Semantic-Based Semi-Automatic Web Service Composition

Abdaladhem Albreshne[#1], Jacques Pasquier[*2]

[#]*Computer Department, Fribourg University Switzerland*

[1]abdaladhem.albreshne@unifr.ch

[*] *Computer Department, Fribourg University Switzerland*

[2]jacques.pasquier@unifr.ch

Abstract*—The presence of software systems in every aspect of our life results in increased requirements and expectations addressed by new development projects. The domain of home computing constitutes a typical example of an environment which is characterized by complex requirements with regard to context awareness, intelligent assistance, autonomy, dynamic discovery, dynamic service composition, and easy services management for end users. This work aims to propose an approach based on semantic description and ontologies in order to discover and compose services in a home changing environment. Furthermore, our goal is to offer a partial automation of web service composition, with a human controller. We propose to enable customers to select and configure available services to meet their requirements and reach a specific goal.*

1 Introduction

Over the past several years, a range of industry languages and frameworks solutions have been realized to enable web service composition. Among these, Business Process Execution Language for Web Services (BPEL4WS) [1] is probably the most prominent. It provides a language for web service composition where the flow of processes and the bindings between services are known a priori. These approaches are of purely syntactical nature. There are still challenges in the web service composition field which need to be addressed and investigated. For example, there is a general lack of methodology and tools which enable the semi-automatic composition and analysis of web services taking into account their semantic and behavioural properties. Even though a lot of work has been done in the field of semi-automatic composition [2-3], there still remain problems. These are mainly related to the question how to help non-expert users to achieve goal oriented service composition. An essential issue is how available semantic services can collaborate and use domain knowledge and user inputs to help achieving semi-automatic service composition for dynamic adaptation to changing business requirements.

We propose an approach which uses domain information and semantic to define a specific environment. Our main objective is to offer non-expert users who tend to not know in advance how to achieve their goals, an assistant mechanism to obtain optimized solutions to compose web services in a way that meets their changing requirements. To attain this objective, we propose to users generic process templates. These templates are oriented toward users'

goals. A process template defines a work-flow which is composed of several activities with specific functions (semantic description of participating web services) linked with control flow, structured activities (loops, If statements, sequences, etc.), and user preferences that need to be involved in the process. The proposed process template is used to define the services types and to offer a suitable decision at each step of the composition depending on the process goal, user choices and the current situation. The end user configures and customizes these generic templates according to his current requirements, preferences and environmental context. We explain our work in more detail in Section 3.1.

2 Motivating Scenario

In this section we explore the presented approach through a case study realized in the field of home environment.

Home Energy Saver Scenario

Thomas home is a home equipped with diverse web enabled sensors and actuator devices such as light switches, home heating system, air-conditioners, shower indicator, temperature sensors, curtains and windows controllers and door controllers, etc. Thomas controls his home environment in a way that allows him to save energy and to adapt his environment to his habits and live conditions.

Daily observation shows that total energy consumption can be minimized by 1) **Reducing wastage**: lighting, heating and air-conditioning (HVAC), and other systems can be turned off when not needed. 2) **Reduced lighting levels** (dimming): a light can be operated at less than 100% when full light is not needed. 3) **Shower time**: limiting the shower time can economise energy. There is no need to have hot water for the shower when the user is outside for hours. The developer has used the recommendations described above in order to create a generic energy saver process template for the final user .

Thomas consults the menu of his automation system which proposes several composition scenarios. He chooses the “Energy Saver Plan” which is a generic predefined process plan. Figure 1 illustrates the Energy Saver Plan workflow. The tool suggests him different options, step by step, and in each step he can define his preferences. The following steps resume the scenario interaction between the system and the final user:

- At first, the system discovers available services (devices, sensors) and understands their functions.
- Then, the system selects the services that are potentially involved in the energy saver process. The system detects a heating system, three air-conditioners, four room lamps, two entrance overhead lights and an external temperature sensor.
- The system starts configuring the energy saver plan by asking the final user to take a decision about the involved services as following:
  - What is the interior comfort home temperature when you are at home?
  - Select from the given list which entrance overhead lights to turn on at night.
  - Choose which air conditioners you like to turn on.
  - Set the shower timer.
  - Adjust the preferred temperature for each air conditioned room.
- After having finished asking the user about services and his preferences, the process is now configured to meet the user’s requirements and the system is able to create a final process. The defined activities in the process are bound to concrete web services and the data flow in the process is

assigned. The process is ready to be executed.

