A Template-Based Semi-Automatic Web Services Composition Framework

Case Study of Smart Home Management

Abdaladhem Albreshne
PhD Student
Computer Science Department
University of Fribourg, Switzerland
abdaladhem.albreshne@unifr.ch

Jacques Pasquier
PhD Supervisor
Computer Science Department
University of Fribourg, Switzerland
jacques.pasquier@unifr.ch

***Abstract*— Recently, a range of industry languages and frameworks solutions have been realized to enable web services composition. Among these, we find the Business Process Execution Language for Web Services (BPEL4WS) [1] and the Semantic Markup Language for Web Services (OWL-S) [2]. But there is still a need to propose a system which offers more flexibility to compose and invoke services and which reinforces the human-computer collaboration paradigm. This work aims to explore the challenges of the current web services composition solutions through a case study realized in the field of smart homes environment and to propose a new approach based on generic processes and semantic description in order to discover, compose and invoke services in a changing environment. In other words, it provides an assistance mechanism for the semi-automatic composition of services where the composition is gradually generated by using a declarative generic template process.**

Keywords- Web Services; Web Services Composition; Ontology; Semantic Web; BPEL4WS; OWL-S; WSDL; Smart Homes

I. INTRODUCTION

Recently, a range of industry languages and frameworks solutions have been realized to enable web services composition. Among these, the Business Process Execution Language for Web Services (BPEL4WS) [1] is probably the most prominent. It provides a language for web services composition where the flow of processes and the bindings between services are known before the execution. Unfortunately, BPEL4WS does not fully support dynamic reconfiguration of a process, where for example a given service must be replaced by another one with the same functionalities, but a different binding type or a different set of methods signatures. This prevents the user from selecting another service afterwards, when many different services are available to provide similar functional components.

Semantic web services composition constitutes another aspect on which the Web community focuses. In order to be able to describe the services, semantic web languages like the Ontology Language for Web Services (OWL-S) [2], and the Web Services Modeling Ontology (WSMO) [3] have been proposed. They introduce an additional level of abstraction. Instead of a syntactic description of a web service, a declarative description of the service's functionalities is given. Semantic services description protects the programmer from the complexity of the implementation and thus simplifies the completion of complex tasks using these services. Semantic could also add machine interpretable information to services' content in order to provide intelligent access to distributed web services. We compare these two approaches and discuss their solutions and limitations regarding the problems of modeling, composing and executing Web services. Additionally, we discuss a new approach to overcome these limitations.

II. MOTIVATING EXAMPLE

A. *Home Energy Saving Scenario*

In this section, we explore the challenges of the current web services composition through a case study realized in the field of home environment. A smart home is a home equipped with diverse networked sensors such as temperature and movement detectors and devices such as alarms, heating systems, air-conditioners, doors/windows/lights controllers, etc. The interested reader is referred to [4] [5] [6] for a good introduction and a thorough discussion in smart homes themselves, for which an in-depth presentation is out of the scope of this paper.

Jane controls her home environment in a way that allows her to save energy and to adapt her environment to her habits and living conditions. Jane knows that a well-planned home energy control system can reduce the total energy consumption of the home. Now, let us imagine that Jane's house is equipped with the following web enabled devices and sensors:

- Light switches - wall-mounted light switches typically using WSN (Wireless Sensor Network) control technology.
- Thermostats - heating/cooling can be adjusted according to the program (for example, to adjust to comfort settings upon arrival at home).

- Door/window controllers - used to open and close doors and windows.
- Temperature sensors - used to measure the internal and external temperature.
- Curtains controllers - used to open and close curtains.
- Movement detection sensor - used to detect objects' movements.

To understand how these services can be orchestrated to save energy, we define a simplified business process for a home energy saving scenario (see Fig. 1). The process invokes sensor services asynchronously in order to wait for the incoming events. We assume that sensors have the capability to provide asynchronous service invocations. Each time the process receives an event, it starts to execute the necessary tasks in order to save energy depending on the new changing context. Then, the process is waiting for a new event to occur.

