Semantic Web and Ubiquitous Computing - Task Computing as an Example -

Ryusuke Masuoka, Yannis Labrou, and Zhexuan Song
Fujitsu Laboratories of America, Inc.
College Park, Maryland, USA
{rmasuoka, yannis, zsong}@fla.fujitsu.com

Introduction

"The Semantic Web and Ubiquitous Computing are quite a synergistic match." This is a lesson we have learned through two and half years of research and development of the Task Computing technology. Ubiquitous Computing is a domain that is amenable to the application of Semantic Web technologies and Ubiquitous Computing needs Semantic Web technologies to deal with its inherently ad-hoc relationships between users, devices and services. In this short article, we describe Task Computing (TC), as one of the first realizations of Semantic Web technology in this domain (for details, see [1, 2, 3] or Task Computing site, http://taskcomputing.org) and then discuss our thesis.

1. Task Computing

Task Computing is a result of close collaboration between Fujitsu Laboratories of America (FLA, http://www.flacp.fujitsulabs.com/) and the MINDLab (http://owl.mindswap.org/) of the University of Maryland. Task Computing is a user-oriented framework that lets end-users accomplish complex tasks in environments rich with applications, devices, and services. Task Computing provides many ways for users to interact with these ubiquitous environments and applies the Semantic Web technologies, such as OWL (Web Ontology Language, http://www.w3.org/2001/sw/WebOnt/) and OWL-S (http://www.daml.org/servcies) as its core enablers.

Imagine that you can display your presentation file from your mobile computer on the projector in the room you visit for the first time without connecting a VGA cable. Or that you can show the photo that you have just taken with your digital camera on the photo frame in the same room immediately and print it on a photo printer without moving memory cards around. Or, that you can display the current weather at an address in your PIM (Personal Information Manager) on the projector with just a few operations of point and click. Simple wireless connectivity does not give solutions for such dynamic combinations of first-encountered functions from disparate sources. Task Computing enables end-users to accomplish all of the above and more through a simple graphical user interface (see **Figure 1**) to the Task Computing environment. You can even use your own voice to make those same things happen through a voice-based Task Computing client, VoiceSTEER.

In the TC, we have created eight kinds of Task Computing clients and more than fifty kinds of Task Computing services, which allow the users to interact with the ubiquitous environment in many modalities (textural, graphical, voice, spatial) and let the user accomplish hundreds of high-level tasks through real-time composition of those services.

Task Computing accomplishes these goals through an architecture (see **Figure 2**) that provides a foundation for finding the services available in the current environment, constructing and manipulating a user-centric task view of the available services, and executing the resulting tasks composed of multiple services. It even lets the end-users dynamically and easily create new services as necessary.

2. Utilizing the Semantic Web in Ubiquitous Computing

We first discuss how Semantic Web technologies are used in Task Computing.The main challenge for Task Computing is the dynamism of a ubiquitous environment. Obviously, a major source of the dynamism stems from a user's movement from one environment to another (home, to office, to airport, to mall, etc.). In different places, the user finds a different set of services, many of which are unique to the place. Another source of this dynamism comes from the constant changes of entities in the environment. Even if the user stays in one environment, the changes in the surrounding people and devices alter the status of the environment.

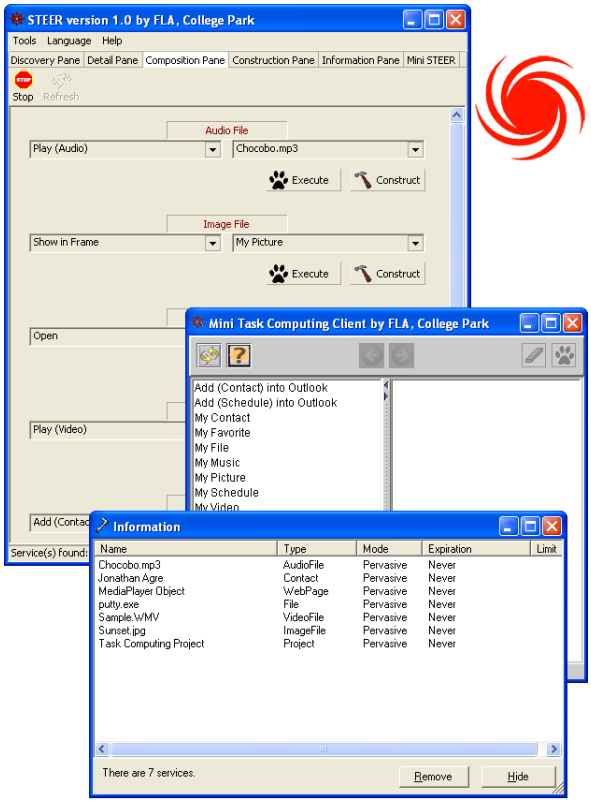

Figure 1 *Task Computing Client Desktop: The figure shows two kinds of Task Computing Clients (tab and mini versions), White Hole, a tool to create services from objects, at the upper right corner, and PIPE, a tool to manage the semantic services, at the bottom. For details of those GUI elements, see [3].*

