특집

Standardization of MPEG Multimedia Middleware

Hendry and Munchurl Kim (Laboratory for Multimedia Computing, Communications, and Broadcasting School of Engineering, Information and Communications University)

Abstract

Recently, MPEG (Moving Picture Experts Group) has started a new standardization activity called MPEG Multimedia Middleware (M3W). This activity covers the standardization for a middleware with a set of APIs. The standard middleware, called M3W, will support independent multimedia applications for hardware and software such as operating systems so that the portability and interoperability of multimedia applications are greatly improved. In this article, we will introduce the recent works on M3W standardization including the rationale that motivates the activity with the advantages of having the standardized middleware for multimedia application, component technologies being considered for the M3W, and the current status of the standardization.

I. Introduction

Early multimedia applications tended to be realized in hardware. However, the complexity of the multimedia applications has been increased and also the hardware has become more and more complex. In the mean while, the software technologies have been advanced so that many complex multimedia applications have become possible in full software implementations as the hardware processing power has been increased. Therefore, multimedia applications have become more dependent of software but less dependent of hardware.

However, the dependency on software has caused a problem by reducing the reusability of software for a variety of multimedia applications.

Thanks to this, there have been requirements for a generic middleware which can provide a bridge between hardware and software platforms. From an application

developer's standpoint, it's easy to develop their own applications with *a priori* knowledge about software platform and hardware. This phenomenon makes the hardware and software become less dependent each other. So there are two obvious players that can be distinguished in term of their roles: semiconductor manufacturers and consumer electronics vendors. Semiconductor manufacturers can only focus on providing the hardware as powerful and generic as possible to serve bigger markets while the consumer electronics vendors focus on developing their prospective multimedia applications in full software without much concern about the hardware. These different focuses of the two players cause a gap which needs to be filled by a new player so called middleware. Middleware bridges the gap between the hardware and the applications[2]. So, different multimedia applications can easily be realized on top of a standardized multimedia middleware.

Figure 1 shows an illustration of the shift

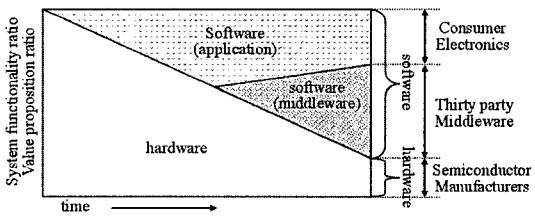


Fig. 1. Shifting in hardware and software dependency[2]

of the multimedia application dependency on the hardware onto software. As the time goes on, the multimedia applications become less dependent of hardware but more dependent of software. However, at certain point, there is a need to have a middleware so that the differentiated applications can run on any generic platform offered by the hardware through the mediation service provided by the middleware.

Currently, various kinds of multimedia applications have been and/or are being developed for a variety of multimedia environments from the end-user applications such as games, DVD player, mp3 player, and digital television to the enterprise applications such as video teleconference, e-learning, etc. But, the advance of information technology has offered multimedia consumption markets with various platforms for multimedia environments. To this end, multimedia applications are often developed on specific platforms so that they can be developed in shorter time and easier in maintenance. But end-users usually desire platform independent applications so that the multimedia applications can run on any platforms/ devices.

To cope with this situation, again, the middleware is needed to bridge this gap. With the middleware that is dedicated for multimedia, it will be possible to have portable and interoperable multimedia applications running on different platforms

with the minimum modification.

II. M3W Objectives and requirements

M3W aims at defining a set of APIs supporting the following functionalities [2]:

- Allowing multimedia applications to execute functions provided by the middleware in a standard way without the need for an in-depth knowledge of the middleware;
- Allowing multimedia applications to trigger the update / upgrade or extension of the API, e.g. because they are not present or outdated.

To satisfy those objectives, the following technologies have been identified [2]:

- Functional APIs to allow the applications to access the multimedia functions;
- APIs to manage the execution behaviour (e.g. resource (memory, power, processor, network usage) of the multimedia functions;
- APIs to manage the isolation (e.g. memory protection) of the multimedia functions;
- APIs to manage the 'life-cycle' of the components of the M3W (e.g. identification, download, installation, activation, de-activation, un-installation).