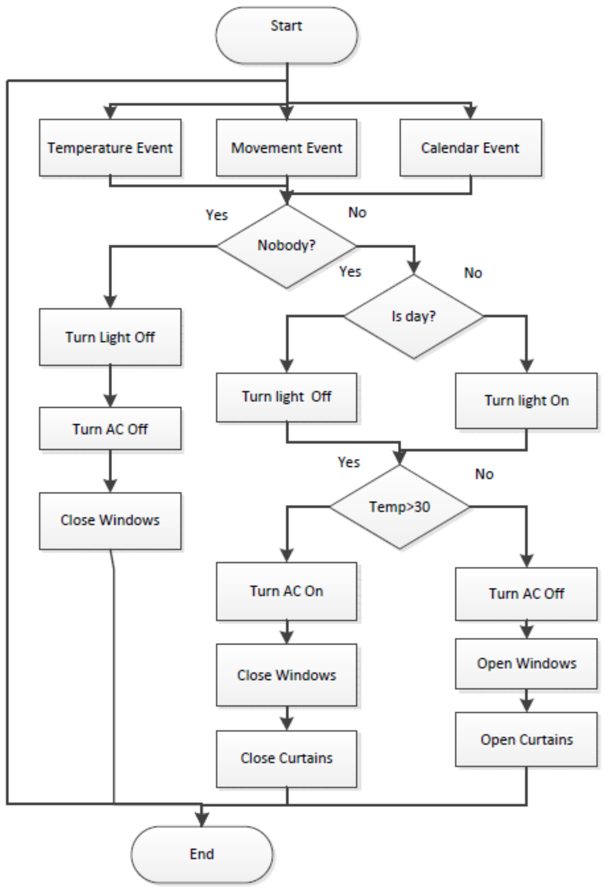


Figure 1. Home Energy Saving Scenario

B. *Web Services Composition Requirements and Challenges*

This scheme demonstrates the features necessary for a home energy saving process which are as follows:

- The process should offer *structured activities* (while, if, pick, flow, etc.), which allow for creating complex home control scenarios.
- Since the process is designed to control services and devices which are not known at design time, there is a need for a mechanism that allows for *discovering* services and for involving them into the process.
- Providing *semantic description* for the involved services helps users to configure the environments. In the home energy saving context, the system needs to understand and interpret the environment in order to take decisions about which services can be involved. Furthermore, it must enable interacting with the user in order to configure the process. Semantic description of involved services facilitates the service discovery and configuration by giving meaning to services parameters, inputs, outputs, operation names and non-functional attributes, such as service name, service type or service location.
- Processes should support *events messages* (an event represents the occurrence of something interesting). Event handlers allow processes to respond to the expiration of timers or to events by executing a specified set of operations independently from the rest of the process. Event handlers help a process to react to changing environments. In the home scenario, movement detection or a temperature change can happen at any moment. Providing an event handler mechanism makes the process more aware of the context.
- The final user should be capable of *interacting* with the system to reconfigure the process according to his needs. For example, in a home energy saving scenario, the system needs to know from the user what the interior comfort home temperature is when he is at home or what is the preferred temperature for each air conditioned room.
- The composition implementation should be hidden from the user. This might imply that an *abstract generic process template* is defined to achieve the composition goal. Raising the level of abstraction of composing web services facilitates normal user's interaction with the system.
- Since there can be many instances of the same process running at the same time, *message correlation elements* or a similar mechanism need to provide a way to decide which process instance a specific message is sent for.
- Finally, *Asynchronous invocations* are extremely useful for home control environments in which a process, such as an energy saving one, must react to many events over a long period of time.

III. FIRST SOLUTION: WSDL+BPEL4WS

The role of BPEL4WS is to define a new web service by composing a set of existing services through a process-integration type mechanism with control language constructs. It interacts with external partner services through a Web Service Description Language Interface (WSDL) [7]. A BPEL4WS process defines the order in which involved Web services are composed. BPEL4WS allows for describing conditional activities [1] [8]. An invocation of a Web service can for example rely on the result of another web service's invocation. With BPEL4WS, it is possible to create loops, declare variables, copy and assign values as well as to use fault handlers. Additionally, BPEL4WS offers the possibility of asynchronous invocation and supports event messages.

A. BPEL4WS Characteristics

In the following, we briefly describe the main elements of a BPEL process as defined in [1]:

1) Partner Links Elements define the interaction of participating services with the process.

2) Structured Activities are provided by the <*sequence*>, the <*while*>, the <*if*> ,the <*flow*> (for executing activities in parallel) and the <*pick*> constructs.

3) Primitive Activities are provided by the <*invoke*>, the <*receive*>, the <*reply*>, the <*assign*>, and <*throw*> constructs.

4) Variable Elements allow for declaring variables in order to receive, manipulate, and send data.

5) Fault Handlers determine the activity which the process has to perform when an error occurs.

