Task Computing for Ubiquitous Multimedia Services

Zhexuan Song Ryusuke Masuoka Jonathan Agre Yannis Labrou
Fujitsu Laboratories of America, Inc.
8400 Baltimore Avenue, Suite 302
College Park, Maryland 20740-2496, USA
{zsong, rmasuoka, jagre, yannis}@fla.fujitsu.com

ABSTRACT

The Task Computing framework is designed to operate in dynamic ubiquitous environments in which a mobile computing user dynamically discovers the current set of available semantically defined services. Task Computing allows the end-user to easily and seamlessly integrate and manipulate services found on their own computer, the nearby devices and relevant remote web services. The user can create, execute and monitor complex tasks resulting from compositions of these services. In this paper, how Task Computing addresses some issues in mobile and ubiquitous multimedia applications and some of the multimedia related semantic services currently implemented in this framework are described.

Keywords

Task Computing, Semantic Web Service, Service Discovery, OWL-S, UPnP

1. INTRODUCTION

To realize true mobile and ubiquitous multimedia applications, we need to tackle a new set of problems, such as:

- In the very near future, a user's high mobility plus the ubiquity of multimedia services will lead to constant change of the environment the user is in and of the ad-hoc relationships between the user and the current multimedia services. Service discovery mechanisms such as UPnP [1] definitely will help to determine the entities (multimedia services) found in the user's environment at each particular moment, but that is not sufficient to provide the user with an agreeable experience that enables productivity.
- Chances are high that the user meets many of the services for the first time. In order for the "system" (the user's client software) to provide the user with useful and easy to understand views of the environment, the system needs to find additional information about those services, and that should occur with as little pre-agreement as possible between the system and the services.
- Given this additional information about the services, the system can help to determine their relevance to the user as well as the relationship between the services to provide a richer view of the user's environment. For instance, for video playing services, additional information, such as name, description, the supported format, resolution, type of the displaying device, and the location, helps user a lot in finding the right service for a given video.
- Another issue arises due to the fact that information sources and applications on PC's and the Internet are already ubiquitous. If the system can integrate such information and applications seamlessly with the mobile and ubiquitous multimedia services, it would improve the system's utility tremendously.
- Once the system has the capability to deal with the dynamic and changing environment, we can use this particular ability to our advantage. By changing the in situ environment, the user can use the environment itself as the medium to communicate with other users. Basically the user creates (or publishes) services dynamically to share information and/or multimedia files. This service-publishing capability can be embedded in the environment itself as services. With such services, the user can share the information and multimedia files with others even after the user leaves the environment.

In order to deal with these problems, service discovery mechanisms should be combined with service semantics based on standards (so that there are minimal pre-agreements.) Based on the semantics of the services found, user-centric views of the environment may be constructed dynamically. Ultimately, the system needs to execute the services to accomplish the tasks specified by the user. Our solution for such a system is Task Computing.

Task Computing [2][3][4] is a user-oriented framework that lets non-expert users accomplish complex tasks in application-, device-, and service-rich environments. Task Computing provides a myriad of ways for users to interact with and through ubiquitous environments.

In this paper, we will describe how Task Computing addresses those issues described above.

The rest of this paper is organized as follows. In the next section, we will discuss the insights and concepts of Task Computing; in Section 3, we present in detail some multimedia producing and consuming services and in Section 4 we describe a possible scenario, known as e-Home. Finally, Section 5 concludes the paper and suggests some future works.

