Management of Web Services Communities, WSC System

Victor Popa[1], Liliana Constantinescu[1], Victoriana Popa[2], Maria Moise[3], Carmen Rotună[1]

1 National Institute for Research and Development in Informatics - ICI, Bucharest,
8-10 Averescu Avenue, Bucharest 1, România,
vpopa@ici.ro, lconst@ici.ro, karma_petcu28@yahoo.com

[2] Reply, 25 Cenisio Avenue, Milano, Italy,
vpopa.Consultant@ubm.unicredit.it

3 Romanian-American University,
1B Expoziției Avenue, Bucharest 1, România,
maria.moise@rau.ro

Abstract: The paper presents the system WSC, designed to provide modeling, organization, management and execution of Web services. The *Service* and *Community* ontologies are used for modeling of the Web services and Web services communities. Web Services are organized in semantic communities, in order to be easily accessed. Thus, the users of system navigate through the taxonomy of communities and invoke the target generic operations of communities. If an invoked operation requires some pre-operations to be executed before it, the system builds process diagram for these pre-operations. The user executes interactively each operation of diagram process using the interface provided by the system.

Keywords: community ontology, service ontology, registration of services with communities, process diagrams, invocation of community's generic operations, execution of Web service's methods.

1. Introduction

The number of Web services is increasing fast. The large scale and dynamics of these Web services hinders the understanding of their semantics and hence their management. To address this issue we propose the system WSC. Compared with other approaches which propose the ontologies for describing Web services (DAML-S, WSMF, OWL-S etc) the WSC system provides means for both semantic description of Web services and their organization into communities.

The system WSC is the result of sustained research work on finding innovative and effective technologies for modeling, managing and accessing Web services.

To achieve this goal WSC system took into consideration both existing industry standards for modeling services (WSDL, SOAP, UDDI, BPEL) as well as academic initiatives in the area of semantic Web (DAML-S, WSMF etc).

The following requirements need to be considered when designing the WSC system:

- organizing web services space in semantic communities in order to effectively discover services
- understanding the capabilities of semantic web services
- discovering and accessing Web services using semantic criteria
- simple and fast designation of the processes used in invoking Web services methods
- using standard BPEL for composing Web services and communities

To meet the requirements listed above, WSC system provides the following features:

- *Community* [1][2] ontology, used as templates for creating community descriptions
- *Service* [2] ontology, used as templates for creating service descriptions
- WSDL-S formalism, used to create descriptions for communities and services, according to the two ontology
- communities taxonomy management
- Web services registration with communities taxonomy
- Web services discovery based on semantic criteria

- mapping the generic operation invocations to Web service method invocations, based on quality attributes of Web services (response time, cost, reliability, confidentiality, etc.)
- automatic generation of process diagrams used to invoke generic operations of Web service communities
- interactive execution of process diagrams
- support for execution of generic operations invoked from BPEL processes

Due to its facilities, WSC system can be used in organizations and institutions that need to organize their own services in semantic communities, so that their business partners may get easy access to these services.

2. Semantic Communities, Communities Taxonomy

A community is specific for an interest domain and is created by a consortium of organizations (community provider) that wants to standardize Web services used in that domain. A community defines a generic operation collection for an area of interest, which Web services from that area have to implement. Community providers create descriptions of their communities using WSDL-S format (WSDL format extended with semantics) Semantic attributes used in the WSDL-S representations are defined in the *Community* ontology and they describe domain, generic operations, input/output messages, the eligibility requirements, order for invoking operations, conditions for chaining operations etc.

The descriptions of communities, in WSDL-S format, are stored in the community register (CR) that is modeled as a taxonomy of communities based on specialization relationship between community domains. Web service providers describe their services in WSDL-S format according to *Service* ontology and transmit this description to WSC system to be stored in the repository service. Descriptions of communities and Web services are used by the system or administrator to register Web services with the communities.

In order to access Web services registered with a given community (C), the user invokes a target operation (O) of C, using the graphic interface of WSC system. The system will generate, in a recursive way, the process diagram attached to operation O. The process diagram is a graphical tree having as root the generic operation O. The level 1 of tree contains the generic operations to be performed before the operation O. The level n of tree contains the generic operations to be performed before operations on level n-1.

The user executes interactively process diagram, starting with final nodes and finishing with the root node of tree. To execute a node operation, the user selects a Web service registered with the community C and then invokes the Web service method that corresponds to node operation. Execution state of each node of process diagram is stored in variables defined in the community C.

Not only the user can access the Web services registered with communities of taxonomy, BPEL processes can also invoke generic operations of a community C.

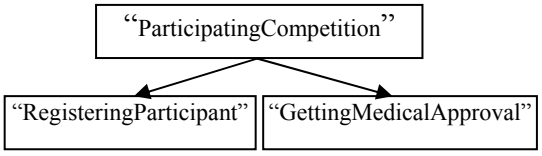
In this case, the system selects a Web service registered with community C and invokes the appropriate method of selected services.