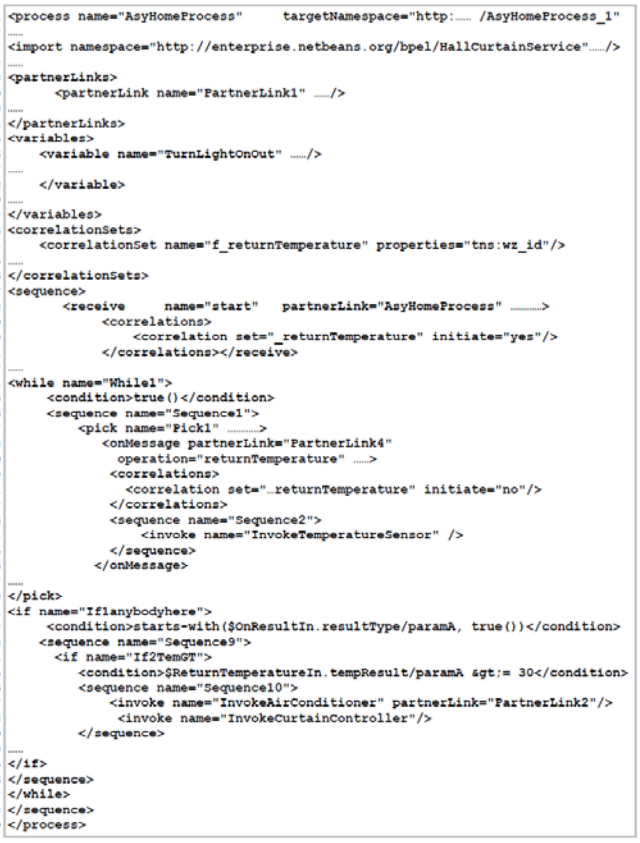
6) Correlation Sets enable several processes to interact in stateful conversations.

7) Event Handlers allow the processes to respond to events.

B. Home Energy Saving Scenario Implementation Using BPEL4WS

Let us solve the services composition problem of the energy saving scenario using the BPEL4WS solution. Fig. 2 shows an extract of the code of the process. Since the BPEL4WS process communicates with other Web services, it depends on the WSDL descriptions of the Web services invoked by the process. In the following, we briefly comment the code of Fig. 2:

- *Namespaces*: lines 1-20 define the target namespace to access the partners' WSDL description files.
- *Import:* lines 21-37 import the WSDL files for all service providers and the process. Services providers must be known a priori.
- *Partner Links*: lines 38-54 specify the partner links which define different partners that interact with the BPEL process. Each partner link is related to a specific *partnerLinkType* that characterizes it.
- *Variables*: lines 55-82 declare the variables which are used to store reformat and transform messages (e.g. temperature, movement detection state, time).
- *Correlation Sets*: lines 83-86 declare the correlations sets which enable partners to interact with the process in stateful conversations. Because there can be many instances of the process running at the same time, message correlation elements provide a way to decide which process instance a specific message is sent for.
- *Process logic definition*: lines 87-259 specify the process main body which defines the order in which the partner Web services are invoked. The work flow structure is divided into two groups: events and services. An event represents the occurrence of a sensor while a service represents the interaction process in response to that event. The first group consists of a pick activity which executes only the first event handler fired. BPEL provides an <*onMessage*> element for event declaration. When an event is fired, the process then executes the second group of structures through different scenarios depending on the events' received messages in order to save home energy. Structured and basic activities (*invoke, flow, if, while,* etc.), are implemented in order to define a complex scenario. Finally, the process is waiting through a while loop statement for a new event to occur.



```
  1 <process name="AsyHomeProcess"     targetNamespace="http:..... /AsyHomeProcess_1"
  2 ......
 21 <import namespace="http://enterprise.netbeans.org/bpel/HallCurtainService"...../>
 22 ......
 38 <partnerLinks>
 39       <partnerLink name="PartnerLink1" ...../>
 40 ......
 54 </partnerLinks>
 55 <variables>
 56     <variable name="TurnLightOnOut" ...../>
 57 ......
 58     </variable>
 59 ......
 82 </variables>
 83 <correlationSets>
 84     <correlationSet name="f_returnTemperature" properties="tns:wz_id"/>
 85 ......
 86 </correlationSets>
 87 <sequence>
 88        <receive     name="start"   partnerLink="AsyHomeProcess" ..........>
 89             <correlations>
 90                 <correlation set="_returnTemperature" initiate="yes"/>
 91             </correlations></receive>
 92 ......
184 <while name="While1">
185      <condition>true()</condition>
186      <sequence name="Sequence1">
187          <pick name="Pick1" ..........>
188             <onMessage partnerLink="PartnerLink4"
189               operation="returnTemperature" .....>
190              <correlations>
191                <correlation set="...returnTemperature" initiate="no"/>
192              </correlations>
193              <sequence name="Sequence2">
194                  <invoke name="InvokeTemperatureSensor" />
195              </sequence>
196            </onMessage>
197 ......
210 </pick>
211 <if name="Iflanybodyhere">
212      <condition>starts-with($OnResultIn.resultType/paramA, true())</condition>
213     <sequence name="Sequence9">
214       <if name="If2TemGT">
215          <condition>$ReturnTemperatureIn.tempResult/paramA &gt;= 30</condition>
216          <sequence name="Sequence10">
217             <invoke name="InvokeAirConditioner" partnerLink="PartnerLink2"/>
218              <invoke name="InvokeCurtainController"/>
219          </sequence>
220 ......
255 </if>
256 </sequence>
257 </while>
258 </sequence>
259 </process>
```