This scenario demonstrates that there is a need to help the final user in a changing environment where services are not known at design time. It shows that decisions sometimes could not been taken without the end-user involvement. It is a challenging problem to search and select the concrete services and involve users to adjust them in order to achieve the desired goal. So as to overcome this lack of flexibility, we propose to develop a framework as a prototype, consisting of an intuitive user tool that allows users to create and run on their home device customized programs that implement the intended smart home tasks. Additionally, this tool need to be aware of the context in which the composition and invocation of the available web services (devices, sensors) occur.

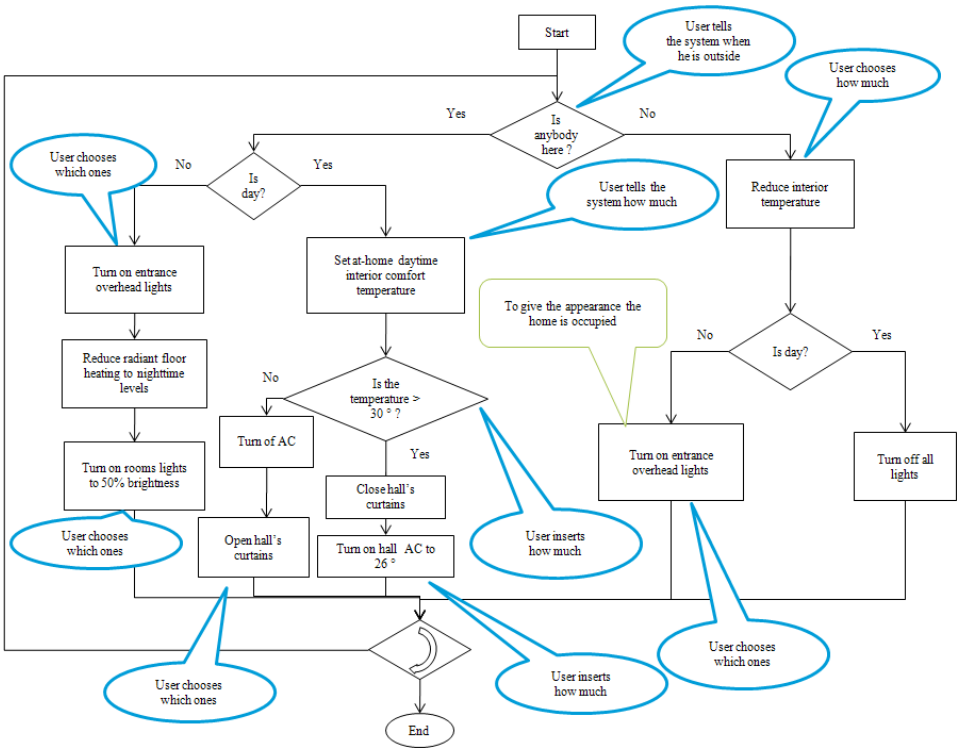

Figure 1 Home Energy Saver Workflow

3 Semi-automatic Service Composition Approach

In general, there are several kinds of service composition. The traditional approach is called ***manual composition*** where users program and tell the system what to do during all the composition process development steps. Processes are defined using a process execution language like BPEL. Many plug-ins for tools like Net-Beans [4], JOpera [5] are available to enable manual composition. 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.

The second approach is called ***automatic composition*** (without human involvement). It is used when the

requestor has no process model but has a set of constraints and preferences. It is based on finding services for executing predefined abstract processes. The tools try to discover the available web services that semantically match as much as possible the user's needs [6]. Several approaches for automatic service composition have been introduced [7], including solutions based on Hierarchical Task Network (HTN), Golog [8], Artificial Intelligence (AI) planning or Rule-Based planning [7, 9-10]. 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 composition result risks to differ from the user's original goal.