Example

For example consider community C with the domain "Sporting Competition" and generic operations "RegisteringParticipant", "GettingMedicalApproval", "ParticipatingCompetition".

To attend a sporting event, a user connects to the WSC system, selects from communities taxonomy the community C and then invokes generic operation "ParticipatingCompetition" of community C.

As a result of this invocation the system generates the process diagram as below:



Then, the user interactively executes the generated diagram as follows:

1. user clicks on generic operation "RegisteringParticipant" providing his personal data in the input variable "*informatiiPersonale*"; if the operation ends successfully it will return the participant's ID in the output variable "*userId*"
2. user clicks on generic operation "GettingMedicalApproval" in order to obtain medical approval, input variable "*informatiiPersonale*" already contains the user's personal data; if operation ends successfully it will return the medical approval in the output variable "*avizMedical*"

```
<wsdl:operation name="RegisteringParticipant" >
    <function name="RegisteringParticipant">
      <synonyms>
        <synonym name="inscriere-in-competitie" />
      </synonyms>
    </function>
  <relations>
  </relations>
 </wsdl:operation>
---------------------------------------------------------------------------------------------------------------------------------
   <wsdl:operation name="ParticipatingCompetition"  goal="yes">
<function name="ParticipatingCompetition">
      <synonyms>
      <synonym name="obtinereNumarTricou" />
      </synonyms>
    </function>
<PreConditions>  virsta > 60, localitate="Bucuresti"  </PreConditions>
  <relations>
      <precedesor name="EliberareAvizMedical">  aviz.avizare = true   </precedesor>
      <precedesor name="InregistrareParticipant">  UserId # empty  </precedesor>
  </relations>
   </wsdl:operation>
---------------------------------------------------------------------------------------------------------------------------------
   <wsdl:operation name="GettingMedicalApproval" >
<function name=" GettingMedicalApproval">
   <synonyms>
   <synonym name="aprobare medicala" />
   </synonyms>
   </function>
   <relations>
  </relations>
   </wsdl:operation>
---------------------------------------------------------------------------------------------------------------------------------
<wsdl:service name="C" provider="municipiu Bucuresti" documentation="">
<domain name=Sporting Competition>
   <synonyms>
   <synoym name="recreare sportiva"/>
   </synonyms>
 </domain>
```

Figure 1. Description of Web services community for the given example

3. user clicks the generic operation "ParticipatingCompetition" to obtain his T-shirt number for participation at competition; input variables "*userId*" and "*avizMedical*" already contain the necessary information if the first operations ended successfully. If operation execution ends successfully it will return the T-shirt number in output variable "*nrtricou*"

Execution of each operation in the above diagram involves checking the eligibility and termination conditions of the operation.

Description of community C from given example, using WSDL-S format, is illustrated in the Figure 1.

The above description contains concepts defined in *Community* ontology: function, synonym, domain, relation, precondition, predecessor, goal etc.

The element *<function>* and synonyms included specify operation category, helping user to understand the operation semantics.

The element *<domain>* and synonyms included specify community domain, helping users to understand the semantics of domain of community.

The element <*preconditions*> specifies the eligibility conditions for an operation to be invoked. Specified conditions include the input variables of the operation or variables defined in the user profile.

The element <*relation*> specifies for each operation, the predecessor operations and their termination conditions. The termination conditions will include output variables of the operation.

Input/output variable of generic operations share the same memory area (global variables) if they have the same identifier, the same data type, and the same semantics. For example variable "userId" defined in the operation "RegisteringParticipant" and "userId" variable defined in the operation "ParticipatingCompetition" shares the same memory area, because they have the same

```
<wsdl:message name="RegisteringParticipant SoapIn">
  <wsdl:part name="infPersonale" type="tns:InformatiiPersonale" >
    <businessRole name="datePersonale"/>
  </wsdl:part>
 </wsdl:message>
 <wsdl:message name="RegisteringParticipant SoapOut">
  <wsdl:part name="userId" type="tns:UserId" >
    <businessRole name="identificareutilizator"/>
  </wsdl:part>
 </wsdl:message>
-----------------------------------------------------------------------------------------------------------------------
 <wsdl:message name="ParticipatingCompetition SoapIn">
  <wsdl:part name="userId" type="tns:UserId" >
    <businessRole name="identificareutilizator"/>
  </wsdl:part>
  <wsdl:part name="avizMedical" type="tns:AvizMedical" >
     <businessRole name="staresanatate"/>
  </wsdl:part>
 </wsdl:message>

 <wsdl:message name="ParticipatingCompetition SoapOut">
  <wsdl:part name="nrtricou" type="tns:Nrtricou" >
     <businessRole name="tricouparticipare"/>
  </wsdl:part>
 </wsdl:message>
-----------------------------------------------------------------------------------------------------------------------
 <wsdl:message name="GettingMedicalApproval SoapIn">
  <wsdl:part name="infPersonale" type="tns:InformatiiPersonale" >
     <businessRole name="datePersonale"/>
  </wsdl:part>
 </wsdl:message>
 <wsdl:message name="GettingMedicalApproval SoapIn">
  <wsdl:part name="avizMedical" type="tns:AvizMedical" >
    <businessRole name="staresanatate"/>
  </wsdl:part>
 </wsdl:message>
```

Figure 2. Description of input/output variables for generic operations

The attribute *goal* specifies whether an operation is a target operation or a helping operation. In the given example only operation "ParticipatingCompetition" is a target operation; the other operations are helping operations.