Figure 2. Code Extract of the Home Energy Saving BPEL Process

IV. SECOND SOLUTION: SEMANTIC WEB (OWL-S)

Most web services provide isolated functions and lack a real capacity to automatically collaborate among each other. To help solving this problem, there is a necessity to use semantic technology. Semantic and ontology facilitate interpreting web services by providing semantic awareness and services filtering capabilities.

Semantic Web Services aim to improve automatic services composition, service discovery and service invocation. In consequence, this assures interoperability and collaboration between different business processes and service partners. A number of solutions have been proposed by the software industry [9]. One of them is OWL-S. It is a framework based on the W3C OWL Web Ontology Language, proposed to help service requestors to search, discover, invoke, compose and monitor Web services. OWL-S allows for describing Web services' features as well as for providing construct activities such as *Sequence*, *Split*, *Split + Join*, *If-Then-Else*, *Repeat-While*, and *Repeat-Until*. In contrast to BPEL4WS, OWL-S does not provide asynchronous invocation, fault and event handlers [8].

A. OWL-S Characteristics

OWL-S consists of three Models: *Service Profile*, *Process Model* and *Grounding*, used to represent different aspects of a service. The Service Profile Model describes the service features to other services or agents that want to use it. It defines the service with regards to its inputs, outputs, effects and precondition parameters. The Process Model is the essential model of the OWL-S architecture. It specifies how the process is used. Services can be composed using a combination of atomic, simple or composite services. Additionally, the Process model defines the order in which involved Web services are composed, either in sequence or in parallel. It allows for describing conditional activities. With the Process Model, it is possible to create loops and declare variables. Finally, the grounding Model defines how to interact with the service by providing the necessary concrete details related to the transport protocol and message format [2] [10]. In the following, we briefly describe the main elements of the OWL-S process as defined in [2]:

1) The *Service Profile* describes a service as a function of three basic types of information:

a) the *provider information*, which consists mainly of the <*serviceName*> and the <*textDescription*> (a brief description of the service);

b) the *functional description*, which consists of the <*hasInput*>, <*hasOutput*>, <*hasPrecondition*> and <*hasResult*> constructs;

c) the *properties description*, which allows for the description of a host of properties that are used to describe features of the service for example its category.

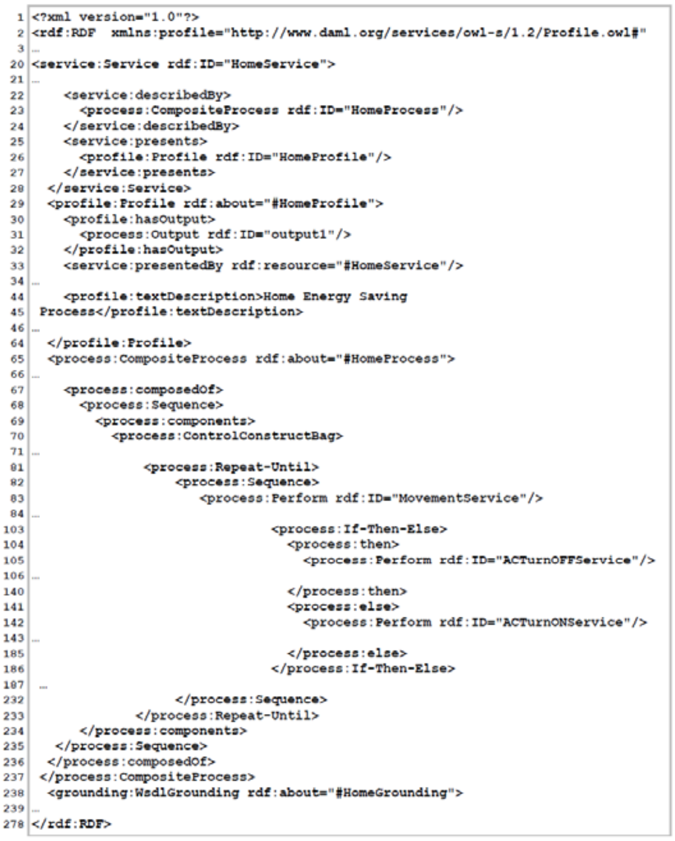
2) The *Process Model* consists of the <*composedOf*>, the <*Perform*> (for invoking other processes), the <*Sequence*>, the <*Split*> (for parallel processing) and the <*If-Then-Else*> constructs.