The third approach is called ***semi-automatic or interactive composition***. We work in this direction. 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 [11], METEOR-S [12] use semantic description of web services to help improving the discovery and composition processes. We believe that the semantic of the provided services could be used by tools or systems to guide the user to limit the available choices and to define his preferences to finally reach the composition goal.

3.1 Goal Oriented Service Composition Approach

The composition of web services is difficult when one is using exclusively the WSDL [13] descriptions with a composition language like BPEL, since each description lacks the semantic description of services' properties and capabilities as well as some non-functional attributes, such as service name, service type or service location. To be able to describe the services, semantic web languages like the Ontology Language for Web Services (OWL-S), and the Web Services Modeling Ontology (WSMO) [14] 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.

Our approach uses web services and semantic web service technologies like OWL-S to facilitate the discovery, as well as the semi automatic composition of web services. We propose an assistance mechanism for the semi-automatic composition of services where the composition is gradually generated by using a declarative oriented generic composition plan. 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. At each composition step, the system could propose to the user which new service can be added to the composition and which kind of actions can be taken. Further possibilities are filtered based on the current context, the composition objective and user preferences.

To validate our approach, we want to develop an interactive service composition smart home prototype that assists non-expert users to compose and generate goal oriented composition plans in which the system is able to provide suggestions and direct the control flow in a step by step style. Depending on the composition goal, context and the user's preferences, the system selects the suitable services and which available decisions could be taken.

However, several challenges need to be addressed in order to build our system: for instance, lack of a generic process language which would facilitate the orchestration of semantic web services, a need to involve user preferences and

context-awareness in the process, the requirement to build a process design tool based on semantic and ontology and finally the question of how services discovery can be optimized. We provide principles for the underplaying software architecture which make it easier to create such a system. In the next Section, we explore those challenges in more detail and present our framework.

3.2 Proposed Framework

In the following, we give an overview of our framework for semi-automatic web service composition. As shown in Figure 2, the user can choose a recommended generic process template from the process repository. The generic process template acts as a configurable module. It defines the semantic of the participating activities, control flow and conditional branches. There are several ways to write process templates. For example, PEBL4SWS [1] is an extension of BPEL that allows to create a semantic process and then generate an executable one. OWL-S can be used as well to execute the process. Our framework is not restricted to theses languages. If none of these solutions corresponds to our requirements, we can propose our own language. The Process Generator component captures the semantic activities' characteristics in the process template and sends it to the Service Discovery Engine as service query. 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. The requestor can provide the inputs and outputs of the required service. Then the discovery process in the registry will find any service which matches these semantic requirements. Several semantic discovery algorithms have been proposed like [6].

