with LSO textual description. Mapping between LST.TD with *LSO.lsd*. Degree of Matching function *DoM* (*LST*, *LSO*) is further divided into two sub functions one for IOPEs mapping DoM_f and another for textual description DoM_{lsd}:

$$DoM(LST, LSO) = (DoM_f + DoM_{lsd}) /2$$

Degree of alignment between two ontologies is defined as five tuple:

$<id, e1, e2, n, R>$

- *id* is a unique identifier of the given correspondence
- *e*1 and *e*2 are entities (e.g., XML elements such primitive data types, properties, classes) of the first and the second ontology, respectively
- *n* is a measure (typically in the (0, 1) range) holding for the matching between e1 and e2
- *R* is a relation (e.g., equivalence (=), subsumption ($\subseteq$), disjointness ($\perp$) holding between *e*1 and *e*2

The degree of matching between two ontologies can be measured in the range (0, 1) is sum of all the equivalence alignments measure and subsumption alignments measures divided by number of total alignments that can be considered say *m*. Disjoint alignments are ignored in the process because they are

unmatched entities. Thus the degrees of matching is given by the function as follows:

$$DoM\ (LST, LSO) = (\sum n= + \sum n_{\subseteq}) /m$$

While finding the matching between two entities in ontologies synonyms to be considered as equivalence relation.

Procedure for matching and discovery of candidate learning object services:
Input:

- Advertised Services (Process, Profile and Grounding Ontologies)
- Learner Service Template LST Ontology

Output: Finding the List of candidates e-learning services/Learning Objects:

- Construct the LST ontology which is service request ontology.
- Send the LST to Semantic matcher filters based on subsumed by and nearest neighbor matching and finds the degree of matching and linguistic similarity between LST and e-learning services objects available in web.
- Semantic Matcher returns a set of Learning Service Object (LSO) references according to their degree of Matching and similarity metric value based on information retrieval with the Learner Service Ontology.
- Consider the maximum *d* for *dom*. Filter is used to Select the Learning service objects which have *DoM* above the threshold d. Consider them as candidate Learning Service Objects and Send them to composer module to create the e-learning-flow.

Semantic matching: This part of the composition process involves finding matching candidate services based on degree of matching and syntactic similarity between LST and advertised learning services objects.

Algorithm 1:
Match: Find advertised learning services objects LSO S that best match in a hybrid fashion with a given Learning Service Template LST R; returns set of (*S*, degree Of Match, SIM_{IR} (*R*, *S*)) with degree of match (*dom*) is greater than or equal to maximum *d* unequal FAIL and syntactic similarity value exceeding a given threshold α:

1: function match (LST *R*, α, *d*)
2: local result, degree Of Match, *hybridFilters* = {subsumed-by, nearest neighbor}
3: for all (*S*, *dom*) ∈ $candidates_{inputset}$ ($inputs_R$) ∧ (*S*, *dom′*) ∈ $candidates_{outputset}$ ($outputs_R$) ∧ (*S*, *dom″*) ∈ $candidates_{preconditionsset}$ ($preconditions_R$) ∧ (*S*, *dom‴*) ∈ $candidates_{postconditionset}$ ($postconditions_R$) ∧ (*S*, *dom⁗*) ∈ $candidates_{textdescriptionset}$ ($textdescriptions_R$) do
4: degree Of Match ← *AVE* (*dom*, *dom′*, *dom″*, *dom‴*, *dom⁗*)
5: if degree Of Match ≥*d* ∧ (degree Of Match ∉ *LogicFilters* ∨ SIM_{IR} (*R, S*) ≥α) then
6: *result*: = *result* ∪ {(*S*, degree Of Match, SIM_{IR} (*R*, *S*))}
7: end if
8: end for
9: return result
10: end function

SIM_{IR} (*R*, *S*) similarity metric calculation function which is meant for content-based service I/O matching. There are variety of ways using which similarity measure can be computed. Cohen *et al*. (2003) and his colleagues suggested few top most similarity metric measurement formulas such as, Extended Jaccard similarity metric, Jensen-Shannon information divergence based similarity metric.

Algorithm 2: Find Learning services objects which *input* matches with that of the Learning Service Template; returns set of (*S*, *dom*) with minimum degree of match *dom* unequal FAIL.

