

# UPMSM: A Lifecycle Management Model for Ubiquitous Personalized Multimedia Services

Zhiwen Yu <sup>1,2+</sup>, Zhiyong Yu <sup>2</sup>, and Xingshe Zhou <sup>2</sup>

<sup>1</sup> Academic Center for Computing and Media Studies, Kyoto University, Japan

<sup>2</sup> School of Computer Science, Northwestern Polytechnical University, P. R. China

(Received December 26, 2006, accepted February 16, 2006)

**Abstract.** The ubiquitous computing environment and users' demand for multimedia personalization precipitate a need for ubiquitous personalized multimedia services (UPMSs). Delivering UPMS is a complicated task because of various user preferences, diverse device capabilities, and dynamic network characteristics. This paper proposes the use of FSM (Finite State Machines) to model the lifecycle of ubiquitous personalized multimedia services. The service context and specification, service state, state transition, service manipulation, service scheduling, as well as service composition strategy, are presented. The prototype experiments proved that by using this model, the management of UPMSs is very clear and easy, and the services are delivered more efficiently, effectively, and robustly.

**Keywords:** service management, service scheduling, service composition

## 1. Introduction

The rapid advance of communication technologies precipitate more and more embedded computing devices, such as PDAs, cell phones, etc., using in a wireless environment. This has lead to a shift from traditional computer-centered to human-centered information access mode what is known as *Pervasive Computing*. Multimedia information is widely used in pervasive computing environments among many application fields, such as multimedia digital libraries, home entertainment, and live camera remote surveillance. A major trend and requirement in today's multimedia service is personalization, which aims at providing multimedia objects according to user preferences. Multimedia personalization can provide what user wants directly, as well as relieve the transmission load by filtering the data irrelevant. The environment of ubiquitous computing (heterogeneous devices and networks) and users' demand for multimedia personalization precipitate a need for ubiquitous personalized multimedia services (UPMSs). Delivering UPMS is a complicated task because of various user preferences, diverse device capabilities, and dynamic network characteristics. It relies on many different technologies, such as media streaming, media transcoding, multimedia recommendation, context representation and exchange, etc.

QoS (Quality of Service) is regarded as an integral part within currently ubiquitous distributed multimedia applications, as [1] reveals. Processing QoS for multimedia delivery mainly involves: QoS assessing, QoS mapping, and QoS negotiation. The heterogeneous and dynamic nature of ubiquitous computing environment demands a flexible and intelligent framework for development and deployment of multimedia services. QCompiler [2] is a programming framework for quality-aware ubiquitous multimedia applications (UMAs), which are modeled with task-flow model. SMART [3] is a scalable middleware for QoS-aware ubiquitous multimedia service delivery.

Several systems have been developed for delivering personalized media content to ubiquitous devices. UP-TV [4] is an EU project aiming to broadcast personalized TV programs to various devices other than traditional television. In [5], the authors present a MPEG-based personalized multimedia content delivery system dealing with both user preferences and terminal/network capabilities. IBM [6] has proposed a solution for video summarization and personalization for pervasive mobile devices. Lemlouma and Layaida [7] discuss an approach of generating TV-like multimedia presentations that are adapted to the user preferences and limited environments.

---

<sup>+</sup> Corresponding author. Tel.: +81 090-9891-1552  
E-mail address: yu@ccm.media.kyoto-u.ac.jp.

Most of existing solutions mainly focus on service framework/architecture, multimedia description model, and media transcoding/adaptation mechanisms. However, they rarely put attention on service running status management, which is critical for a complex system like multimedia personalization in ubiquitous environments. Thus, a big challenge for ubiquitous multimedia service provider is to present an approach that can model the running process and manage the lifecycle of UPMS. In this paper, we propose a ubiquitous personalized multimedia service model (UPMSM) based on FSM (Finite State Machine). FSM is a technique that allows simple and accurate design of sequential logic and control functions, which has been widely used in designing computer programs, sequential logic circuits, and electronic control systems [8].