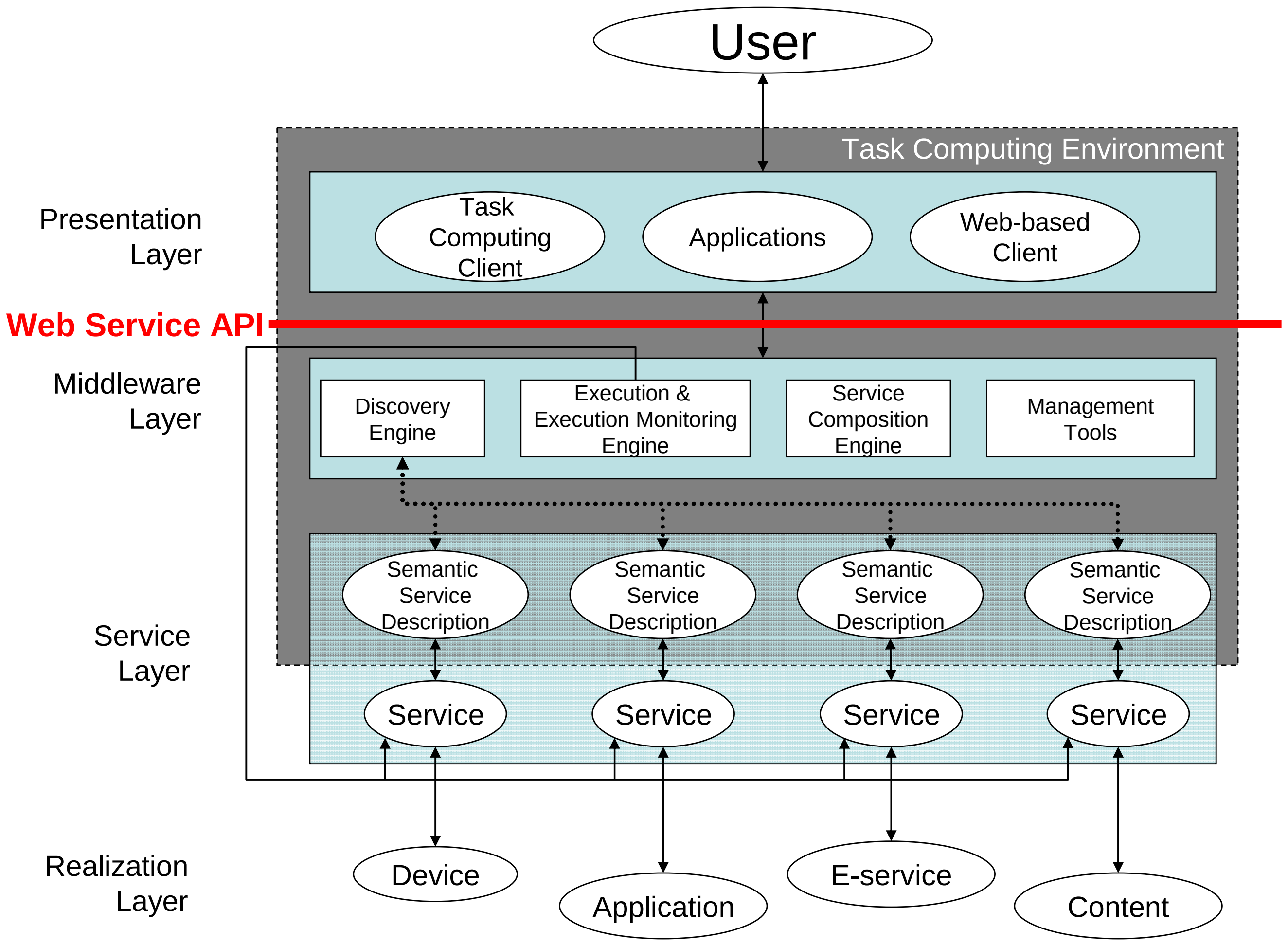


Figure 1 Task Computing Architecture

2. TASK COMPUTING AND MOBILE AND MULTIMEDIA APPLICATIONS FOR UBIQUITOUS ENVIRONMENTS

In this section, we describe how Task Computing helps building mobile and ubiquitous multimedia applications.

Figure 1 illustrates the Task Computing architecture with its four distinct layers.

• [REALIZATION LAYER] The bottom layer encompasses the universe of "services" including multimedia producers and consumers, where all functionality available to the user originates.

• [SERVICE LAYER] These various sources of functionality are made computationally available as semantic services, in the sense that service interfaces containing semantic descriptions are employed to access (execute) this functionality.

• [MIDDLEWARE LAYER] This layer provides components for discovering, composing, executing, monitoring, managing, and publishing services through its Web Service-based API.