Description of input/output variable for the generic operations from given example is illustrated in the Figure 2.

Note that each input/output message is composed of several parts (variables), each part having attached a data type and a *businessRole* specifying semantics of variable.

identifier, the same data type and the same businessRole.

Using global variables in describing communities allows interconnection among communities operations. The interactive execution of generic operations involves some editing of the input variable or viewing of the output variables.

The system WCS uses data type of each input/output variable to generate automatically editing and viewing controls.

Data types for input/output variables used in the given example are defined in the Figure 3, using the XMLSchema formalism.

```
<s:complexType name="InformatiiPersonale">
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="nume" type="s:string" />
        <s:element minOccurs="1" maxOccurs="1" form="unqualified" name="virsta" type="s:int" />
        <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="localitate" type="s:string" />
      </s:sequence>
    </s:complexType>
-------------------------------------------------------------------------------------------------------------------
<s:complexType name="UserId">
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="userId" type="s:string" />
      </s:sequence>
    </s:complexType>
 -------------------------------------------------------------------------------------------------------------------
 <s:complexType name="AvizMedical">
      <s:complexContent mixed="false">
        <s:extension base="tns:InformatiiPersonale">
          <s:sequence>
     <s:element minOccurs="1" maxOccurs="1" form="unqualified" name="avizare" type="s:boolean" />
          </s:sequence>
        </s:extension>
      </s:complexContent>
    </s:complexType>
  -------------------------------------------------------------------------------------------------------------------
<s:complexType name="Nrtricou">
      <s:sequence>
        <s:element minOccurs="1" maxOccurs="1" form="unqualified" name="nrTricou" type="s:int" />
      </s:sequence>
    </s:complexType>
```

Figure 3. Data types used in the given example

```
  [SoapRpcService()]
public class Sporting Competition : WebService
{
   [WebMethod]
   public UserId RegisteringParticipant (InformatiiPersonale infPersonale)

   [WebMethod]
   public Nrtricou ParticipatingCompetition (UserId userId, AvizMedical avizMedical)

   [WebMethod]
   public AvizMedical GettingMedicalApproval (InformatiiPersonale infPersonale)
 }
```

Figure 4.

Note

Data types used in WSC are only user types. If you want to use a system type, it must be incorporated into a user type.

For example, if the user needs an array of medical approval, then the system type "[]" will be incorporated into class "AvizMedicalArray" as below:

```
class  AvizMedicalArray
{
   AvizMedical []   items;
}
```

Manual creation of the file containing community description for given example is difficult. To ease this task we can use the following procedure:

1. create the appropriate class having the same name as community's domain and the methods identical with community operations – Figure 4.
2. create the WSDL file format for this class using WSDL.exe instrument of .Net platform
3. manually extend WSDL file format with semantic attributes specified in Community ontology (domain, function, synonym, etc.)
4. modify variables name, created by the WSL tool, if needed (only output variables)

The value of "URL" attribute, from the obtained file, has to refer the community site. The community site allows execution of

community operations when they are invoked from BPEL processes.

InsertCommunity that displays the insertion panel, as is illustrated in Figure 5.