The strengths of the UPMSM lay in its unified model for personalized multimedia services in ubiquitous environments independent on particular implementation techniques, and its simplicity and easy computability on diverse devices. By using UPMSM, the management of UPMSs is very clear and easy, and the services are delivered more efficiently, effectively, and robustly. Further more, the UPMS state machine sets in where standards left personalization engines opened for competitive market forces. One example is MPEG-21 *Digital Item Adaptation (DIA)*, defining metadata for the adaptation of resources to device capabilities or usage preferences [9]. The adaptation engine and how descriptors are matched is not standardized. Exactly here the UPMSM sets in and describes an excellent model describing how such an adaptation engine can be implemented.

The rest of this paper is organized as follows. Section 2 introduces the context and representation of ubiquitous personalized multimedia service (UPMS). Section 3 describes the proposed ubiquitous personalized multimedia service model. The service state, state transition, and service manipulation are described in detail. Section 4 presents the service scheduling strategy. Section 5 illustrates the service composition under different conditions. The preliminary prototype implementation is presented in Section 5. Finally, Section 6 concludes the paper.

## 2. Service Context and Representation

Fig. 1 is the conceptual context diagram of ubiquitous personalized multimedia service (UPMS). The service context involves multimedia content, multimedia metadata, terminal profile, user profile, network profile, and personalized content. The arrows pointing to UPMS represent inputs of a service, while the arrow deriving from UPMS to user represents output of a service.

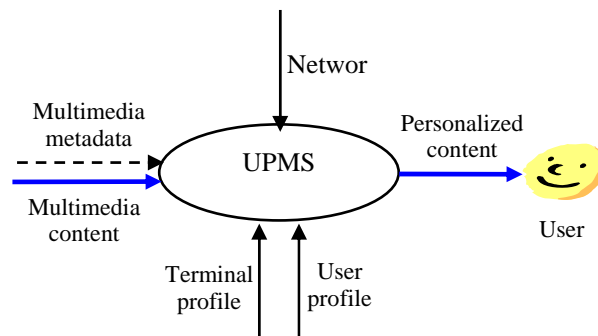


Fig. 1: Context of a ubiquitous personalized multimedia service

The following presents the definitions of UPMS specification.

**Definition 1 (UPMS Specification):** UPMS specification presents detailed information of the context of the service including service requirements, resources, inputs, and the output. It is independent with the service implementation. A UPMS Specification is a 6-tuple:  $(MM, MC, UP, TP, NP, PC)$ . *MM* (Multimedia Metadata) denotes multimedia description metadata; *MC* (Multimedia Content) denotes multimedia digital content; *UP* (User Profile) denotes user preferences about multimedia showing; *TP* (Terminal Profile) denotes terminal device capabilities; *NP* (Network Profile) denotes network connection characteristics; *PC* (Personalized Content) denotes ultimate personalized content user accesses. *UP* is the requirement of the service, and *MM*, *MC*, *TP*, and *NP* are resources of the service. *MM*, *MC*, *UP*, *TP*, and *NP* are inputs of the service, while *PC* is the expected output of the service.

Requirement and resources in the UPMS specification are defined as follows.

**Definition 2 (Service Requirement, UP),**  $UP=(p_1, p_2, p_3, \dots, p_k)$ . UP means user interesting values, such

as probability weights on  $K$  categories of multimedia, or querying keywords about multimedia description fields, e.g. actor, genre, etc.

**Definition 3 (Service Resource 1, MC)**,  $MC=(c_1, c_2, c_3, \dots, c_n)$ . MC includes  $n$  multimedia content. Each one maybe has difference variations, that is, different modalities or formats.

**Definition 4 (Service Resource 2, MM)**,  $MM=(m_1, m_2, m_3, \dots, m_n)$ . MM includes  $n$  multimedia description metadata. Each description metadata maybe contains descriptions for different variations of the same content if there are any.

**Definition 5 (Service Resource 3, TP)**,  $TP=(t_1, t_2, t_3, \dots, t_m)$ . Terminal profile is divided into  $m$  dimensions, e.g.,  $t_1$ =modality (e.g., video, image, text),  $t_2$ =format (e.g., mpeg-2, H263),  $t_3$ =frame rate (e.g., 30, 20, 10),  $t_4$ =frame size (e.g., 740\*480, 480\*360, 360\*240),  $t_5$ =color depth (e.g., 8, 4).