• [PRESENTATION LAYER] The top-most layer uses the capabilities of the layers below to provide the user with a "Task" abstraction of the user's environment and lets the user interact through this semantic-level abstraction.

This architecture enables Task Computing to address the problems described in Section 1.

First, Task Computing employs a universal abstraction of functionality as a "service." Any function, whether it is multimedia producers or consumers, or it is from applications on PC's or the Internet, is treated as a "service" in Task Computing. Even a piece of information or a file is made into an information- or file-providing service.

We adopted this "service" abstraction because of recent wide acceptance of the SOA (Service-Oriented Architecture) and its advantages such as transparency of the location of the service and standardized access. Many of the functionalities are readily available through one of the SOA frameworks such as Web Services, UPnP, and .NET. It is crucial to adopt widely accepted frameworks for mobile and ubiquitous applications as they need to incorporate functionalities from many sources.

The "service" abstraction is also generic and flexible enough to cover not only multimedia functionality, but also information and applications on PC's and the Internet.

Task Computing can handle Web Services and UPnP Actions readily and can incorporate many other SOA frameworks as well as remote object architectures. Currently, the majority of the services in Task Computing are wrapped with Web Service interfaces, described by WSDL [5], which provides a standard and easy way for users to retrieve or submit a multimedia object. However, the descriptive capability of WSDL is still very limited. Many forms of descriptive information, such as the type of media the service generates, are missing. Unfortunately, this type of information is very important for users to determine what the service is about and how to use the service.

This leads to the second and very important point of Task Computing, which is "semantics" of the services. To be exact, all the services in Task Computing are semantic services, in the sense that they are accompanied with their semantic service descriptions (SSD's).

The SSD's in Task Computing are encoded in the OWL-S language [6], which is a description language for semantic services based on the Web Ontology Language (OWL) [7]. With OWL-S, an SSD can have richer information such as service names, descriptions, semantic input/output, service properties and grounding information that relates the semantic level to the service implementation in the realization layer.

Using the SSD's of the services, Task Computing Clients (TCC's) can compose services semantically and present the service compositions as "tasks" to the user. As the SSD is based on ontologies described by OWL, TCC's can compose the services more flexibly and in a semantically richer way than WSDL enables.

Figure 2 shows a small branch of the ontology we use in Task Computing (see [8] for complete ontology), which is related to file objects. *Media File* is a sub-class of *File* class. The *Media File* includes common properties of multimedia object. It has three sub-classes*: Image File* which includes all formats of images*; Audio File* which includes both audio files of different format and audio streams; and *Video File* which includes both video files of different format and video streams.

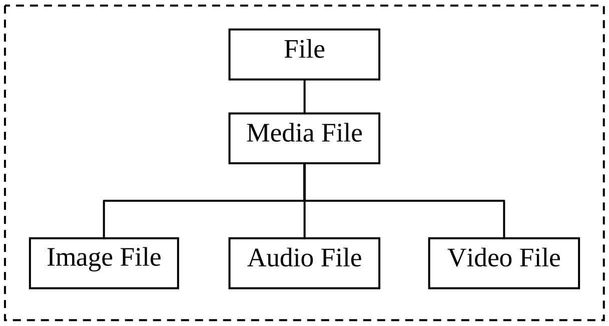


Figure 2 Part of the ontology tree used in Task Computing

TCC's can compose the services based on this ontology. For example, *Image File* consumer and provider services, "Show in Frame" and "My Picture" respectively (see Section 3) can be composed and presented to the user as "Show in Frame My Picture." There will not be presented a composition such as "Show in Frame My Music" since "Show in Frame" and "My Music" are semantically incompatible. Another example showing flexibility is the composition of "Open" "My Picture" and "Open" "My Music." As "Open" is a *File* consuming service, it can be composed with any *File* subclass-providing services in TCC's. TCC traverses the ontology tree at run time to determine if these services are compatible. In general, this allows more flexibility then traditional type checking.