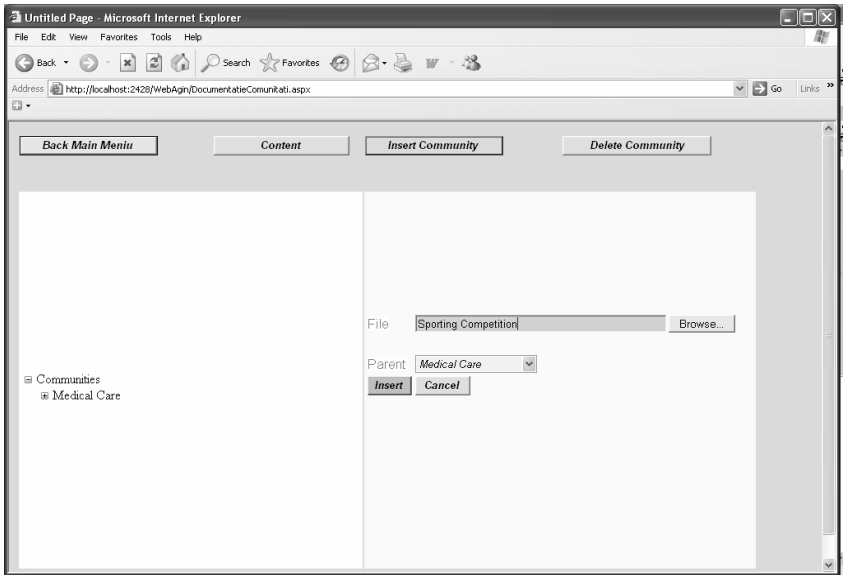


Figure 5. Insertion of a new community description in taxonomy

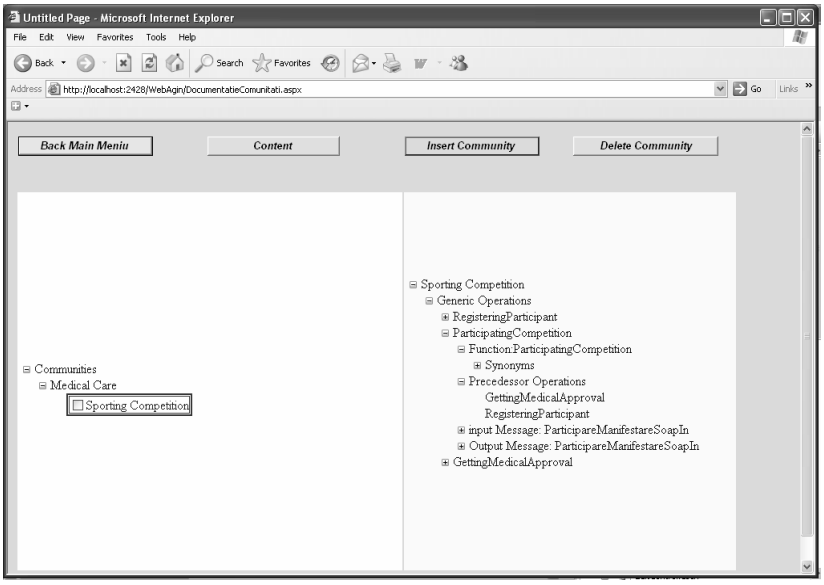


Figure 6. Visualization of the community's content

The file, thus obtained, will be parsed by WSC system to record the community description in the taxonomy of communities.

1.1 User interface for managing communities

WSC system provides communities management tool for insertion, deletion, navigation and display of community descriptions. [6] [7]

To insert a community description into the taxonomy, the user presses the button

The user specifies the place in taxonomy where new description will be inserted, by selecting from *Parent* list the parent community, then enters the WSDL-S file address in *File* box, which contains the description of community to be inserted, or uses *Browser* button to select the file containing community description. Actual insertion is done by pressing *Insert* button; the system will parse the specified file, will make a series of validations on the syntax and semantics of information and will build data structures necessary to record the new community description.

In order to view the content of a community description, the user expands Communities tree, then clicks the label of community to select the community, then clicks Content button to list the content. The Figure 6 visualizes the content of the community given in the example.

Deleting a community from communities taxonomy can be done only by its supplier. To delete a community from the taxonomy, the supplier marks it by clicking community square, and then presses the *DeleteCommunity* button.

3. Modeling and Registering Web Services

Services implement generic operations defined in one or more communities. Service providers describe their services in WSDL-S files according to *Service* ontology. Semantic attributes defined in *Service* ontology are similar to those defined by the *Community* ontology with the following exceptions:

- Unlike Community ontology where all generic operations inherited the community domain, the *Service* ontology requires as every method of a Web service owns a domain, because a service can implement generic operations for several communities.
- The *Service* ontology does not define the eligibility conditions, termination conditions and sequence relationships between methods, but defines quality attributes for methods: cost, response time, reliability, confidentiality.

Using *Service* ontology, service providers create descriptions of services implemented by them and then send them to register with the appropriate communities. The registration process of services with semantic communities includes the following steps:

1. compare syntactic attributes of community operations with syntactic attributes of service methods (identifiers, data types etc.)
2. compare semantic attributes of community operations with semantic attributes of the methods services (domain, function, synonym, businessRole etc.)

As a result of registration procedure, a service (S) will be registered with a semantic community (C) if for some generic operations(O) belonging to community C, was found some methods (M) belonging to the service S, so that the description of M fits the description of O.

A Web service can be registered with one or several communities. [6] [7]

To illustrate the services modeling and registration method, we consider a Web service that implements all generic operations of community given in the above example. The WSDL-S file containing description of service methods is listed in Figure 7.

The description of input/output variables for each service method is done inside the <*element*> tag, corresponding to the method, not inside a message.

For example, input variables “userId” and “avizMedical” of “ParticipatingCompetition” method are defined in Figure 8.