While the first bullet of the aforementioned essential technologies is the technology that provides direct access to the multimedia functionality provided by the M3W services, the rest three technologies are the technologies for managing the middleware including its services. To be more precise, the set of those needed APIs can be formulated in requirement APIs in the categories as follow [2][3]:

- Multimedia APIs. They are intended to give the requesting application access to multimedia functionalities provided by the middleware components. This APIs can be in the categories such as media processing Services (coding, decoding, transcoding, etc), digital right management (DRM) Service, access to metadata Service (such as editing, searching, etc), and many more categories;
- Management APIs. They are intended to allow the handling and management of the middleware components, and the procedures are required to deal with essential processes like Service localization, decision/validation, downloading, instantiation and related operations;
- In-Operation APIs. They are intended to take extra functionalities into account that are essential for the sound execution of the applications that request the Services of the middleware which, on their turn, requests Services from the underlying platform. The middleware (or its Services) mediates the request by the application with

what is possible on the platform;

- Other APIs. It is intended to accommodate ideas/concepts brought by the technologies that are not stated in detail yet, for example the composition of the component/Service, communication protocol, etc.

By defining those functionalities and having the needed technologies, it is expected that the middleware can benefit the stakeholders who are involve a in the process of multimedia application creation by allowing [2]:

- Application developers to quickly develop and easily maintain multimedia applications;
- Application and Service providers to limit the cost of application provisioning;
- Manufacturers to provide compelling and interoperable application development environment.

III. Service-based Middleware Architecture

To fulfill the aforementioned objectives and requirements, M3W is designed to be positioned in the layer as shown in Figure 2. M3W lies between the computing platform and the applications/(other)middleware to bridge the access gap between them. Please note that we consider other middleware as the user of the M3W, just like the applications.

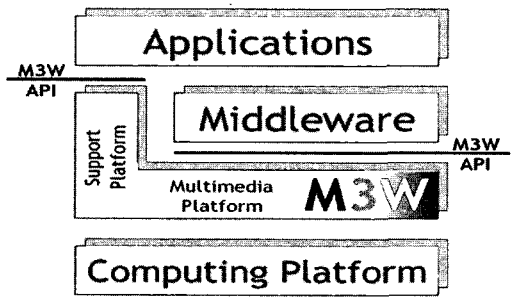


Fig. 2. M3W position in the layered computing environment [7][8].

The communication between M3W and the upper part (applications) is differnet from that between M3W and lower part. M3W applications interact only through the defined APIs which will be standardized by MPEG. In the other part, M3W and the platform interact in a way that varies from one platform to another platform. It might depend on the implementation of the M3W. Note that although the applications do not have a direct communication to the platform in Figure 2, the M3W does not prohibit the direct access from applications to the computing platform as what is happening now when there is no M3W.

M3W is designed as a service-based multimedia middleware. All the functionalities offered by M3W are encapsulated in the form of Services; hence, the Services themselves utilize a group of APIs as their interfaces to the entities that access them. Figure 2 shows that in the M3W layer there are the support platform and the multime-

dia platform. The functionality of the support platform is to provide the management functionality to the M3W. It manages the execution of the M3W. On the other hand, the multimedia platform contains the Services that are dedicated for multimedia. For example, it may contain a Service for video or audio codec, a Service for encryption to be used in an DRM system, and a file system Service, etc.

For its functionality, the support platform may contain various support Services such as Service Manager, RunTimeEnvironment, QoS manager, Download Manager, etc. Those support Services offer APIs for the non-functional APIs.

Each functional Service, on the other hand, is designed to be flexible, modular, and efficient in its implementation. It can be achieved by letting an implementation of a Service with the interfaces exposed to other Services.

Figure 3 illustrates the implementation of a Service that efficiently uses the functionality from other Services. In Figure 3, a video decoder Service, for example, may consist of several independent functions such as a variable length coding (VLC) function, discrete cosine transform (DCT) function, quantization function, motion estimation/compensation function, etc. The Video Decoder Service can be implemented as modular as possible by using other implementations that offers the required functionalities. Hence, in its implementation, the implementer needs to code the execution flow of those sub Services.

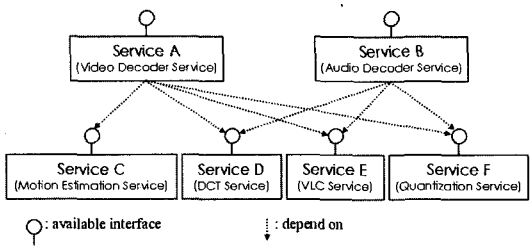


Fig. 3. Service composition examples

In other cases, suppose that, in the same middleware, an audio decoder Service also exists at the same time. With the hierarchical concept of the Service implementation, it may not be necessary to have two DCT, VLC, and Quantization Services. The same Services that are used for the Video Decoder Service (Service A) can be used to serve the Audio Decoder Service (Service B) as well so that the middleware can be very efficient in the sense that there is no duplication in the offered function.

The binding between a Service and its dependencies is mediated by the Service Manager. During its initialization, a Service can request to the Service Manager to provide its dependent Service(s). The Service Manager gives the best possible Service by using the knowledge from the available metadata of Services.