This is also totally different from the pre-programmed semantics such as used in UPnP where one needs to know in advance the semantics of the UPnP devices in order to use them. Programmers need to pre-program all the logic to use such devices. On the other hand, with this description-based semantics, the semantics are very easy to share as it can be materialized in files. To share the semantics is simply to share the file. The semantics can be used in any other system if the other system can parse and understand the file.

This, in turn, leads to the third point of Task Computing, which is "virtualization." The Middleware Layer, the third layer of the Task Computing Architecture, provides its functionality (discovering, composing, executing and publishing services) based only on the SSD's of services. In the middle layer, all the services are virtualized by their SSD's. For example, discovering a service in Task Computing is basically obtaining its SSD through one of the discovery mechanisms. This virtualization gives Task Computing the prowess necessary to deal with the dynamically changing environments. After obtaining the SSD's, a TCC lets the user manipulate the services at his/her fingertips.

This virtualization also makes managing and publishing services very easy. Publishing a service can be done by creating the Web Service or UPnP Device and providing its SSD through one of the discovery mechanisms such as UPnP. After that, managing the service is reduced to managing the SSD. For example, you can make a service unavailable to others just by making the SSD undiscoverable.

We have created many unique and interesting functions in Task Computing allowing a user to dynamically publish and manage the services (see [4] for details).

"PIPE" is a common module in Task Computing framework to publish semantic instances. For semantic instances, PIPE first dynamically creates a web service, which returns the semantic instance as its output when invoked; next, a semantic service description for the newly created service is generated. During this process, the name, description, output type, and grounding details of the service are determined and described in a high level (in OWL-S); and finally the semantic providing service is published.

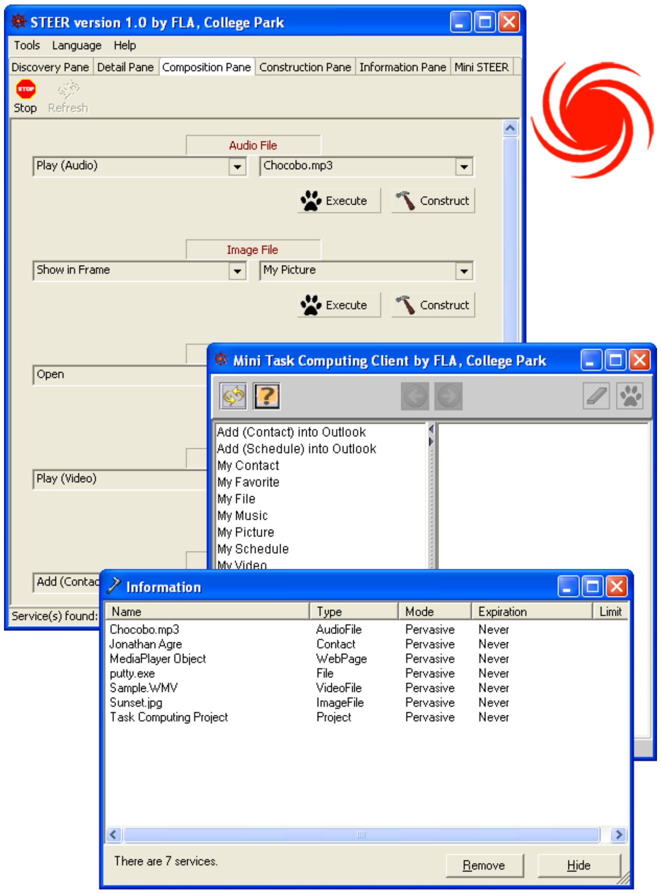
The "White Hole" (the swirling icon at the upper right in Figure 3) is a tool for the user to dynamically publish services from OS/application objects (such as multimedia files from the file system of the local operating system (OS) on a computer and Contact and Schedule items from PIM application). When the user drags and drops an OS/application object into the White Hole, a service is created on user's machine, which provides the semantic equivalent of the object.

A "Bank" is a standalone service, which allows a user to deposit an item (any semantic object) in the environment and publishes a service providing the semantic item. When a user moves to an environment with a "Bank" service, the user can use it to leave any semantic object including multimedia objects in this environment in the form of an object- providing service. The service is persistent even after the user leaves the environment, making it possible for the other users later coming into the environment find and retrieve the object.