For automating the WSDL-S file creation containing the service description we can use the following procedure:

1. create the appropriate service class having same methods as the service description – Figure 9.
2. create the WSDL file for this class, using WSDL.exe tool provided by .Net platform
3. extend the created WSDL file with the semantic attributes specified in the *Service* ontology (domain, function, synonym etc.)
4. change the name of variables in the created file, if it is necessary

```
<wsdl:operation name="RegisteringParticipant" >
    <domain name="Sporting Competition>
     <synonyms>
       <synoym name="recreare sportiva"/>
     </synonyms>
    </domain>

    <function name="RegisteringParticipant">
     <synonyms>
       <synonym name="inscriere-in-competitie" />
     </synonyms>
    </function>
    <ServiceQuality  cost="0"  responseTime="30"   reliability="0.7" confidentiality="0" />
  </wsdl:operation>
-------------------------------------------------------------------------------------------------------------------------------
   <wsdl:operation name="ParticipatingCompetition">
    <domain name="Sporting Competition">
     <synonyms>
       <synoym name="recreare sportiva"/>
     </synonyms>
    </domain>
    <function name="ParticipatingCompetition">
     <synonyms>
       <synonym name="obtinereNumarTricou" />
     </synonyms>
    </function>
    <ServiceQuality  cost="0"  responseTime="30"   reliability="0.8" confidentiality="0" />
  </wsdl:operation>
-------------------------------------------------------------------------------------------------------------------------------
   <wsdl:operation name="GettingMedicalApproval">
    <domain name="Sporting Competition">
     <synonyms>
       <synoym name="recreare sportiva"/>
     </synonyms>
    </domain>

    <function name="GettingMedicalApproval">
     <synonyms>
       <synonym name="aprobare medicala" />
     </synonyms>
    </function>
    <ServiceQuality  cost="0"  responseTime="20"   reliability="0.9" confidentiality="0.7" />
  </wsdl:operation>
-------------------------------------------------------------------------------------
<wsdl:service name="Competition-1" provider="sector1" documentation="" />
```

Figure 7. Description of the service's methods

```
<s:element name="ParticipatingCompetition">
     <s:complexType>
       <s:sequence>
         <s:element minOccurs="0" maxOccurs="1" name="userId"
                 type="tns:UserId" businessRole="identificator" />
        <s:element minOccurs="0" maxOccurs="1" name="avizMedical"
                 type="tns:AvizMedical" businessRole="aviz" />
       </s:sequence>
     </s:complexType>
   </s:element>
```

Figure 8. Description of service's variables

```
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class Competition-1 : WebService
{
   [WebMethod]
   public UserId RegisteringParticipant (InformatiiPersonale infPersonale)

   [WebMethod]
          public Nrtricou ParticipatingCompetition (UserId userId, AvizMedical avizMedical)

   [WebMethod]
   public AvizMedical GettingMedicalApproval (InformatiiPersonale infPersonale)
}
```

Figure 9.

The file thus obtained, will be parsed by the WSC system to store the service description in repository and register the service with communities from taxonomy.

order to parse, validate and insert the new description. The system will assign a unique identifier for each inserted description. Figure 10 shows the insertion panel.

