JACINTA: a Mediator Agent for Human-Agent Interaction in Semantic Web Services

Mariano Rico, Pablo Castells
Universidad Autónoma de Madrid
Ctra. de Colmenar Viejo km. 15, 28049 Madrid
{mariano.rico, pablo.castells}@uam.es

Abstract

The proposed system aims at filling a gap in the current Web Service technology (semantic or not): the interaction with the end-user. These systems must be aware of access device capabilities for the end user, as well as his/her aesthetic preferences. The system request from the user the input data needed to invoke services, displays the results from service execution, and prompts the user in case of error, or when the system needs some feedback and propose to the user some hints to obtain a smaller result. We propose a mediator agent for such needs in the interaction between human users and web services.

1 Introduction

The currently predominant technologies for web service development, based on WSDL, allow application programmers to use web services as a traditional API. As such, the necessary documentation needs to be shipped with the API for its use. For example, Amazon, one of the pioneers in these technologies, provides a manual of 78 pages. Like with traditional APIs, any change to the API would imply having to adapt all the applications that used the old API.

The Semantic Web Services (SWS) vision promises a more conceptual and intelligent communication, so that the effects of the changes in terms of required manual development adjustments are drastically reduced. These new Web Services would be dynamically invoked (used) by semantic agents (SA-SWS).

The technology that can make SWS and SA-SWS a reality is yet under development, and to a large extent an open area of research. As a first stage in this direction, we propose the creation of a simple, specialized type of agent, to which we will refer as Jacinta, who mediates between traditional web services and end-users. This mediator agent is invoked by human users who interact with it using a traditional web browser, so that the mediator (Jacinta) invokes traditional Web Services on behalf of the user.

Jacinta can be seen as a SA specialized in interaction with human users. The aim of Jacinta is not the creation of SWS or SA-SWS but the development of techniques for interacting with the end user. When SA-SWS becomes a reality, they could use Jacinta as a module to communicate with end-users.

An example of a traditional Web Services invoker that does not take into account end-user aspects is BPEL4WS[1]. This system does not provide support for human end-user related issues. It is designed to be used only by software programs (agents).

Another example is WSMO[2], where you have the same limitation: it is not targeted at human users, and you have to provide some kind of plug-in for dealing with user interaction, preferences or characteristics. Even, the way of showing the result is not defined, probably because this task should be carried by other agent.

Most efforts in the area of (semantic) web services to date have focused on the communication between agents, without considering that at the beginning and the end of the whole process there will be a human user.

2 Goals of the Jacinta System

The main objective of the Jacinta System is to cover the deficiencies found. It can invoke external Web Services, as the initiatives mentioned, but it's not its main objective. Those systems are much better designed for the orchestration (case of BPEL4WS) or the semantic matching of the goals and the Web Services (case of WSMO). Jacinta aims to provide support for the Human-Agent interaction, providing solutions such as:

- Dealing with the physical characteristics of the interaction user device: web navigators running in very different visualization devices, or not web at all, as are the applications for PDA (infrared protocols), mobile phone (GPRS, UMTS) or other small devices (BlueTooth).
- The aesthetic can be delegated to specialized enterprises and the human user could choose most appropriated or fashionable.
- For every concept in Jacinta, it knows how to show it to the user and how to interact with the user to get all the needed data.

[1] http://www.alphaworks.ibm.com/tech/bpws4j
[2] http://www.wsmo.org

- For every task that the user can solicit, Jacinta will know the Web Services involved and will ask to the user the needed input data, will invoke the services, and will show the results. Even, if the amount of result is excessive, the system will make some hints to the user for getting a more appropriated result.

3 Architecture and design of the Jacinta System

Jacinta is a Web application running in Tomcat, so it can be easily ported to any other Web Server. All the management of the system is done via web. This is too the default user device supported by the system. Users via different protocols, as is the case for PDA or mobile phone users, are considered but are not supported yet. However, it is very common that PDA or mobile phones include web navigators, as is the case of WAP mobiles.