Assume there are n number of inputs in LST (R) and s number of inputs in LSO (S):

1: function $candidates_{inputset}$ ($inputs_R$)
2: local *H*, *dom*, *r*
3: // If *LSO* input matches with multiple LST inputs the best degree is returned
4: $H := \{(S, IN_{S,i}, dom) \in \cup_{j=1..n} candidates_{input}(IN_{Rj}) \mid dom = argmax_l\{ (S, IN_{S,i}, dom_l) \mid 1 \le l \le n, 1 \le i \le s\}\}$
5: //If all inputs of service *S* are matched by those of the request, *S* can be executed and the minimum degree of its potential match is returned
6: for all *S* ∈ *LSOs* do
7: if $\{(S, IN_{S1}, dom_1), \cdots, (S, IN_{Ss}, dom_s)\} \subseteq H$ then
8: $r := r \cup \{(S, \min(dom_1, \cdots, dom_s))\}$
9: end if
10: end for
11: return
12: end function
13: function $candidates_{input}$ ($IN_{R,j}$) //Classify request input concept into ontology and use the auxiliary concept data to collect services that at least nearest neighbor match with respect to its input
14: local *r*
15: $r := r \cup \{(S, IN_S, \text{nearest-neighbor}) \mid S \in LSO, IN_S \in inputs_S, IN_S \cong IN_R, j,\}$
16: $r := r \cup \{ (S, IN_S, \text{Hybrid-Subsumed-by}) \mid S \in LSO, IN_S \in inputs_S, IN_S \ge IN_R, j,\}$
17: return *r*
18: end function

Algorithm 2 is also applicable for output, preconditions and post conditions (Effect) matching.

For text description matching, the text description in LST should be broken into words. Each word can be searched with that of LSO's text description. Synonyms of the words can be obtained from Word Net Thesaurus.

Algorithm 3: Find Learning services objects which text description matches with that of the Learning Service Template; returns set of (*S*, *dom*) with minimum degree of match *dom* unequal FAIL. Assume there are n number of words in LST (R) and s number of words in LSO (S):

1: function candidates$_{\text{textdescriptionset}}$ (inputs$_R$)
2: local *H*, *dom*, *r*
3: // If *LSO* text description word matches with each *LST* text description word and the average degree is returned
4: $H := \{(S, TD_{S,i}, dom) \in \cup_{j=1..n}$ candidates$_{\text{textdescription}}$ $(TD_{Rj}) \mid dom = argmax_l \{(S, TD_{S,i}, dom_l) \mid 1 \le l \le n, 1 \le i \le s\}\}$
5: //If all words in text description of service S are matched by those of the request, S can be executed and the minimum degree of its potential match is returned
6: for all $S \in LSOs$ do
7: if $\{(S, TD_{S1}, dom_1), \cdots, (S, TD_{Ss}, dom_s)\} \subseteq H$ then
8: $r := r \cup \{(S, AVE (dom_1, \ldots, dom_s))\}$
9: end if
10: end for
11: return
12: end function
13: function candidates$_{\text{textdescription}}$ ($TD_{R,j}$) //Classify request input concept into ontology and use the auxiliary concept data to collect services that at least nearest neighbor match with respect to its input
14: local *r*
15: for all Word Net Synonyms ($TD_{R,j}$) do
16: $r: = r \cup \{(S, TD_S,$ nearest-neighbor) $\mid S \in$ LSO, $TD_S \in$ inputs$_S$, $TD_S \cong IN_R, j,\}$
17: $r: = r \cup \{(S, TD_S,$ Hybrid-Subsumed-by) $\mid S \in$ LSO, $TD_S \in$ inputs$_S$, $TD_S \ge TD_R, j,\}$
18: end for
19: return *r*
20: end function

Word Net Synonyms ($TD_{R,j}$) is the function returns the set of strings which are synonyms for j^{th} word in Text description TD of LST R.

Procedure to compose the learning objects/learning material: In discovery module itself the candidate learning web services identified. Those services should be sorted according to DoM (Degree of Semantic Matching) and Similarity metric here.

Input: Learning Service Objects references (Grounded Learning Service) and their ranks.

Output: E-Learning Workflow.

- Sort the services obtained from match function according to their degree of match *dom* and similarity metric obtained from the function SIM_{IR} ().
- Use linking Documents which contain link related information for example reading material e-learning object should be followed by Exercise Learning Object to arrange the learning Service objects.
- Get the sorting order which is to be considered as selection order or e-learning flow.
- Send the selection order or e-learning flow to the execution function that automatically relates the realization with the list of LSOs, causing them to change the states to grounded services.
- A set of data mapping or linking document presented helps to the designer suggesting a possible interconnection among the newly created task interfaces and the grounded task interfaces.

Execution of e-learning flow: Realizes the work flow sequence obtained in composing module. Messaging happens to communicate among various entities in the sequence. Correct sequence of services executed and sequence will be provided to the Learners.

The theoretical background, architecture and algorithms for composition have been devised and discussed elaborately. The Learning Management System which incorporates this e-learning service composition is under implementation using Java, Jena API, Protégé knowledge management tool and OWL match making tool such as OWL-MX.

CONCLUSION

This study provides novel insight on how learning objects which are provided as e-learning services can be discovered and composed using hybrid non logic based matching. Algorithms are devised. For particular course/subject the learning objects are to be converted as data services and the interfaces can be modeled in such a way that user interaction, behavior captured and learning style detected for that user and provide the e-learning flow which consists set of learning objects for learning the subject. Suggested architecture and algorithms can also be utilized in any other domain other than e-learning where personalization and semantic web based context aware applications are to be focused. Learning is endless. World Wide Web plays a significance role in current academic community as well as in industry world. The days have gone when teacher centric learning methodologies are

predominantly adhered. Now education has no constraints or barriers such as time, space and people. Learning through web has become mandate requisite in all way of life. Semantic Web Services facilitate this personalized e-learning with the availability of applications through platform independent technology. At the core, semantic web services provide meaningful information and faster the way the business and people interact with each other.