**Definition 6 (Service Resource 4, NP)**, Network profile is divided into two types: static capacities and dynamic characteristics. Network static capacity,  $NSC=(nsc_1, nsc_2, nsc_3, \dots, nsc_l)$ , is divided into  $l$  dimensions, e.g.,  $nsc_1$ =maxCapacity,  $nsc_2$ =minGuaranteed,  $nsc_3$ =errorDelivery,  $nsc_4$ =errorCorrection,  $nsc_5$ =inSequenceDelivery. Network dynamic characteristics,  $NDC=(ndc_1, ndc_2, ndc_3, \dots, ndc_j)$ , is divided into  $j$  dimensions, e.g.,  $ndc_1$ =AvailableBandwidth,  $ndc_2$ =minimum,  $ndc_3$ =maximum,  $ndc_4$ =average,  $ndc_5$ =interval,  $ndc_6$ =Delay,  $ndc_7$ =PacketOneway,  $ndc_8$ =PacketTwoWay. Network static capacities can be obtained easily. It is usually maintained by the terminal. Dynamic network characteristics are determined by a reliable detecting mechanism in the server on-the-fly.

### 3. Ubiquitous Personalized Multimedia Service Model

#### 3.1. Service State and State Transition

We deploy the FSM (Finite State Machine) to represent service state and manage running status of ubiquitous personalized multimedia services. A ubiquitous personalized multimedia service (UPMS) is modeled as a deterministic FSM.

**Definition 7 (UPMS State Machine)**, A UPMS State Machine is a 5-tuple  $UPMS=(S, I, f, s_0, F)$ :

- $S$  is a finite set of service states;
- $I$  is a finite set of elements, which is defined in Definition 8, each element is an input to the state machine;
- $f: (S \times I) \rightarrow S$  is the transition function;
- $s_0 \in S$  is the initial state;
- $F \subset S$  is a finite set of final states.

**Definition 8 (Input of UPMS State Machine, I)**,  $I$  is a finite set of elements as inputs to the state machine. Each element is a 2-tuple  $(R, E)$ .  $R$  represents the reason why the event  $E$  should take place. In each element,  $R$  can be NULL, while  $E$  cannot be NULL.

Fig. 2 shows the state machine model and complete state transition of UPMS. A service request from terminal user brings a new UPMS into birth. In its lifecycle, a UPMS could be in several states, represented by ellipses in the figure. The arrows between states represent state transitions, and corresponding character strings (I1, ..., I10) stand for conditions of the transitions.

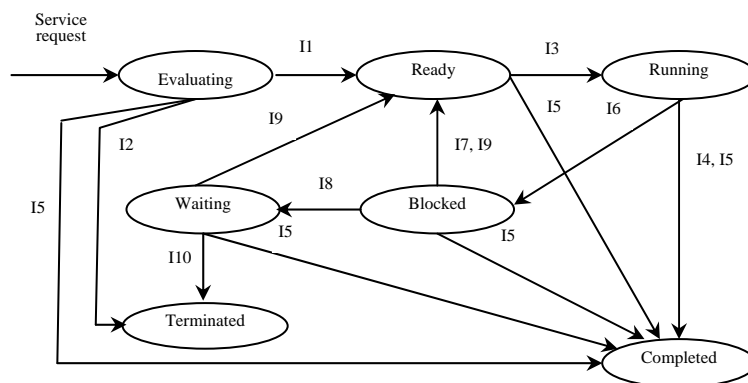


Fig. 2: UPMS State Machine Model

The element values of UPMS State Machine are presented in Table 1. The states in UPMS State Machine are described in detail as follows.

**Evaluating** When a client sends a request to the server for multimedia service, the service is in the *Evaluating* state. In this state, the server selects or generates a multimedia object according to user preferences, terminal capabilities, and network characteristics.

**Ready** When a multimedia object is assigned to a service, it jumps to the *Ready* state, and is ready for scheduling. If a service is selected to run, it jumps to the *Running* state.