3) The *Grounding model* refers to specific elements within the WSDL specification by using a set of constructs such as the <*wsdlDocument*>, *wsdlOperation*>, <*wsdlInput*>, etc.

B. Home Energy Saving Scenario Implementation Using OWL-S

Let us now try to solve the service composition problem of the energy saving plan from a Semantic Web perspective. Fig. 3 shows an extract of the code of the process. It is briefly commented below:

- *Service Profile*: lines 20-64 define the service profile information as mentioned in the previous section.
- *Process Model*: Lines 64-237 specify the process main body which defines the order in which the partner Web services are invoked. The work flow structure starts gathering temperature and movement detection data. Then the process is executed through different scenarios depending on the received data in order to save energy. Finally, the process repeats the previous steps through a while loop statement.
- *Grounding*: lines 237-278 define the grounding model of the process which maps the elements of the WSDL and OWL-S documents. It provides necessary concrete details related to the transport protocol and messages format.



```
  1 <?xml version="1.0"?>
  2 <rdf:RDF  xmlns:profile="http://www.daml.org/services/owl-s/1.2/Profile.owl#"
  3 ...
 20 <service:Service rdf:ID="HomeService">
 21 ...
 22     <service:describedBy>
 23       <process:CompositeProcess rdf:ID="HomeProcess"/>
 24     </service:describedBy>
 25     <service:presents>
 26       <profile:Profile rdf:ID="HomeProfile"/>
 27     </service:presents>
 28   </service:Service>
 29   <profile:Profile rdf:about="#HomeProfile">
 30     <profile:hasOutput>
 31       <process:Output rdf:ID="output1"/>
 32     </profile:hasOutput>
 33     <service:presentedBy rdf:resource="#HomeService"/>
 34 ...
 44     <profile:textDescription>Home Energy Saving
 45  Process</profile:textDescription>
 46 ...
 64   </profile:Profile>
 65   <process:CompositeProcess rdf:about="#HomeProcess">
 66 ...
 67     <process:composedOf>
 68       <process:Sequence>
 69         <process:components>
 70           <process:ControlConstructBag>
 71 ...
 81               <process:Repeat-Until>
 82                   <process:Sequence>
 83                      <process:Perform rdf:ID="MovementService"/>
 84 ...
103                               <process:If-Then-Else>
104                                 <process:then>
105                                   <process:Perform rdf:ID="ACTurnOFFService"/>
106 ...
140                                 </process:then>
141                                 <process:else>
142                                   <process:Perform rdf:ID="ACTurnONService"/>
143 ...
185                                 </process:else>
186                               </process:If-Then-Else>
187  ...
232                   </process:Sequence>
233              </process:Repeat-Until>
234       </process:components>
235    </process:Sequence>
236   </process:composedOf>
237  </process:CompositeProcess>
238   <grounding:WsdlGrounding rdf:about="#HomeGrounding">
239 ...
278 </rdf:RDF>
```

Figure 3. Code Extract of the Home Energy Saving OWL-S Process

V. Solutions Comparison

BPEL4WS and OWL-S provide different solutions to compose web services. However, many concepts that are implemented in these two approaches are similar. Below is a high level summary comparing the features and characteristics of the two approaches:

- *Primitive and Structured Activities*: Both OWL-S and BPEL4WS allow for invoking, receiving messages or replying to external web services and partners. Structured activities are provided by BPEL4WS as well as OWL-S and they allow for executing services in sequence or in parallel in order to build complex scenarios. The While (BPEL4WS) and the Repeat-While (OWL-S) structures have similar functionalities. OWL-S offers Split+Join and BPEL4WS uses Flow activity for concurrency invocation [10].

- *Service Discovery*: The energy saving scenario shows the necessity of a mechanism which allows for discovering involved services. OWL-S can help to find services that match with a service request based on semantic service description [11]. BPEL4WS just enables composing explicit web services into composite processes without providing capabilities of semantic service discovery.