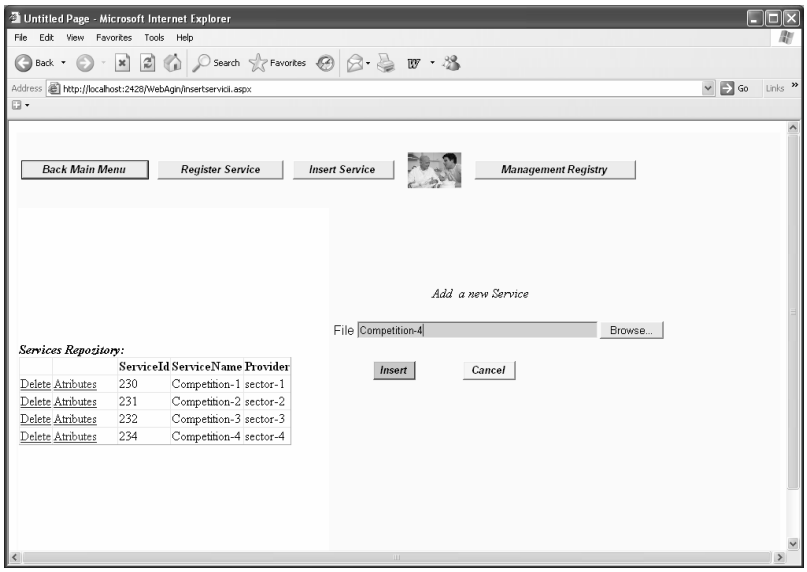


Figure 10. Insert a new Web service description

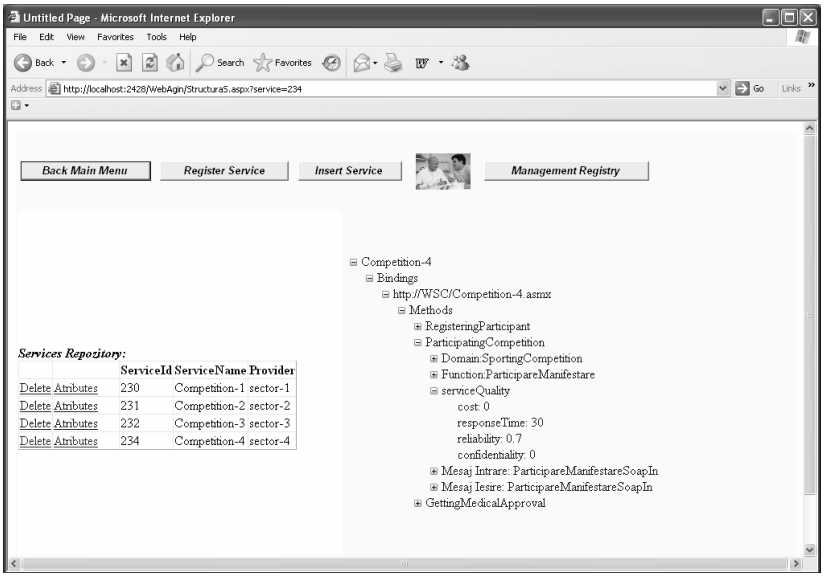


Figure 11. Visualization of service description attributes

2.1 User interface for services management

WSC system provides services management tool for inserting, deleting and viewing service descriptions. To display the insertion panel, the user clicks the *Insert Service* button. The user clicks the *Browser* button on the panel for selecting the WSDL-S file containing the description to be inserted. Then, the user presses the *Insert* button in

To view attributes of a service description, the user clicks Attributes button corresponding to service description. Figure 11 shows attributes of services Competition-4.

To delete a service description from repository the user clicks the *Delete* button corresponding to service description.

Once inserted in the repository, a service description will be used to register it with the communities from taxonomy. The

administrator can manually register the service with a community from taxonomy. For this purpose the administrator presses the *Register Service* button to display the registration panel – Figure 12.

Then, the administrator selects from the list *Services* the service (W) to be registered, and from *Communities* list the community (C) with which the service W will be registered.

click on it). Box Operation will display selected operation.

– The administrator selects a method (M) fro the service W, that fits operation O (doing click on it). Box Method will display the selected method.

– The administrator clicks Register button to create the pair (O, M) in registry.

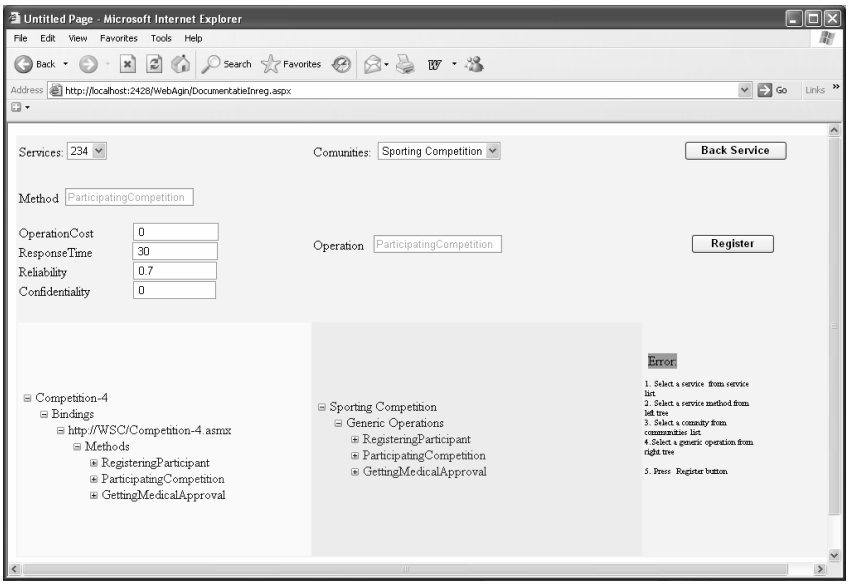


Figure 12. Registration of Web services with the communities

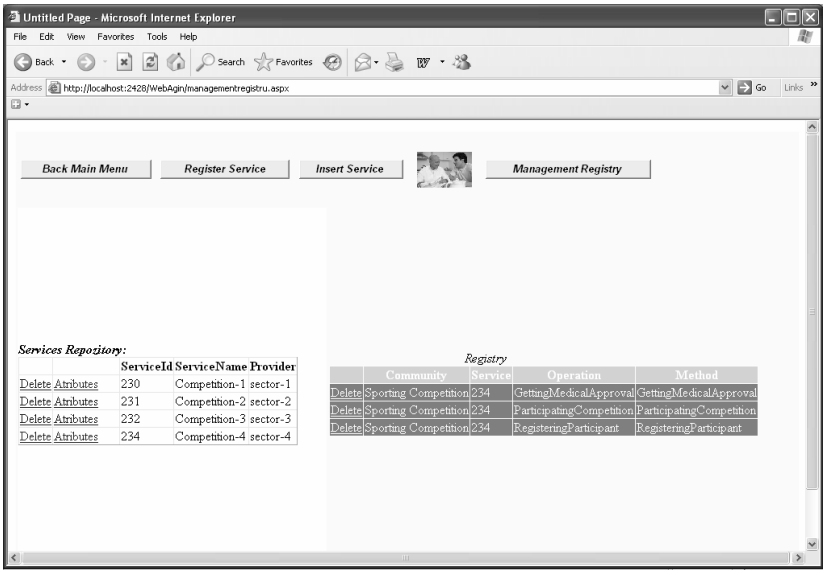


Figure 13. Visualization and deleting the registry

Next activity consists in mapping each generic operation of community C with the corresponding service method of service W, as follows:

– The administrator selects a generic operation (O) from community C (doing

The administrator can list or delete pairs from registry by clicking the Management Registry button. The grid containing existing entries from registry is displayed. To remove a pairs from registry the user clicks the *Delete* button corresponding to that pair – Figure 13.