A "Media Publisher" service described in Section 3 is another example of the use of this dynamic publishing.

3. MULTIMEDIA PRODUCING AND CONSUMING SERVICES

All "frameworks" are just pie in the sky if there are only a few services (or functionalities) that can be used in them. In order to understand the real issues of this framework, it is necessary to implement many services and many kinds of clients.



Play (Audio)
Chocobo.mp3
Show in Frame
My Picture
Open
Play (Video)
Add (Contac

Add (Contact) into Outlook
Add (Schedule) into Outlook
My Contact
My Favorite
My File
My Music
My Picture
My Schedule
My Video

Name	Type	Mode	Expiration	Limit
Chocobo.mp3	AudioFile	Pervasive	Never	
Jonathan Agre	Contact	Pervasive	Never	
MediaPlayer Object	WebPage	Pervasive	Never	
putty.exe	File	Pervasive	Never	
Sample.WMV	VideoFile	Pervasive	Never	
Sunset.jpg	ImageFile	Pervasive	Never	
Task Computing Project	Project	Pervasive	Never	

There are 7 services.

Figure 3: Task Computing Environment Desktop: Two TCC's (Tab and Mini versions), White Hole at the upper left corner, and PIPE (Service Management Tool) at the bottom are shown.

So far, we have implemented nine Task Computing clients using different modalities and more than 50 kinds of services. Some of the semantically-described multimedia services that have been implemented are introduced in this section.

3.1 Multimedia Providing Services

The multimedia providing services accept no input. Once invoked, they generate *Media File* type semantic object instances.

- "My Pictures/My Music/My Videos" allows user to select an image/audio/video file from their local machine and generate an *Image File/Audio File/Video File* instance.
- "Image generator" allows a user to operate available image grabbing devices (such as Scanner, DV) and generates an *Image File* instance.
- "Media Publisher" is a tool to help a user to publish media files from plug-and-play devices such as USB flash memories, CD, and DVD. For example, after "Media Publisher" detects that a user plugs a digital camera into the computer, it will dynamically create and publish a service called "Picture from the Camera". Once invoked, the newly created service will allow users to select a picture from the camera memory and generate an *Image File* instance.
- "Semantic Streaming Service" encodes the real-time contents generated by a camera and wraps it into a *Video File* instance.

3.2 Multimedia Consuming Services

Multimedia consuming services are services that accept a multimedia instance and present them to users.

- "Show in Frame" service takes an *Image File* instance as input and shows it in an electronic digital photo display. The service also provides a Web-based user interface which allows the user to manipulate the image, such as zoom in/out, rotate, etc.
- "Play (Audio/Video)" service takes an *Audio File*/*Video File* instance as input and plays it. Similarly, a Web-based user interface is provided.
- "Tell Me" service takes any semantic instance as input and serializes it into a human-understandable phrase and then reads it out.

4. AN E-HOME SCENARIO

With the semantic services described in Section 3, many interesting and truly useful tasks can be realized through composition.

Imagine that Person A is at her home with her friend, Person B, and there are Task Computing-enabled devices and services and Task Computing Clients on their PDA's (Figure 4) in this ubiquitous environment. Specifically, there is Person A's PDA, a local computer, a digital picture frame, a digital camera, a video surveillance camera, a monitor, a stereo and a local network consisting of wired and wireless connections of these devices and a wireless access point. Person B has a PDA with a wireless capability that can connect to the access point. Suppose for example, Person A takes some pictures with her digital camera and plugs the camera into her local computer. Immediately a service which provides the photos from the camera is created by "Media Publisher". Without tedious copy-paste actions, she simply uses the Task Computing Client running on her PDA to construct a task composed from basic services called "Show in Frame" "Picture from the Camera" to display the selected photo in the digital photo frame in her room. Her friend, Person B, can share the photo by executing the task, "Save" "Picture from the Camera" where "Save" is a service on Person B's PDA which saves the object as a file on the PDA.