**Running** Service in the *Running* state means media streaming or file downloading. In the process of running, (i) if the service finishes successfully or is broken by the user or service provider, it jumps to the *Completed* state; (ii) if network available bandwidth decreases, and cannot deliver the multimedia object, the service enters the *Blocked* state.

**Blocked** When a service is in the *Blocked* state, (i) it makes transcoding for the multimedia object; if the adjusted object can be delivered in the current network condition, then the service is woken up, and put into the *Ready* state; (ii) whatever transcoding is made, the object cannot be delivered in the current network condition, then the service jumps to the *Waiting* state.

**Waiting** When a service is in the *Waiting* state, (i) the network bandwidth increases, and is enough for the object to be delivered, then the service is woken up, and put into the *Ready* state, waiting for another scheduling; (ii) the preset waiting time is over, then the service enters the *Terminated* state.

**Completed** The service releases all of its resources (such as service identifier, priority, etc.).

**Terminated** The function is similar to *Completed* state. It is different from *Completed* state in that *Completed* state means a service finishes successfully, or is broken by the user or service provider, not due to resources dissatisfaction (e.g., multimedia, terminal capabilities, and network), while *Terminated* state means a service abends due to resource dissatisfaction.

Element	Value
S	{Evaluating, Ready, Running, Blocked, Waiting, Completed, Terminated}
I	{I1, I2, I3, I4, I5, I6, I7, I8, I9, I10} I1=(Satisfy user preferences, device capabilities, and network static characteristics, Join ()) I2=(Cannot satisfy user preferences, device capabilities, and network static characteristics, Discard ()) I3=(NULL, Schedule ()) I4=(Service successfully finishes, Finish ()) I5=(User or service provider stops the service, Finish ()) I6=(Cannot transfer for network bandwidth changes, Block ()) I7=(Can transfer after transcoding, Wake up ()) I8=(Cannot transfer after transcoding, Lay aside ()) I9=(Network bandwidth is satisfied for delivery, Wake up ()) I10=(Time over, Discard ())
f	f(Evaluating, I1)=Ready; f(Evaluating, I2)=Terminated; f(Evaluating, I5)=Completed; f(Ready, I3)=Running; f(Ready, I5)= Completed; f(Running, I4)=Completed; f(Running, I5)=Completed; f(Running, I6)=Blocked; f(Blocked, I7)=Ready; f(Blocked, I9)=Ready; f(Blocked, I8)=Waiting; f(Blocked, I5)=Completed; f(Waiting, I9)=Ready; f(Waiting, I5)=Completed; f(Waiting, I10)=Terminated
s <sub>0</sub>	Evaluating
F	{Completed, Terminated}

Table 1: UPMS State Machine Element Values

### 3.2. Service Manipulation

The manipulation methods used for managing service lifecycle include (1) *service creation*, (2) *service preparation*, (3) *service state transformation events*, (4) *service adaptation*, and (5) *service termination*. They are described as follows.

### 3.2.1 Service creation

The service creation method (Create ()) assigns a unique identifier for the new service, sets the state of the new service as *Evaluating*, and initializes the priority as 0.

### 3.2.2 Service preparation

The service preparation method (Prepare ()) selects or generates a multimedia object satisfied with user preferences, terminal capabilities, and network characteristics. It is actually a recommendation process, consists in matching multimedia content with above factors. It will invoke the function of service adaptation if there is multimedia object that satisfied with user preferences, but not satisfied with terminal capabilities or network characteristics. The service preparation method is invoked when the service is in the *Evaluating* state.

### 3.2.3 Service state transformation events

The service state transformation events include:

- Join (): join a service into the queue of ready services.
- Schedule (): chose a service to run. It will call the function of service scheduling, which is described in Section 4.
- Block (): break off a running service, and then call the function of service adaptation.
- Lay aside (): put a service aside waiting for sufficient network bandwidth, and then call the function of bandwidth detection.
- Wake up (): transform a service from the state of *Blocked* or *Waiting* to *Ready*.
- Discard (): drop a service because no multimedia object satisfied with user preferences, device capabilities, and network static characteristics, for passing over bearable waiting time.
- Finish (): when the streaming is completed or file has been downloaded, or user actively stops the service, set the state of the service as *Completed*.