- *Semantic Description*: Home devices are isolated and provide concrete functions but they lack a real capacity to describe themselves. OWL-S describes the features of the service through its service profile model. The latter contains information such as the service's name, inputs, outputs, enterprise contacts and category. By describing a service through its semantic features, the OWL-S approach helps web services discovery and composition. Conversely, BPEL4WS does not allow that. There are, however some propositions for increasing the semantic features of BPEL4WS [12].

- *Event Handler*: Events are essential to inform the process about any new change which may occur. The event handler is a BPEL4WS characteristic. OWL-S does not have yet a concept of an event handler. Thus, in our home saving scenario, we had to go around this limitation by using loops for checking permanently the temperature sensors and movement detection services.

- *User Involvement*: In the home energy saving scenario, it is a challenging problem to search and select the concrete services and involve users to adjust them in order to achieve their desired goals. Neither BPEL nor OWL-S does directly support human involvement since the user must tell the system what to do during the development of the composition process (i.e. before its execution).

- *Abstract Process Definition*: The end-users expect a high-level of abstraction in a service composition process. They are not able to use very technical and complex tools. Therefore, the composition implementation should be hidden from them. This implies that an abstract process should be defined to perform the composition goal. This can be achieved using semantic service description and defining generic process templates which contain the involved services as well as the interaction between the user and the system in order to adapt the process to the user's needs. BPEL4WS only addresses the syntactical aspects of web services, which prevent involving semantic web service description in the process [13].

- *Message Correlation*: Message correlation is the BPEL feature which enables several processes to interact in stateful conversations. OWL-S does not provide such a feature.

- *Asynchronous Invocation*: OWL-S does not support asynchronous web services invocation and process persistence. This limits its ability to manage both atomic transactions as well as long-running business transactions. Conversely, BPEL4WS supports both asynchronous and synchronous invocation mechanisms.

- *Dynamic Composition*: Composition of web services using languages like BPEL or OWL-S is normally generated off-line. Modifying any part of a process may result in the reconfiguration and redeployment of the whole process. Currently, only BPEL supports (partially) fail-over and dynamical redesign [9] [14].

VI. Proposing a Combined Approach

To have a complete and efficient system for web services composition, there is a need to propose a semantic-based framework which offers flexibility to integrate services and reinforces the human-computer collaboration paradigm. We propose a template based composition technique to facilitate the discovery and the semi-automatic composition of web services. It is not up to the user to tell the system what to do but rather to establish and negotiate about goals and how to accomplish them. When possible, the system must involve the user in selecting the appropriate services and actions, which both respect the context (i.e. the available discovered services) and the initial objectives (i.e. the process generic template). This demands understanding the capabilities of those services as well as the conditions and requirements which must be met to accomplish the composition goal.

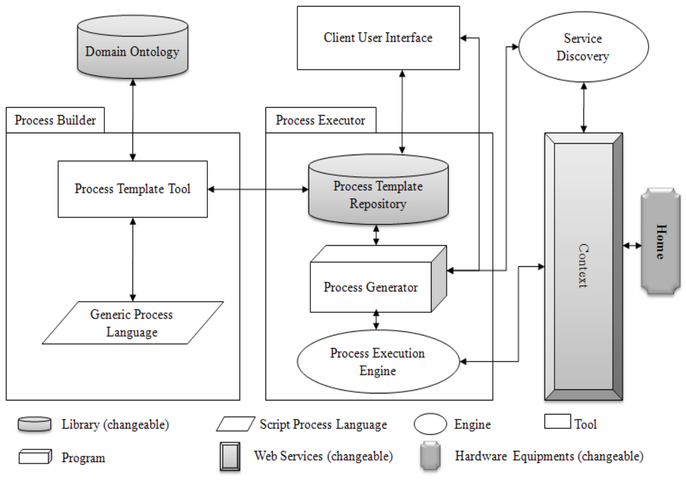


Figure 4. Semi-automatic Service Composition Framework

Fig. 4 gives an overview of our approach for semi-automatic services composition. The user can choose a recommended generic process template from the *process* *repository*. The generic process template acts as a configurable module. It defines generic participating activities and the control flow. We are still investigating the best way (language) to describe such templates. Indeed, this question will be one of our major research focuses. It is possible that arbitration will be required between being totally open to all domains or being more adapted to specific tasks (e.g. smart homes).