The specific challenge of this dynamism is how to provide maximum interoperability with as few pre-arrangements as possible between the currently available functionality and the clients who use that functionality and yet, still give the user a manageable view of the possibilities in terms of what one can do in the environment. Although it is impossible to make them fully interoperable without any pre-arrangements, it is imperative to minimize pre-arrangements because the user does not know what to expect in the environment that one has entered or will move into next. Task Computing addresses this problem through the following two key ideas:

Uniform abstraction of all functionality as services

Abstraction of functionality as services makes functionality universally accessible and allows the Task Computing infrastructure to interact with it. The current wide acceptance of SOA (Service Oriented Architecture) in many areas (Web Services, UPnP, .NET, etc.) and the convergence to Web Services, make "services" the standard abstraction of functions. In Task Computing we transform the functionality of the user's computing device (from applications and OS), of the devices in the environment, and of the available e-services on the Internet into services. This abstraction paves the way for having fewer pre-arrangements to deal with the functionalities available in the environment, but by itself does not suffice for end-user, real-time manipulation and composition of functionalities into tasks.

Intuitive manipulation of services based on semantic service descriptions (SSD's)

The other necessary ingredient is the intuitive manipulation of services by end-users made possible through the use of Semantic Service Descriptions (SSD's); ontologies are the mechanism for achieving such a manipulation. Typically, WSDL (Web Service Description Language) is used to describe the functional characteristics of a Web Service; but using WSDL-described Web Services requires that programmers understand their semantics (beyond the WSDL descriptions) and develop the code to use the services in the right way. As a result, end-users' interaction with functionalities is limited by the scope of these programs in ways predefined by the developers. The additional semantics (supplied in an SSD), allows the Task Computing infrastructure to help users manipulate the services without this deep knowledge. Semantics can be used to constrain the manipulation of services by users, or to present the user possible tasks in the current environment. For example, service composition based on semantic inputs and outputs of services, if relied on WSDL only, would not restrict any compositions of a service that produces an XSD string with another one that consumes an XSD string. However, the SSD's of services provide finer granularity of the services inputs and outputs, so that, for example, a service that generates an "Address" semantic object will only be composable with semantically compatible services.

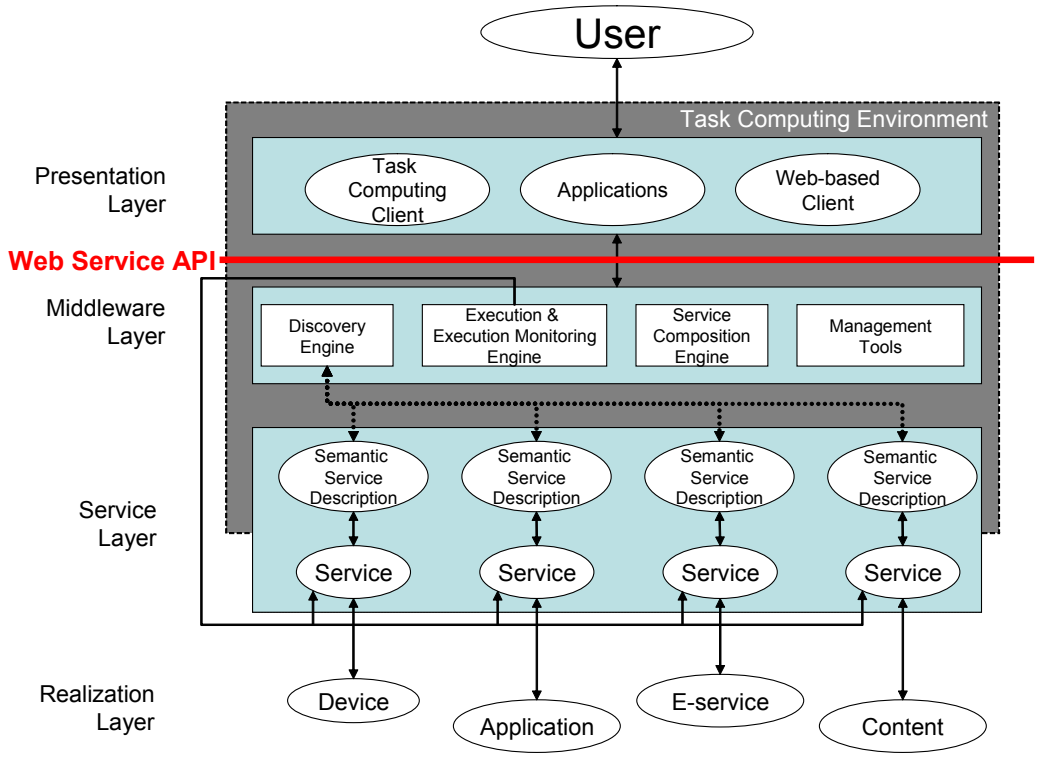

Figure 2 Task Computing Architecture: It has a four-layer architecture. For details, see [3].