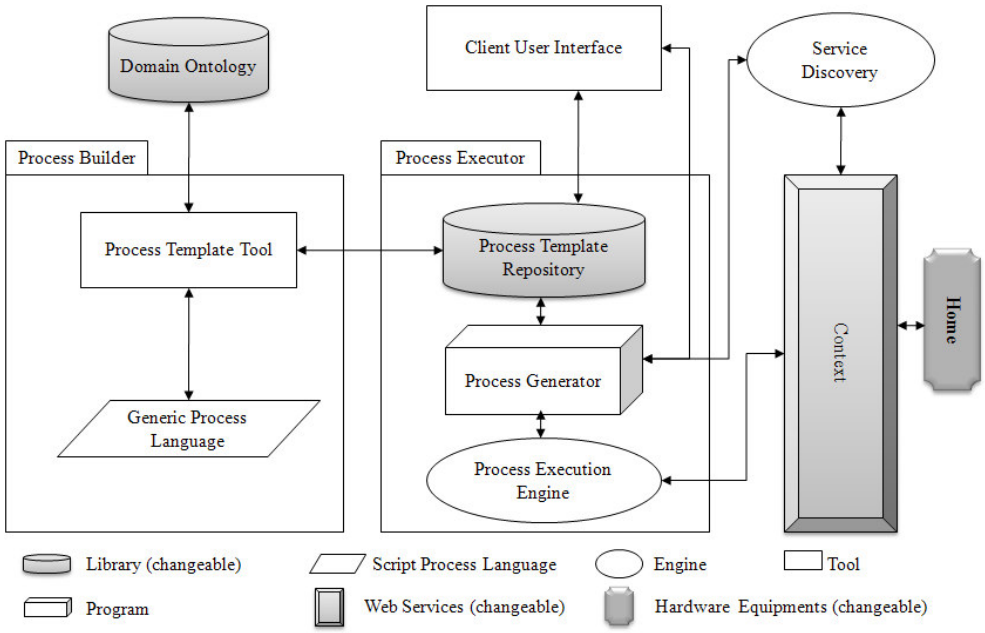

Figure 2 Semi-automatic Service Composition Framework

In our framework, service discovery is based on services' semantic description and their relation with the domain ontology. After Services have been discovered by the Service Discovery Engine, the user binds all desired activities and defines his preferences according to the activities configuration requirements which can be involved in the generic process or can be declared separately in another format. When a service is put into the composition, the information about input, output, preconditions and effects (IOPE) of this service is checked automatically to assure that all needed input data are provided, all operations can be executed and all links are established. The process now can be converted into an executable process by the Process Generator. The Process Execution Engine component has the capability to execute the generated process using an execution language such as BPEL4WS or OWL-S API. In the next section, we explain briefly the role of our framework components:

3.3 System Architecture

The proposed architecture (See Figure 3) supports the construction and execution of semi-automatic service composition. The system architecture is based on three categories of components: Service Discovery Component, Process Building Components, and Process Configuration & Execution Components.

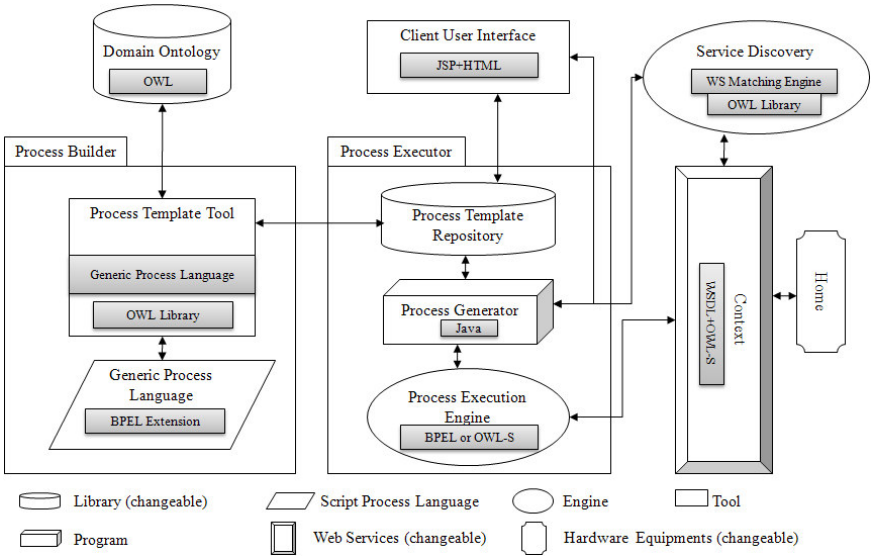

Figure 3 – Semi-automatic Service Composition Architecture

1. **Service Discovery Component**
   The providers publish their web services on a web services registry.
   - [***Service Discovery***]: The Service Discovery & Registry has registry, discovery and selection functions. It uses a matching engine to find the requested services by comparing their semantic descriptions with the available registered services.
2. **Process Building Components**
   The process developer uses a graphical design tool in order to build a generic process template. He uses a published domain ontology which is related to a specific organization to describe participating activities semantically.

- [***Domain Ontology Library***]: stores the detail information of a specific domain like services or devices. The domain ontology is defined using the OWL language.
- [***Process Template Design Tool***]: a graphical composition tool. It lets the expert user create a generic process template by defining the workflow of services according to the composition goal. It uses the capabilities of the domain ontology in order to provide the user with a task abstraction of the complexity of whatever lays underneath. The tool includes various libraries such as BPEL4SWS or OWL-S and OWL.
- [***Process Template Repository***]: contains abstract services composition templates. These composition templates define the capabilities and functionalities of needed services as well as the conditions and requirements that must be applied to achieve the composition goal.