### 3.2.4 Service adaptation

The service adaptation method (Adaptation ()) performs multimedia content adaptation by using two techniques. The first one is content selection, which is concerned with selecting the most appropriate qualities of single media types (e.g. JPEG or GIF images) or selecting the most appropriate media types (e.g. video, image, text, audio, or synthetic model) of the same content. The other technique is multimedia transcoding, consists in transforming the content from one media type to another so that the content can be processed by a particular device or delivered by a specific network condition. For instance, most handheld computers are not capable of handling video data due to their hardware and software constraints. Transforming video into sets of images will enable the devices to access the information contained in the video. The service adaptation function is invoked when the service is in the *Blocked* state. It also may be invoked by the method of Prepare ().

### 3.2.5 Service termination

The service termination method (Destroy ()) releases resources (such as service identifier, priority, etc.) allocated for a service in the condition that a service enters the states of *Completed* or *Terminated*.

## 4. Service Scheduling

The service scheduling module includes scheduling algorithm and priority calculation. It is invoked at two occasions: (1) a service enters the queue of ready services, or (2) a specific running service enters the state of *Completed* or *Blocked*.

### 4.1. Scheduling Algorithm

The scheduling mode is Non-Preemptive based on dynamic priority. Supposing the maximum number of running services the server can provide is  $N$ , the scheduling algorithm is depicted as follows:

Step 1: When the system starts up, the total number of current running services is initialized as 0, and the queue of ready services is initialized as empty;

Step 2: If the queue of ready services is not empty, and the total number of current services is less than  $N$ , choose the service with the highest priority to run, and the total number of current services plus 1;

Step 3: If the queue of ready services is empty or the total number of current services is  $N$ , stop scheduling;

Step 4: If a running service jumps to the state of *Completed* or *Blocked*, and the queue of ready services is not empty, choose the service with the highest priority to run, and the total number of current services plus 1;

If there are several services with the same highest priority, we randomly choose one of them to run.

## 4.2. Priority Calculation

We define that the larger the priority is, the earlier the service will run. The determination of priority conforms to two principles: FCFS (First Come First Serve), and BWFS (Blocked or Waiting First Serve). The priority is determined according to the following rules:

- **Rule 1:** If a service enters the state of *Ready* for the first time, its priority is initialized as 0;
- **Rule 2:** While in the state of *Ready*, once a period ( $T$ ) is passed, the priority plus 1;
- **Rule 3:** If a service enters the state of *Ready* coming from the state of *Blocked*, the priority plus 1;
- **Rule 4:** If a service enters the state of *Waiting* coming from the state of *Blocked*, the priority plus 1;
- **Rule 5:** If a service enters the state of *Ready* coming from the state of *Waiting*, the priority plus 2.

## 5. Service Composition

A ubiquitous personalized multimedia service is built as the composition of components. In our system, service composition is the selection of suited service components in order to deliver the service to a terminal in appropriate form. It consists of two steps. The first step is abstract function selection, e.g. multimedia adaptation. The second one is concrete handler selection, e.g. Video-to-Image transcoder.

Fig. 3 illustrates some composition paths for multimedia services under different conditions.  $S$  denotes the start point of services, while  $T$  stands for the terminal point. The rectangle contains components for recommendation purposes including components A, B and C. The oval contains adaptation components D, E, F, G, and H. The delivery components I and J are represented in the rounded rectangle. In Fig. 3(a), since the appropriate variation already exists, the service goes directly to deliver it. In Fig. 3(b), the service firstly performs video-to-text transcoding, and then downloads the text. In Fig. 3(c), the service first performs video streaming under high bandwidth, then for the decrease of bandwidth, it performs video-to-text transcoding, and then delivers the text. In Fig. 3(d), the service first performs video-to-audio transcoding, then audio streaming, and then for bandwidth decrease dramatically it has to perform audio-to-text transcoding, last sends the text. In Fig. 3(e), component A and B are combined for the recommendation.