By giving appropriate service names, the composed service names of compatible services can serve as task representations (ex. "View on Project" + "My File", "Route from FLA to" "BWI Airport" [1]). Ontologies can also support mechanisms such as compositions based on subclass-super-class relationships and semantic object translations that are very natural for end-users. At this point, Task Computing does not utilize preconditions and effects of services to determine service composability, as the standards for expressing them are still fluid and the experience in them is lacking in the broader Semantic Web community. Even so, Task Computing still allows very rich and interesting ways for the end-users to interact with the services of the environment.

3. Ubiquitous Computing as an Application Domain of Ontology

In this section, we will discuss why the Ubiquitous Computing environment is an easier domain to apply Semantic Web technologies to than other application domains, such as the World Wide Web or Enterprise Computing.

End-user functionality is grounded to physical reality

In Ubiquitous Computing, the things that users do (tasks), are based on the functionality of physical devices (printers, projectors, cameras, etc.). The semantics of such functionality is easier to capture than that of information, because this functionality has a limited scope (domain) which is intuitively shared by all users. In other words, the implicit semantics of that functionality is relatively constrained and well-understood by its intended users.

Each semantic description can be applied to thousands (or even millions) of entities

In Ubiquitous Computing, descriptions about a device based on a relatively stable ontology can be reused as many times as the number of the devices of that kind. This high-rate of reuse is not the case for ontologies on the WWW, where each document may require its own ontology or in enterprise applications where application customizations need to be reflected at the semantic layer, in turn necessitating complex ontology translation.

Another advantage of Ubiquitous Computing is the rate of change of ontologies. Because the ontologies describe functionality that is rooted into physical embodiments (devices) which do not change that frequently (consider the slow rate of change of a printer's functionality over the last decade), Ubiquitous Computing is relatively stable over time. Such adaptation is much more frequent in other domains.

Manageable cost of ontology development

The cost of ontology development is another important issue and this is closely related to the previous point. Because manufacturers produce a large number of devices for Ubiquitous Computing environments, they can afford to put some serious effort to create quality ontologies for their devices as long as they see the value in providing them. In return, these ontologies can be reused many times for the same kinds of devices.

We believe a few consortia will be formed to standardize the ontologies for their devices. Even so, we do not believe that a single ontology will rule the world. Like Beta vs. VHS for the video tapes, there are going to be a small number of competing consortia, but even then, the problem is manageable by relatively simple ontology mappings. Such translation services can be used in the Task Computing framework in a way that is transparent to the end-user (see [2]).

4. Conclusion

One of our goals for Task Computing is to bring the values and benefits of Ubiquitous Computing environments to end-users. We believe that Task Computing is a happy marriage between the Semantic Web and Ubiquitous Computing. Semantic Web technologies have been quintessential in accomplishing our goal. As Prof. Jim Hendler has pointed out in many occasions [4], "A little semantics goes a long way." We started with relatively shallow semantics and have gradually incorporated more semantics in order to provide increasing values to end-users. The Task Computing software is available to academic institutions for research purposes. If interested, please visit the Task Computing site, http://taskcomputing.org.

References

[1] Masuoka, R., Parsia, B., and Labrou, Y., "Task Computing - The Semantic Web meets Pervasive Computing -." In D. Fensel et al. (Eds.), "The Semantic Web - ISWC 2003," the Second International Semantic Web Conference (ISWC 2003), Sanibel Island, FL, USA October 2003 Proceedings, LNCS 2870, pp. 866-881, 2003.

[2] Masuoka, R., Labrou, Y., Parsia, B., and Sirin, E., "Ontology-Enabled Pervasive Computing Applications," IEEE Intelligent Systems, vol. 18, no. 5, Sep./Oct. 2003, pp. 68-72.

[3] Song, Z., Labrou, Y., and Masuoka, R., "Dynamic Service Discovery and Management in Task Computing," pp. 310 - 318, MobiQuitous 2004, August 22-26, 2004, Boston, USA.

[4] J. Hendler, "On Beyond Ontology", Keynote talk, International Semantic Web Conference, 2003. Available online from http://iswc2003.semanticweb.org/hendler_files/v3_document.htm