4. Web Service Execution

Execution of Web services registered with semantic communities is done as follows:

- the user selects a community from taxonomy
- the user invokes one of the target operation belonging to selected community
- the system recursively generates process diagram attached to operation invoked; the diagram includes all previous operations to be performed before the operation invoked
- the user interactively executes every node of the process diagram, starting with final nodes and finishing with the root node of the diagram as follows:
  - selects a Web service from the services registered with the selected community, taking in account quality attributes of services
  - initializes the input variables of the generic operation contained in the current node
  - executes the service method corresponding to the generic operation contained in current node
  - analyze output variables returned by the execution of method.

4.1 User interface for Web services execution

The user browses the taxonomy of communities, analyzing their semantic attributes for understanding semantic capabilities of communities. In the end he selects a community of interest and invokes a target operation from selected community. The system generates the process diagram attached to the invoked operation, the diagram is a tree having invoked operation as root , on the first level diagram includes all predecessors of invoked operation, on second level the diagram includes the predecessors of each operation belonging to the first level, and so on. [3], [4], [5]

The WSC system provides a user interface for navigation, selection and process diagram generation as is illustrate in Figure 14.

The user selected the community "Sporting Competition" as current community, clicking its label in taxonomy, then he listed the content of selected community by clicking the Content button, then he selected the target operation "ParticipatingCompetition" clicking its label in the content; the text box *SelectedOperation* will contain the selected operation.

Finally, the user clicked the *Generate Process Diagram* button to generate process diagram for invoked operation.

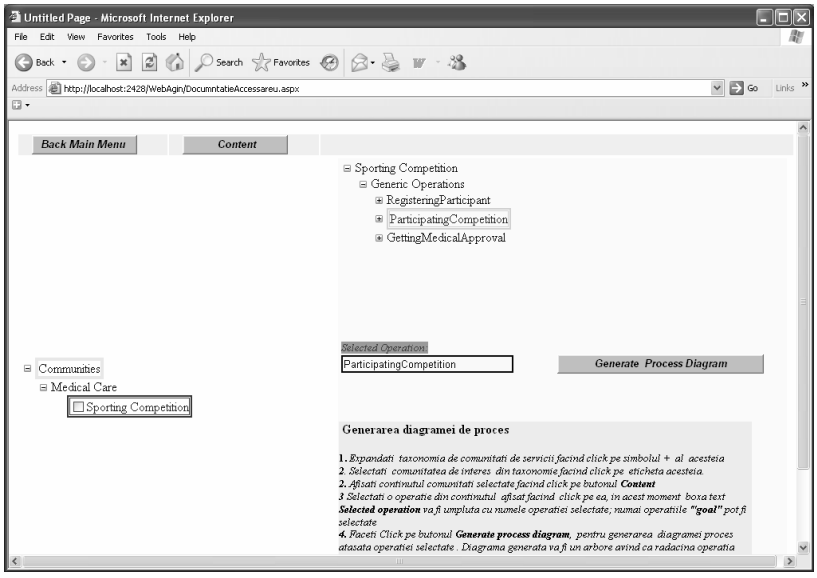


Figure 14. User interface for generating process diagram

As result, the system generated the process diagram and displayed the execution panel, as shown in Figure 15.

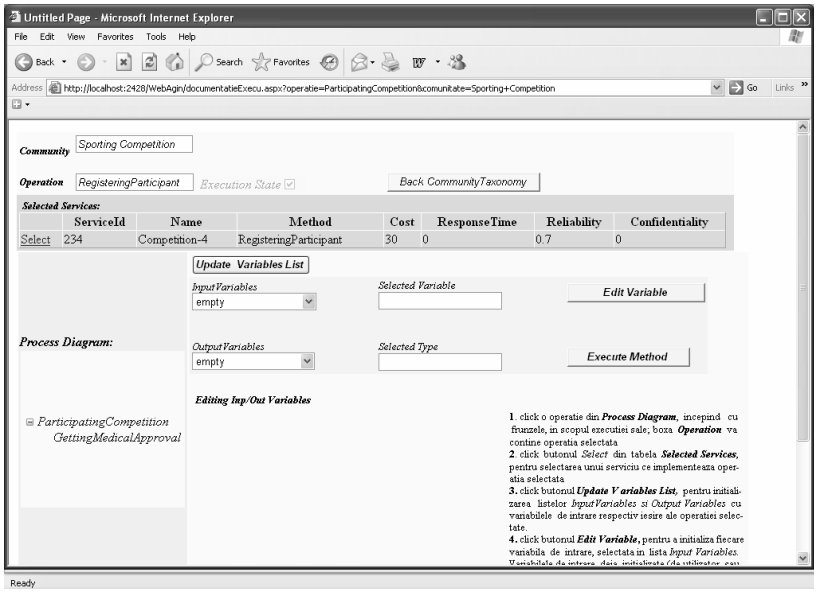


Figure 15. Execution panel for process diagrams

In our example, the generated process diagram has the root the operation "ParticipatingCompetition" and the first level contain the operations "GettingMedicalApproval" and "RegisteringParticipant".