Person A hears the doorbell ring and she uses her PDA to build another task: "Play (Video)" "Entrance Video" ("Entrance Video" is a semantic streaming service described in Section 3) to forward the live video from the surveillance camera at the home entrance to the monitor in the room.

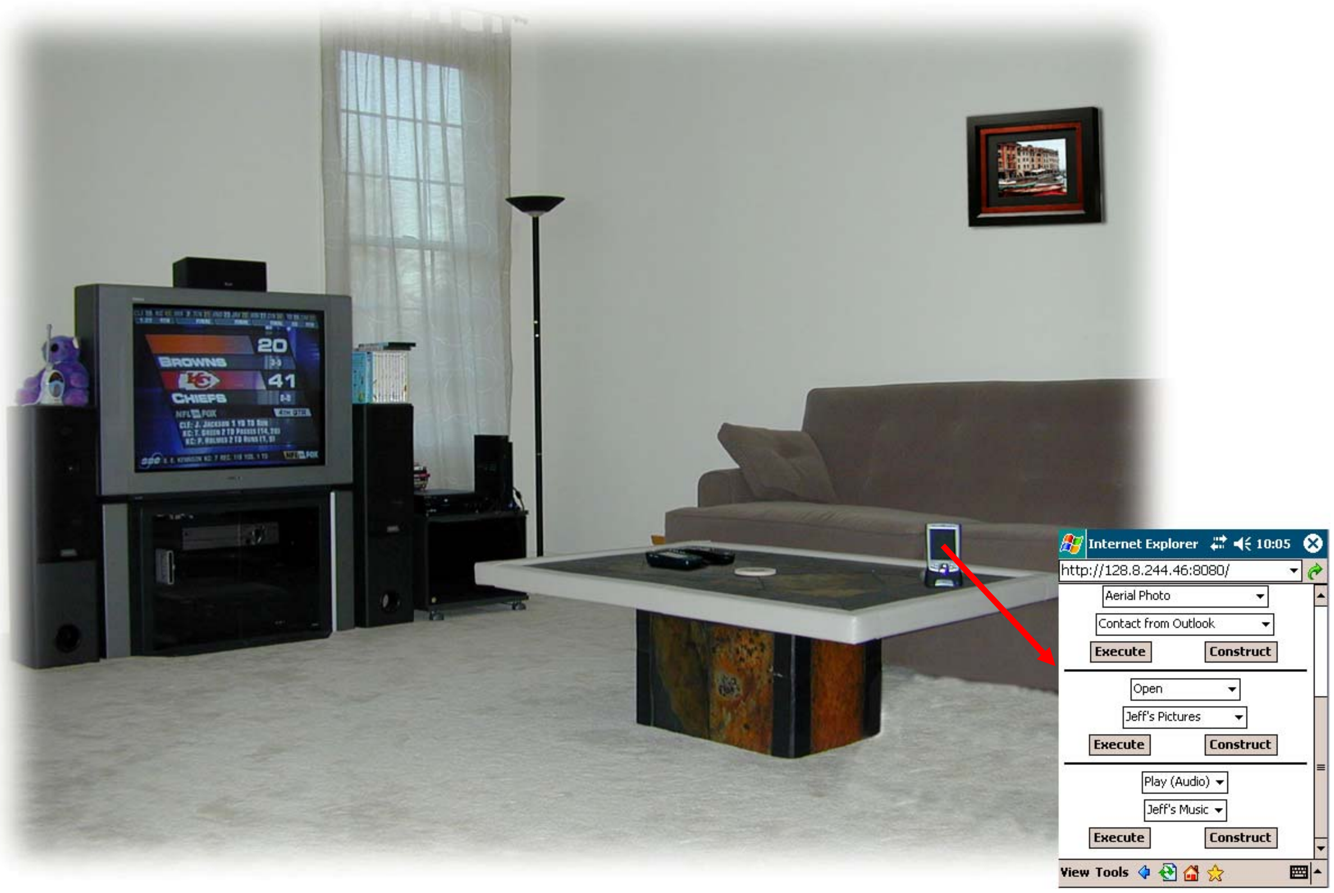


Figure 4 Task Computing in Home: Many Task Computing services such as "Play (Video)" for the monitor, "Play (Audio)" for the stereo system, "Show in Frame" for the digital photo frame are there. "Entrance Video" for the security camera at the entrance and many services from the PC on the second floor can not be seen in the picture, but are available for Task Computing clients in the living room.