The *Process Generator* component captures the generic activities' characteristics in the process template and sends them to the *Service Discovery Engine* as services queries. Web services are usually published in registries (Discovery Engine). Consumers can request available services by a keyword-based search engine (e.g. expedia.com, google.com) or by looking it up in a web services registry (e.g. UDDI – Universal Description, Discovery and Integration Registry) [15]. Improving service discovery involves adding a semantic research mechanism [12]. The requestor can provide the inputs and outputs of the required service. Then the discovery process finds any service which matches these semantic requirements. In our framework, service discovery is based on the semantic description of services and of the *domain ontology*.

After the services have been discovered, the user interacts with the system in order to make the appropriate choices. When a service is put into the composition, the information about inputs, outputs, preconditions and effects (IOPE) is checked automatically to assure that all needed input data are provided; that all operations can be executed; and that all links are established. The process can now be converted into an executable process by the Process Generator. Finally, the *Process Execution Engine* executes the generated process using an execution language. We are still investigating the best solution for such an engine, but it will most probably be similar to BPEL4WS with the addition of some OWL-S features.

To validate our approach, we intend to develop such a prototypical semi-automatic composition framework in the domain of smart homes. We also intend to base the framework on pluggable generic components (the darkened elements in Fig. 4), which will allow to change its domain (e.g. hospital, university environments).

VII. Related Work

Manual composition approach is generally used in the situation where the requestor has a well-defined process model. Processes are defined using a process execution language like BPEL or OWL-S. The problem with such an approach is that it demands too much knowledge on the part of the user and it becomes more and more difficult with the explosion of web services resources. Automatic composition (without human involvement) is used when the requestor has no process model but has a set of constraints and preferences. Several approaches for automatic service composition have been introduced, including solutions based on Hierarchical Task Network (HTN) [16], Goal Description Languages for Services Composition [17], Artificial Intelligence planning [18] or Rule-Based planning [19]. However, automatic composition is still viewed as a task of high complexity because of the rapid proliferation of available services to choose from and, the heterogeneity of the data formats they offer. Finally, the composition result risks to differ from the user's original goal. The third approach (closed to our views) is called semi-automatic or interactive composition. In this kind of composition, the system usually helps users to find, filter, and integrate automatically the desired services by matching the users' requests with the available services. Moreover, it enables end-users to intervene continuously during the composition process. Some efforts like OWL-S, METEOR-S [27] used semantic description of web services to aide in improving discovery and composition processes.

[11] [12] [20] [21] [26] provide mechanisms for semantic annotation of existing services, services discovery and arbitrary service composition. Others approaches have emerged to support specifically end-users in the service composition process. These approaches for example [22] [23] [24] [25] focus mainly on using techniques for involving users to compose services or inform the user about issues to be addressed in the current workflow. These approaches are applicable for some type of implementation. We argue that some problems still exist in current semantic services composition frameworks. In particular, there is a lack of a good generic process template language which involves generic services and user preferences to reach a specific goal.

Conclusion

We have explored a few emerging concepts in the area of web services composition, business process and workflow. A set of open, standards-based technologies like BPEL4WS and OWL-S are available for designing and executing interactions between numerous web services. Our home energy saving scenario demonstrates several challenges, which need to be addressed in order to build a flexible and generic web services composition framework. One problem is certainly the definition of a simple, yet flexible, language in order to describe generic process templates for various rich scenarios given specific domains ontologies, (e.g. energy saving scenario or maximum security scenario for a smart homes domain). Another interesting challenge consists in the creation of a process generator engine able to appropriately transform a generic scenario into an executable one with the help of both the end-user and of a powerful semantic discovery mechanism. Finally, the selection (or creation) of an execution engine, which will be able to react to a changing context, represents another non trivial task.

References

[1] OASIS, "Web services business process execution language," http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html, [Accessed 05 20, 2011]

[2] W3C, "OWL-S:semantic markup for web services", http://www.w3.org/Submission/OWL-S/, [Accessed 05 15, 2011].

[3] W3C, "Web service modeling ontology (WSMO)," http://www.w3.org/Submission/WSMO/, [Accessed 05 22, 2011].

[4] AMIGO, "Ambient Intelligent for the Networked Home Environment," http://www.hitech-projects.com/euprojects/amigo [Accessed 05 31, 2010].

[5] André Botaro, Anne Gérodolle, Philippe Lalanda, "Pervasive Service Composition in the Home Network," proceedings of 2007 21th International Conference on Advanced Nteworking and Applications.

[6] Thinagaran Perumal, A.R.Ramli, Chui New Leong, “Design and Implementation of SOAP-Based Residential Management for Smart Home Systems,” IEEE Transactions on Consumer Electronics, 2008, pp. 453-559.

