A Ubiquitous Personalized Multimedia Service Model Based on FSM

Zhiwen Yu[1], Xingshe Zhou[1], Daqing Zhang[2], Artur Lugmayr[3], Zhiyong Yu[1]

[1] *School of Computer Science, Northwestern Polytechnical University, Xi'an, P.R.China, 710072*
zhiwenyu@nwpu.edu.cn
[2] *Context Aware Systems Department, Institute for Infocomm Research, Singapore*
[3] *Digital Media Institute, Tampere University of Technology, Finland*

Abstract*

The ubiquitous computing environment and users' demand for multimedia personalization precipitate a need for ubiquitous personalized multimedia services (UPMSs). Delivering UPMSs is very complex because of diverse device capabilities, dynamic network characteristics, etc. This paper proposes a lifecycle management model based on FSM (Finite State Machines) for ubiquitous personalized multimedia services (UPMSs). By using this model, the management of UPMSs is very clear and easy, and the services are delivered more efficiently, effectively, and robustly.

1. Introduction

Multimedia information is widely used in pervasive computing environment in many application fields, such as multimedia digital libraries, the home entertainment, live camera remote surveillance, and etc. A major trend and requirement in today's multimedia service is personalization, which means 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 not relevant. The environment of ubiquitous computing and users' demand for multimedia personalization precipitate a need for ubiquitous personalized multimedia services (UPMSs). Delivering UPMSs is very complex for diverse device capabilities, dynamic network characteristics, etc.

Most of existing related research is mainly focused on service framework or architecture, multimedia description model, and media transcoding or adaptation mechanisms. However, they rarely put attention on service running status management, which is critical for a complex application like multimedia personalization in ubiquitous environments. Thus, a big challenge for ubiquitous multimedia service provider is to build a model that can model the running process and manage the lifecycle of UPMSs. In this paper, we propose a ubiquitous personalized multimedia service model (UPMSM) based on FSM (Finite State Machines). 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 or electronic control systems [1]. In our solution, a UPMS is modeled as a deterministic FSM.

* This work was supported by the Doctorate Foundation of Northwestern Polytechnical University of China.

The strengths of the UPMSM lie in its unified model for personalized multimedia services in ubiquitous environments, i.e. independent of 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.

2. UPMSM Model

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

Definition 1 (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 2, 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 2 (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* takes place. In each element, *R* can be NULL, while *E* cannot be NULL.

Figure 1 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.

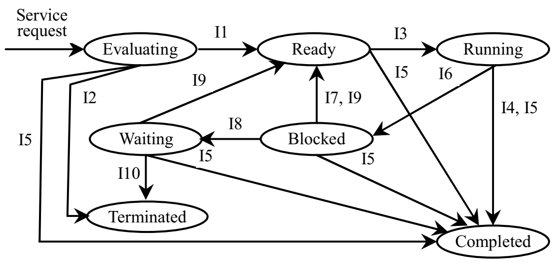

Figure 1. UPMS State Machine Model

The element values of UPMS State Machine are presented in Table 1.

Table 1: UPMS State Machine Element Values

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_0	Evaluating
F	{Completed, Terminated}

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 trancoding 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.

3. 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.

We have applied UPMSM to implement a service manager for ubiquitous personalized multimedia service lifecycle management. The preliminary experimental results proved the effectiveness of the proposed scheme.

References

[1] David Gibson, “Finite State Machines: Making simple work of complex functions”, SPLat Controls Pty Ltd., *http://www.microconsultants.com/tips/fsm/fsmartcl.pdf*