Her friend, Person B, can play a piece of music from B's PDA on A's stereo by creating and executing the task "Play (Audio)" "My Music" to play the music on the stereo in the room. If Person A likes it, B can share it by a drag-and-drop of the file into the "White Hole" running on B's PDA. Immediately a service, "Song X," (let us assume the filename is "Song X" so the "White Hole" automatically gives the service the name same as the filename.) is available and Person A uses task "Save" "Song X" to save the song on A's local computer.

The important point is that none of the above scenarios has been pre-programmed. All the tasks are realized as service compositions that are dynamically created and immediately executable by Task Computing Clients, like the ones in Figure 3 and Figure 4. In addition to multimedia services, all functionality is available for manipulation in a uniform fashion, greatly reducing the user's learning curve and increasing the utility of the provided services.

Another advantage of the architecture is that the presentation layer is separated from the middleware layer allowing various forms of presentation including graphical and voice-actuated. Using our voice-based Task Computing client one can command by just talking to it, using phrases like "Computer, Show in Frame Picture from the Camera" or "Computer, Play Entrance Video."

5. CONCLUSION AND FUTURE WORKS

We envision Task Computing as a very useful and powerful framework in ubiquitous/pervasive environments like hospitals, offices, and homes where the end-user can seamlessly integrate and manipulate functionalities on their own computer, the surrounding devices and remote web services; enabling one to easily and dynamically define, execute and monitor complex tasks, in ways that can only be accomplished today by painstaking, design-time integration provided by experienced programmers.

In this paper, we briefly discussed applications of Task Computing in ubiquitous environments and highlighted some multimedia services. We discussed how Task Computing can address the problems that the mobile and ubiquitous multimedia applications will present. We also introduced a few of the multimedia related services currently implemented in our Task Computing environment.

The Task Computing environment is an ongoing research project. Our future work includes increasing the degree of automation (or assistance to the user) during task creation (e.g., service composition); utilizing some planning capabilities in the inference engine in our system; and addressing various aspects of security, and incorporating web services security-related standards and semantics-based security policy work [9]. Also, it would be very helpful to know the limits of dynamics caused by mobility, and the scalability of services after a series of measurements and additional practical experience.

A demonstration of Task Computing for Mobile and Ubiquitous Multimedia is scheduled at the MUM 2004 conference. For more information about Task Computing, including the latest development, please visit the Web site [10].

REFERENCES

[1] UPnP (Universal Plug and Play). http://www.upnp.org.

[2] R. Masuoka, Y. Labrou, B. Parsia, and E. Sirin. Ontology-enabled pervasive computing applications. IEEE Intelligent Systems, 18(5):68-72, September/October 2003.

[3] R. Masuoka, B. Parsia, and Y. Labrou. Task computing – the semantic web meets pervasive computing. In Proceedings of 2nd International Semantic Web Conference (ISWC), volume LNCS 2870, pages 8660881, Sanibel Island, FL, USA, October 2003. Springer-Verlag Heidelberg.

[4] Z. Song, Y. Labrou, and R. Masuoka. Dynamic service discovery and management in Task Computing. In proceedings of 1st International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous), Boston, USA, August 2004.

[5] Web Services Description Language (WSDL). http://www.w3.org/TR/wsdl

[6] OWL-S: Semantic markup for web services. http://www.daml.org/services/owl-s/1.0/owl-s.html.

[7] Web Ontology Language (OWL). http://www.w3.org/TR/owl-features/.

[8] Task Computing Environment (TCE) Object Ontology, http://www.flacp.fujitsulabs.com/tce/ontologies/2004/03/object.owl

[9] L. Kagal, T. Finin and A. Joshi. A policy based approach to security for the semantic web. In Proceedings of the 2nd International Semantic Web Conference (ISWC), 2003.

[10] Task Computing Project official Web site. http://taskcomputing.org