3. **Process Configuration and Execution Components**
   The Client uses the client interface to configure a process template that matches her goal and provides her preferences. Then the process can be executed. The following components allow realizing this objective:

- [***Client User Interface***]: a graphic tool which handles the communication between the end-user and the platform. It lets the user choose a template process in order to achieve a specific goal. Then the user configures this process according to her preferences. The tool will present the available choices at each step according to the proposed plan. After process configuration, the process is executed by the execution engine.
- [***Process Generator***]: handles the process configuration and converts the generic process into an executable one.
- [***Execution Engine***]: has the capability to execute the generated process. It uses an execution language such as BPEL4WS or OWL-S API [16].

4 Related work

On the one hand, a range of industry solutions have been realized to enable web service composition. Among these, the Business Process Execution Language for Web Services (BPEL4WS). These approaches only address the syntactical aspects of web services. On the other hand, there are efforts which go in a different direction. They aim to establish standards enabling the syntactic and semantic description and composition of Web Services. For instance, the Ontology Web Language for Services (OWL-S [17]) has triggered significant research efforts to build frameworks and tools which allow the composition of semantically annotated Web Services.

Artificial Intelligence (AI) approaches for web service composition focuses on automatically generating a plan of a composite service that satisfies users given goals [9-10, 18]. We propose an alternative approach which involves users in the composition process, including the selection of components and their configuration.

[19] proposes a system which informs the user about issues to be addressed in the current workflow. It is limited by the fact that the composition assistant occurs during the design phase. Consequently, it does not enable users to discover and then choose concrete services which meet their requirements. On the contrary, the user interaction

methods and tools proposed by [2, 20] capture the user's intentions in an interactive and continuous manner during the whole composition process. These systems utilize the semantics of the available services to guide the user and limit the available choices. Nevertheless, they do not guide the user to reach the objective of the composition.

[21] proposes a goal description language for automatic web service composition (GDL4WSAC). The latter describes the goals to be achieved and the corresponding constraints unambiguously. Its limitations are that it fits developers more than end-users, since the goal must be predefined and since there is no way for users to intervene during the composition. [22] proposes a semantic-based ontology language (OWL-T) used to formulate business demands in terms of structured task templates. An automatic composition method is used to transform task templates into executable processes. Neither in the composition process nor in services selection the user is involved.

[23] proposes a web service composition framework SWSCF which is based on semantic and has the ability to integrate services according to application domain semantics and dynamic business requirements. A hierarchical activity mechanism (AI technique) enables this framework to dynamically decompose business requirements. This helps to identify suitable semantic process templates.

Task Computing approach [24] exposes the functionality found in smart environments (i.e. networked devices, web services) as semantic web services, which in turn the user can discover and arbitrarily compose. It focuses namely on dynamic service discovery as well as service publishing and management.

5 Conclusion

The contribution of this work is: First, proposing a semantic-based framework which offers flexibility to integrate services and reinforces the human-computer collaboration paradigm. Second, we use the semantic descriptions and ontologies to provide an assistant mechanism to obtain optimized solutions to compose services in a way that meets user's changing requirements.

Some key features of our approach are:

- defining a generic process template to describe semantic activities for goal oriented composition
- using a semantic matchmaking mechanism to discover and select the suitable services that conform to semantic activities defined in the process template
- enabling users to adjust the process, where the process template acts as a configurable module for user preferences and requirements.
- enabling end-users to choose concrete services to be invoked according to their changing business requirements.
- Providing context awareness through a context discovery module.
- providing an ontology-based tool that enables semantic-based, goal oriented, semi-automatic service composition.
- providing a process execution engine, which has the capability to execute the generated process.

6 Future Work

In order to have a good evaluation of our approach, we are currently working on our smart home prototype. Several challenges need to be addressed in order to build our system: for instance,

a generic process languages which facilitate the orchestration of semantic web services, a need to involve user preferences and context-awareness in the process, as well as building a process design tool based on semantic.