REFERENCES

Ardagna, D. and P. Barbara, 2007. Adaptive service composition in flexible processes. IEEE T. Software Eng., 33(6): 369-384.

Aydın, O., K.C. Nihan and C. Ilyas, 2008. Automated web services composition with the event calculus. Proceeding of the 8th International Workshop on Engineering Societies in the Agents World. Springer Berlin, Heidelberg, pp: 142-157.

Berners-Lee, T., J. Hendler and O. Lassila, 2001. The Semantic Web. Scientific American, pp: 29-37.

Cardoso, J. and S. Amit, 2003. Semantic e-workflow composition. J. Intell. Inf. Syst., 21(3): 191-225.

Chinthaka, E., J. Ekanayake, D. Leake and B. Plale, 2009. CBR based workflow composition assistant. Proceeding of the World Conference on Services-I, pp: 352-355.

Chiu, D., S. Deshpande, G. Agrawal and L. Rongxing, 2009. A dynamic approach toward QoS-aware service workflow composition. Proceeding of the IEEE International Conference on Web Services, (ICWS, 2009).

Cohen, W., P. Ravikumar and S. Fienberg, 2003. A comparison of string distance metrics for name-matching tasks. Proceeding of the IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03), DBLP.

Fayed, G., D. Sameh, A. Hasna, A.M. Jihad, E.A. Samir and E. Hosam, 2006. E-learning model based on semantic web. Int. J. Comput. Inf. Sci., 4(2): 63-71.

Fei, X. and S. Lu, 2012. A data flow based scientific workflow composition framework. IEEE T. Serv. Comput., 5(1): 45-58.

Gil, Y., P.A. González-Calero, J. Kim, J. Moody and V. Ratnakar, 2011. A semantic framework for automatic generation of computational workflows using distributed data and component catalogues. J. Exp. Theor. Artif. In., 23.4(2011): 389-467.

Groenmo, R. and M.C. Jaeger, 2005. Model-driven semantic web service composition. Proceeding of the 12th Asia-Pacific Software Engineering Conference (APSEC, 2005), pp: 8.

Ibert, B.I., C. Evandro, S. Elvys and A. Pedro, 2008. Towards new generation of web-based educational systems: The convergence between artificial and human agents. IEEE Multidisciplinary Eng. Educ. Mag., 3(1): 17-24.

Kim, J., S. Marc and G. Yolanda, 2004. An intelligent assistant for interactive workflow composition. Proceedings of the 9th International Conference on Intelligent User Interfaces (IUI, 2004), pp: 125-131.

Klusch, M., 2008. Semantic Web Service Description. Chapter 3, CASCOM: Intelligent Service Coordination in the Semantic Web. ISBN: 978-3-7643-8575-0.

Klusch, M., A. Gerber and M. Schmidt, 2005. Semantic web service composition planning with owls-xplan. Proceedings of the AAAI Fall Symposium on Semantic Web and Agents. AAAI Press, Arlington VA, USA.

Laukkanen, M. and H. Heikki, 2004. Composing workflows of semantic web services. Ws So Ag Te., Springer, US, pp: 209-228.

Lécué, F., S. Eduardo and F.P. Luís, 2008. A framework for dynamic web services composition. Ws So Ag Te., Birkhäuser, Basel, 2: 59-75.

Majithia, S., D.W. Walker and W.A. Gray, 2004. Automated web service composition using semantic web technologies. Proceedings of the International Conference on Autonomic Computing, pp: 306-307.

Martin, D., M. Burstein, D. McDermott, S. McIlraith, M. Paolucci, K. Sycara, D.L. McGuinness, E. Sirin and N. Srinivasan, 2007. Bringing semantics to web services with OWL-S. World Wide Web, 10(3): 243-277.

McIlraith, S.A., T.C. Son and H. Zeng, 2001. Semantic Web Services. IEEE Intell. Syst., pp: 46-63.

Sohrabi, S. and S.A. McIlraith, 2010. Preference-based web service composition: A middle ground between execution and search. Proceeding of the 9th International Semantic Web Conference on the Semantic Web (ISWC, 2010). Springer, Berlin, Heidelberg, pp: 713-729.

W3C, 2004. OWL-S: Semantic Markup for Web Services. Technical Report. Retrieved from: http://www.w3.org/Submission/OWL-S/.

Zeng, L., A.H.H. Ngu, B. Benatallah, R. Podorozhny and H. Lei, 2008. Dynamic composition and optimization of web services. Distrib. Parallel. Dat., 24(1-3): 45-72.