Jacinta administrators can create tasks to be used by humans. These tasks will invoke the web services provided by many companies. Jacinta administrators will have web tools for creating these tasks, define the interaction and the visualization(s), and bind all this with the references to the WSDL files and the concepts defined in the ontology of the system.

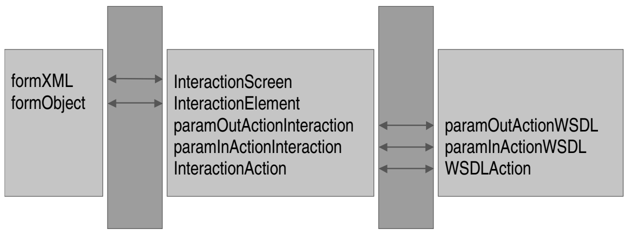
The system uses AXIS as SOAP engine for invoking the external Web Services.

Jacinta uses Jena 2 for the management of its own ontology. This ontology is designed for the definition of the task that the user can solicit and the involved concepts. This ontology is currently written in RDFS due to these reasons:

- Simple concepts yet.
- Queries are made in RDQL (provided by Jena2) and are limited to RDF data.
- Basic reasoning requirements.

The ontology includes classes like "SWSConcept", "SWSImplementation" among others. The "SWSConcept" concepts are the tasks that will be offered to the end user. Every task will involve one or more Web Services provided by one or more external Enterprises ("SWSImplementator" class). The system administrators will have to add all the binding needed to include new Web Services provided by new enterprises. No automatic matching is provided right now. The system administrators will bind the messages needed for the web Service invocation with the concepts stored in the ontology. Every "SWSConcept" will use some related concepts that will have to be added to the ontology. For example, if we define the "Solicit Mortgage" service, we will have to add to the system references to the WSDL files offered by these companies for this task, and we will have to define concepts as "Euribor", "Payment", "Geographical Location", etc.

One key point is that visualization and interaction are not part of the ontology. This data are used as XML strings stored in the ontology, but there is no any "inputField" class or similar. Our aim is to keep the ontology concepts away from the visualization or even interaction. The relationship will be established by means of binding mechanisms. The next figure shows these relationships.



Visualization-Interaction Binding **Interaction-Concept Binding**

The motivation for this decision is to get the most flexible visualization capabilities knowing that most visualization details, such as aesthetic ones, are not relevant for the interaction. Besides, the interaction is linked to the task details, providing a navigation model, but is loosely linked to the visualization. The specification of visualization and interaction models in terms of an XML Schema is enough for our objective. Besides, this representation allows the easy development of editing tools. For example, we can make an editing tool to create interaction models for each service provided, useful for Jacinta administrators. Furthermore, we can make an editing tool for creating different visualization models, useful for specialized enterprises.

Although there are many specialized models in XML for visualization (XUL[3], XFORMS[4], etc.), we prefer to use our own models from scratch. Once the system be stable and tested, we will consider the adoption of the most suitable standardized specification, if any suits our system.

4 Conclusions

Other efforts in this direction [Kassoft *et al.*, 2003; Petrie *et al.*, 2003] have been made, but the results are not very clear. Our viewpoint focuses on how to assign to every ontology concept the abilities for visualization and data request in the most flexible scenario, and how to interact with the end-user in the most complex cases in Web Services invocations.

Our aim is to publish the source code as soon as it will be stable at SourceForge.net.

References

[Kassoff *et al.*, 2003] Michael Kassoff, Daishi Kato, and Waqar Mohsin. *Creating GUIs for Web Services.* IEEE Internet Computing Press, September-October 2003.

[Petrie *et al.*, 2003] Charles Petrie, Michael Genesereth, Hans Bjornsson.... "Adding AI to Web Services". *LNCS Volume 2926 / 2004*: pp. 322 - 338, March 2003.

[3] http://www.mozilla.org/projects/xul/
[4] http://www.w3.org/MarkUp/Forms/