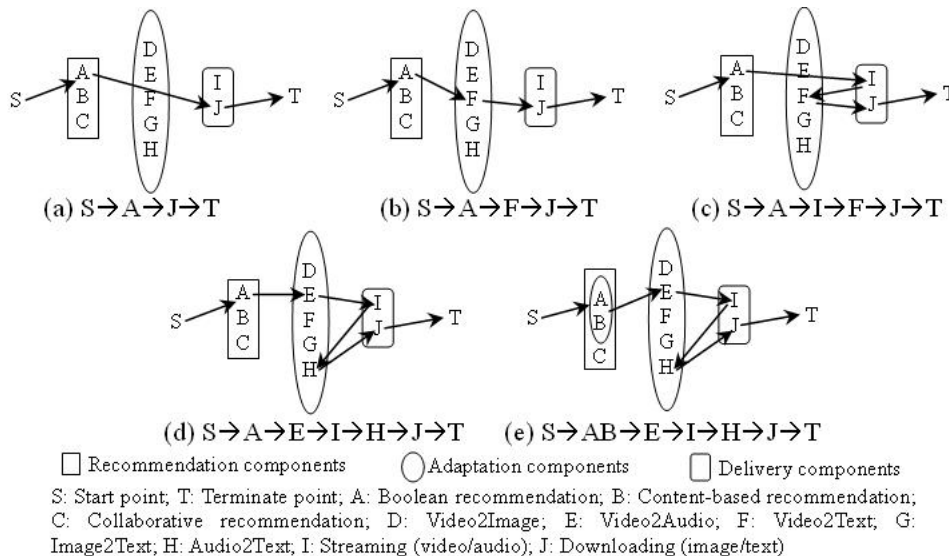
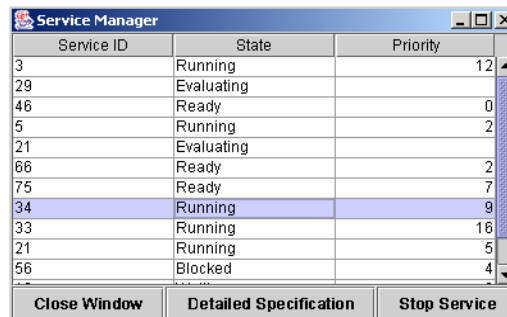


Fig. 3. Service composition examples

## 6. Prototype Implementation

We have applied UPMSM to implement a service manager for ubiquitous personalized multimedia service lifecycle management. The prototype is developed with Java programming language. The experimental environment comprises heterogeneous networks: an IEEE 802.11 wireless LAN and a wired Ethernet network. Three devices are included in the prototype system: a Pocket PC (HP iPAQ H5500), a personal computer, and an IBM server (xSeries 235). The Pocket PC is used to simulate different mobile devices accessing Internet through different communication technologies (e.g., GPRS, 802.11, Bluetooth). The personal computer acts as a general wired client. The service manager is running on the IBM server with an Intel Pentium 4 processor of 2.8 GHz and 512 MB of memory. The multimedia content (TV programs, movies, advertisements) and corresponding metadata (in XML format) are also stored in the IBM server. We have integrated the service manager into our ubiquitous personalized multimedia service platform (UPmP) [10].

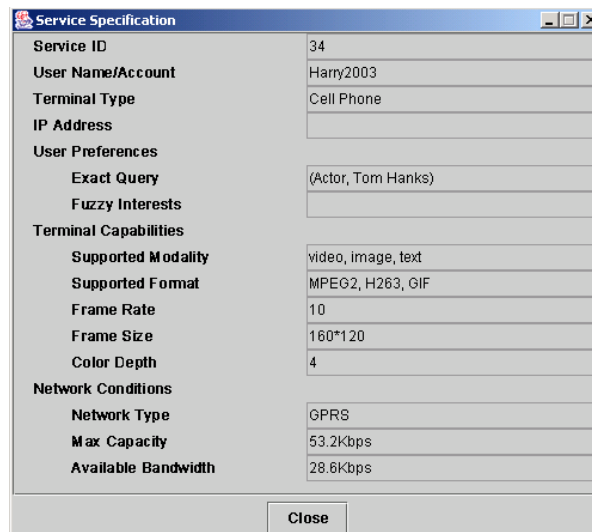
Fig. 4 shows the main interface of service manager. It provides basic information of the services including service ID, state, and priority. When the service provider wants to break off a specific service, he can select it and click the “Stop Service” button to stop it. Once a service is selected and the button of “Detailed Specification” is clicked, details of the service will be showed (see Fig. 5). In this dialogue, service information, such as user name/account, terminal type, IP address, user preferences, terminal capabilities, as well as network conditions, are displayed in detail.