[7] W3C, “Web services description language (WSDL) 1.1. 2001. http://www.w3.org/TR/wsdl [Accessed 05 25, 2011].

[8] Milanvoic, Nikola et Malek, Miroslaw, “Current solutions for web services composition,” IEEE Computer Society, 2004.

[9] I. Smeureanu, A. Diosteanu, “Knowlegde Dynamics in semantic web services composition for supply chain management applications,” Journal of Applied Quantitative Methods, 2005.

[10] D. Barreiro Claro, P. Albers. “Approaches of web services composition comparison between BPEL4WS and OWL-S,” Miami-USA : In Proceedings of ICEIS’05, 2005, pp. 208-213.

[11] B. Fries, M. Klusch and P. Kapahnke, “Hybrid OWL-S web service matchmaker” 2008, http://www-ags.dfki.uni-sb.de/~klusch/owls-mx/index.html [Accessed 03 20, 2010].

[12] D. Karastoyanova, T. van Lessen,F. Leymann, J. Nitzsche and D. Wutke, “WS-BPEL Extension for semantic web services (BPEL4SWS),” Stuttgart, Germany : Institut für Architektur von Anwendungssystemen, 2008.

[13] P. Martinek, B. Szikora, “Semantic Execution of BPEL Processes,” Advances in Information Systems Development, 2008, pp. 361-367.

[14] J. Dong, Y. Sun, S. Yang & K. Zhang, “Dynamic web service composition based on OWL-S,” in China Press co-published with Springer, Science in China Series F: Information Sciences, 2006, vol.49, No.6, pp. 843-863.

[15] UDD, “UDDI Committee Specification,” http://uddi.org/ [Accessed 31 05, 2011].

[16] Incheon, Paik and Daisuke, Maruyama, “Automatic Web Services Composition Using Combining HTN and CSP,” Aizu-Wakamatsu City, Fukushima, Japan : 7th IEEE International Conference on Computer and Information Technology (CIT 2007), 2007. pp. 206-211.

[17] Manshan Lin, Heqing Guo, Jianfei Yin, “Goal Description Language for Semantic Web Service Automatic Composition,” Washington, USA : Proceedings of the The 2005 IEEE, Symposium on Applications and the Internet, 2005. pp. 190 – 196

[18] Luciano A. Digiampietri, José J. Pérez-Alcàzar, Claudia Bauzer Medeiros, “AI Planning in Web Services Composition: a review of current approaches and a new solution,” Anais do XXVII Congresso da SBC, ENIA VI Encontro Nacional de Inteligenia Artificial , Rio de Janeiro, 2007. pp. 983 - 992

[19] Hui, Li, Haiyang, Wang and Lizhen, Cui, “Automatic Composition of Web Services Based on Rules and Meta-Services,” Melbourne, Australia :Proceedings of 11th International Conference on CSCW in Design, 2007. pp. 496-500.

[20] H. Janqiang, “SWSCF:_ A semantic-based web service composition framwork”, Journal of networks, vol. 4, June 2009, pp. 290-297.

[21] S. Bleul, T. Weise, and M. Kirchhoff, “Semantic web service composition for service-oriented architectures,” Washington, USA : Proceedings of the 2008 10th IEEE Conference on E-Commerce Technology, 2008, pp. 355-358.

[22] E. Toch, I. Reinhartz-Berger, A. Gal and D. Dori, “Generating and optimizing graphical user interfaces for semantic service compositions,” Barcelona, Spain : Springer, 2008, vol 5231, pp. 528–529.

[23] P. Xiaoming, F. Qiquing, H. Yahui, Z. Bingijan, “A user requiremnts oriented dynamic web serivce composition framwork,” Internaional Form on Information Technology and Applications IEEE, 2009, pp. 173-177.

[24] E. Sirin, J. Hendler and B. Parsia, “Semi-automatic composition of web services using semantic descritpitons,” Angers, France : workshop in conjunction with ICEIS, 2003.

[25] OASIS, “WS-BPEL Extension for People (BPEL4People) Specification Version 1.1,2009” http://docs.oasis-open.org/bpel4people/bpel4people-1.1-spec-cd-06.pdf, [Accessed 05 31, 2010].

[26] PonHarshavardhan, J. Akilandeswari, M. Manjari, “Dynamic web service composition problems and solution - a survey,” MES Journal of Technology and Management, 2011, vol. 2, issue 1, pp. 1-5.

[27] LSDIS, “METEOR-S: Semantic Web Services and Processes,” http://lsdis.cs.uga.edu/projects/meteor-s/, [Accessed 08 21, 2011]