7 References

[1] IBM. (2002, 11.05.2010). Business Process Execution Language for Web Services version 1.1. Available: http://www.ibm.com/developerworks/library/specification/ws-bpel/

[2] B. Parsia, et al., Semi-automatic composition of Web Services using Semantic Descriptions. Angers, France: Web Services: Modeling, Architecture and Infrastructure Workshop in Conjunction with ICEIS, 2003.

[3] B. Steffen, et al., "Semantic Web Service Composition for Service-Oriented Architectures," presented at the 10th IEEE Conference on E-Commerce Technology, Washington, USA, 2008.

[4] NetBeans.org. (2007, 11.05.2010). NetBeans IDE 6.1 SOA Pack Documentation. Available: http://netbeans.org/kb/61/soa/index.html

[5] JOpera. (2010, 12.05.2010). JOpera for Eclipse. Available: http://www.jopera.org

[6] F. Benedikt, et al. (2008, 02.02.2010). Hybrid OWL-S Web Service Matchmaker. Available: http://www-ags.dfki.uni-sb.de/~klusch/owls-mx/index.html

[7] H. Li, et al., "Automatic Composition of Web Services Based on Rules and Meta-Services," presented at the 11th International Conference on CSCW in Design, Melbourne, Australia, 2007.

[8] S. Narayanan and S. A. Mcllraith, "Simulation Verification and Automated Composition of Web Service," presented at the 11th International World wide Web conference, Hawaii, USA 2002.

[9] I. Paik and D. Maruyama, "Automatic Web Services Composition Using Combining HTN and CSP," presented at the 7th IEEE International Conference on Computer and Information Technology (CIT 2007), Aizu-Wakamatsu City, Fukushima, Japan, 2007.

[10] X. Li and C. Wu, "Research on OWL-S Service Automatic Composition Based on Planning," Wuhan, China 2009.

[11] W3C. 09.05.2010). OWL-S: Semantic Markup for Web Services. Available: http://www.w3.org/Submission/OWL-S/

[12] LSDIS. (2005, 06.03.2010). METEOR-S: Semantic Web Services and Processes. Available: http://lsdis.cs.uga.edu/projects/meteor-s/

[13] W3.org. (2001, 10.05.2010). Web Services Description Language (WSDL) 1.1. Available: http://www.w3.org/TR/wsdl

[14] W. W. W. Consortium. (2005, 11.05.2010). Web Service Modeling Ontology (WSMO). Available: http://www.w3.org/Submission/WSMO/

[15] OASIS. (2004, 10.04.2010). UDDI Spec Technical Committee Draft. Available: http://www.uddi.org/pubs/uddi_v3.htm

[16] Mindswap. (2010, 12.05.2010). OWL-S API. Available: http://www.mindswap.org/2004/owl-s/api/

[17] DAML. (2009, 07.05.2010). DAML Services. Available: http://www.daml.org/services/owl-s/

[18] K. Srividya, et al., "Automatic Composition of Semantic Web Services," presented at the IEEE 2007 International Conference on Web Services (ICWS), Salt Lake City, USA, 2007.

[19] K. Jihie, et al., "An Intelligent Assistant for Interactive Workflow Composition," presented at the 9th international conference on Intelligent user interfaces, Madeira, Portugal, 2004.

[20] E. Sirin, et al. (2004). Composition-driven Filtering and Selection of Semantic Web Services.

[21] M. Lin, et al., "Goal Description Language for Semantic Web Service Automatic Composition," presented at

the IEEE, Symposium on Applications and the Internet, Washington, USA, 2005.
[22] V. X. Tan and H. TSUJI, "OWL-T Ontology-based Task Template Language for Modeling Business Processes," presented at the Fifth International Conference on Software Engineering Research, Management and Applications, Busan, South Korea, 2007.
[23] J. Hu, et al., "SWSCF: A Semantic-based Web Service Composition Framework," Journal of Networks, ACADEMY PUBLISHER, vol. 4, pp. 290-297, 2009.
[24] R. Masuoka, et al., "Semantic Web and Ubiquitous Computing -Task Computing as an Example-," Maryland, USA, 2004.