Service ID	State	Priority
3	Running	12
29	Evaluating	
46	Ready	0
5	Running	2
21	Evaluating	
66	Ready	2
75	Ready	7
34	Running	9
33	Running	16
21	Running	5
56	Blocked	4

Buttons: Close Window, Detailed Specification, Stop Service

Fig. 4: Service manager interface



Service ID	34
User Name/Account	Harry2003
Terminal Type	Cell Phone
IP Address	
User Preferences	
Exact Query	(Actor, Tom Hanks)
Fuzzy Interests	
Terminal Capabilities	
Supported Modality	video, image, text
Supported Format	MPEG2, H263, GIF
Frame Rate	10
Frame Size	160*120
Color Depth	4
Network Conditions	
Network Type	GPRS
Max Capacity	53.2Kbps
Available Bandwidth	28.6Kbps

Close

Fig. 5: Service detailed specification

## 7. Conclusion

In this paper, we propose UPMSM, which provides a model for lifecycle management based on FSM for ubiquitous personalized multimedia services. The UPMSM is a powerful model for dynamically adapting arbitrary consumer desired services to personal profiles. A service management module adopting UPMSM emphasizes the need for transparent access to content, where system tasks (e.g. dynamic content adaptation to available network bandwidth) are hidden from the consumer. For future work, we will integrate dynamic

management policies, e.g. through using mobile agents, with our current mechanism.

## 8. Acknowledgments

This work was supported by the National Science Foundation of China, and the Doctorate Foundation of Northwestern Polytechnical University of China.

## 9. References

- [1] Vogel A., Kerherve B., von Bochmann G., Gecsei J. Distributed Multimedia and QOS: A Survey. *IEEE Multimedia*, Summer 1995; 10-19.
- [2] Wichadakul D. D., Gu X. H., Nahrstedt K. A Programming Framework for Quality-Aware Ubiquitous Multimedia Applications, *ACM Multimedia 2002*; 631-640.
- [3] Cui Y., Xu D. Y., Nahrstedt K. SMART: A Scalable Middleware Solution for Ubiquitous Multimedia Service Delivery, *ICME2001*.
- [4] Pappas N., Kazasis F. G., Moumoutzis N., Tsinaraki C., Christodoulakis S. Personalized and Ubiquitous Information Services for TV Programs, *Workshop on Multimedia Contents in Digital Libraries*. Chania, Greece, 2-3 June 2003.
- [5] Steiger O., Ebrahimi T., Sanjuan D. M. MPEG-based Personalized Content Delivery. In *Proceedings of the IEEE International Conference on Image Processing 2003*. Barcelona, Spain, September 2003: 14-17.
- [6] Tseng B. L., Lin C. Y., Smith J. R. Using MPEG-7 and MPEG-21 for Personalizing Video. *IEEE Multimedia 2004*; 11(1): 42 -52.
- [7] Lemlouma T. and Layaida N. Encoding Multimedia Presentations for User Preferences and Limited Environments. *ICME2003*
- [8] Gibson D. Finite State Machines: Making simple work of complex functions. SPLat Controls Pty Ltd. <http://www.microconsultants.com/tips/fsm/fsmartcl.pdf>
- [9] ISO/IEC 21000-7 (N5845), MPEG-21 Multimedia Framework--part 7: Digital Item Adaptation (FCD), 2003.
- [10] Yu Z. W. et Al. UPmP: A Component-Based Configurable Software Platform for Ubiquitous Personalized Multimedia Services. *The 3rd International Conference on Ubiquitous Intelligence and Computing 2006*. , Wuhan, China: Springer-Verlag, LNCS 4159, 2006; 1069-1079, 2006.