To execute the generated process diagram, the user will execute each operation in the process diagram by: initializing the input variables of operation, selecting a service registered with the current community, executing the appropriate method of selected service, visualizing the output variables returned by executed method.

So, to execute the process diagram for our example, the user will perform:

1. Click on the operation "RegisteringParticipant" from diagram to select it; the box *Operation* will be filled automatically by the system with the selected operation.
2. Initialize the input variables of selected operation; for this purpose, the user selects from the list *InputVariables* the variable *infPersonale*. If it doesn't yet exist in the list, the *Update Variable List* button will be pushed. The user clicks *EditVariable* button to display the editing form. The editing form is automatically built by,system according to the data type of variable; for an uninitialized variable, the editing form contains value –1 for all its fields. After the user enters personal data in the fields of form, he clicks *Update* button on form to initialize the variable with entered values.
3. Select a service registered with the current community by clicking the Select button on the table "*Selected Services*". The table contains for each method of a Web server registered with the current community the quality attributes of service.
4. Execute the appropriate method of selected service; the user presses the button *Execute Method* for executing the method of selected Web service. The system will perform the following steps:

   assessment of eligibility conditions for operation "RegisteredParticipant";

   invocation of service method;

   assessment of termination conditions, at the end of method invocation;

   filling the output variables with returned values
5. Visualize the output variables returned by method invocation; the user selects the output variable *userId* from *OutputVariable* list, then click the *Edit Variable* button to visualize it.

If the eligibility conditions or termination conditions are not met, checkbox ExecutionState will become unmarked and diagram execution cannot continue.

After executing "RegisteredParticipant" operation, the system will delete it from process diagram and the user will go through the same process for executing the rest of diagram nodes.

5. Composition of Communities Using the Language BPEL

The description of a community contains the URL address for the website that implements the generic operations of that community.

The website redirects the invocation of each generic operation of community towards the invocation of a method of service registered with the community. The selection of a Web service from the services registered with the community, is done in accordance with website policy (default is selected service with lowest cost).

A website for a community also implements following methods:

- register for registering a service with the community
- selectService for selecting a service using a selection criterion

BPEL processes can use the community websites as partners, for this is necessary to know the WSDL files of the community websites. The order of invocation of generic operations for a given community, within a BPEL process, must comply with the order relation defined in the given community.

6. Conclusions

The WSC system was developed for organization, management and execution of Web services. The Web services are organized in semantic communities, all services registered with a given community have to implement all generic operations of given community. To access Web services, the users navigate through the taxonomy of communities, visualize the content of communities in order to understand the capabilities of service registered with them, and finally invoke the generic operations of communities.

Invocation of a generic operation can require execution of an entire process of pre-operations, in this case the WSC system will generate process diagram for invoked operation and enable the interactive execution of this diagram.

Due to its facilities, the WSC system could be used to manage the Web services delivered through e-Romania portal[8].

REFERENCES

1. BRAHIM, M., **A Dynamic Foundational Architecture for Semantic Web Services**, Distributed and Parallel Databases, 17, 2005, pp. 179–206.

2. POPA, V., L. CONSTANTINESCU, C. ROTUNĂ, **Senior Citizen Service Management using WebAging System**, Studies in Informatics and Control, Volume 18, Issue 4, 2009.

3. RAN, S., **A Model for Web Services Discovery with QoS**, SIGecom Exchanges, vol. 4, no. 1, 2003.

4. MOISE, M., V. POPA, M. ZINGALE, L. CONSTANTINESCU, AL. PÎRJAN, **WebAgeing - A Flexible System for Personalized Accessing of Services for Ageing Population**, International Journal of Computers, Communications & Control, Supplementary Issue – Proceedings of ICCCC 2008, Vol. III, 2008, www.journal.univagora.ro.

5. MOISE, M., V. POPA, **E-learning System Architecture for Personalized Publication and Access of Learning Resources, Using the Approach Based on Semantic Metadata**, in Metalurgia International Journal Vol. XV, Special

Issue, nr. 3, 2010, ISSN 1582-2214, pp. 207-210.

6. MEDJAHED, B., A. BOUGUETTAYA, A. ELMAGARMID. **Composing Web Services on the Semantic Web**. The VLDB Journal, Special Issue on the Semantic Web, September 2003.

7. OUZZANI, M., B. BOUGUETTAYA, **Efficient Access to Web Services**. IEEE Internet Computing, 37(3), March 2004.

8. SANDU, G., **Achieving Modernization through ICT and New Technologies**, Romanian IT&C Directory - The 10th Edition, March 2010.