The metadata itself, in M3W, serves as a

"Model". It provides a representation or description for service and its packager (component). It is a deployment unit which is delivered to the Service Manager to be stored.

The metadata is defined by using XML Schema. We design the metadata to carry the information such as:

- –Component identification information such as globally unique identification (GUID), title, creator, abstract/synopsis of the component.
- –The content of the component.
  - ●List of offered interfaces for multimedia APIs.
  - ●List of Services that implement those offered interfaces.
- –List of the target platforms (if the implementation can be deployed into several platforms).
- –The license information that governs the usage of the component/Services.
- –Required usage environment for the best usage of the component.
- –Signature information to guarantee the integrity of the carried information.

When a service is deployed into the terminal (with a certain platform), the metadata of the service is registered to the Service Manager so that when the Service is requested later, the Service Manager might know how to fulfill the request by exploiting the metadata. Figure 4 shows an illustration of what happens when a Service is requested by an application.

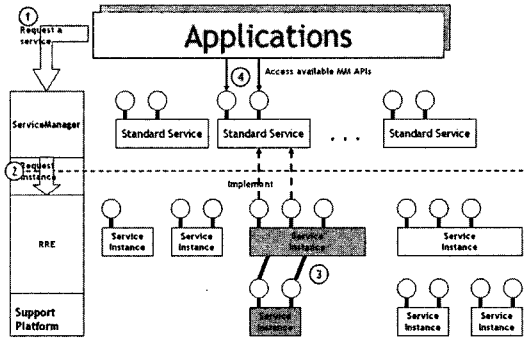


Fig. 4. Process when a service is requested

In Figure 4, as we can see, there is a dashed-line separating the upper part and lower part. The upper part is the logical part which is in high level concept while the lower part is the implementation which is transparent to the application. When an application requests a Service, it contacts the Service Manager by giving the identification of the standard Service. Upon receiving this request, the Service Manager starts examining its knowledge (collection of metadata in its memory) and seeks whether there is such Service instance configuration for that request. If there exists one, then, it requests the RunTimeEnvironment (RRE) to instantiate all the necessary Service(s). Once the instantiation of all the necessary Services is done, the Service Manager binds them and gives the pointer to access the Service to the requesting application.

So finally the application can start using the instantiated Service without having to know the detail of the Service.

The Service request procedure as shown in Figure 4 is a typical procedure. Other more advanced procedure might be used. For example, rather than requesting the Service one by one, an application may notify the Service Manager *a priori* about all the necessary Services. So the Service Manager can check whether the middleware can support the required Services and let the application know in advance. This might be useful in the case of streaming application for example. The application might check whether a terminal can process the streamed media before the actual content streaming happens. In the case that the terminal cannot afford it (Service Manager says not enough service available) then the network bandwidth will not be wasted.

IV. Current status of M3W development and standard

The time table of the M3W development is as follow [4]:

- March 2004: Requirements Document Version 1.0
- July 2004: Requirements Document Version 1.0
- October 2004: Requirements Document Version 2.0
- January 2005: Call for Proposals (CfPs) on M3W
- April 2005:
  - -Evaluation of CfPs' responses
  - -M3W Working Draft Version 1.0
  - -Extended CfPs for functional APIs
- July 2005:
  - -M3W Working Draft Version 2.0
  - -Extended CfPs for functional APIs

In the 73rd MPEG meeting in Poznan, Poland, the M3W standardization has been decided to be a new independent project (before it was under the MPEG system group). The M3W standard was assigned a new standard number ISO/IEC 23004 which comprises 7 parts. The M3W includes the following standard parts [6]:

- Part 1: Architecture
- Part 2: Multimedia API
- Part 3: Component Model
- Part 4: Resource and Quality Management
- Part 5: Component Download
- Part 6: Fault Management
- Part 7: System Integrity Management

For each part, reference software and conformance will be included in the future.

All the abovementioned parts contribute to build the complete M3W. Part 1 defines how the middleware shall be organized. It

describes the position of M3W in the layered software environment, and the interaction amongst those layers with the M3W. The interfaces for those interactions are defined in Part 2. Part 3 defines how the encapsulation of the multimedia APIs is defined in Part 2 into the services and components. It also defines the execution framework for the implementation of the Services and Components. Part 4 defines the mechanism of how M3W ensures the quality of its offered Services so that the performance can be adjusted according to the situation of the terminal. When a multimedia application is requested, it maybe necessary to get downloaded all the required Sevices and Components into the middleware if they are available in it. Part 5 defines how this download mechanism can be realized. When a Service misbehaves, there should be a mechanism to isolate it, or, if possible, to replace it. This mechanism is realized via the fault management, defined in Part 6. Finally Part 7 defines the integrity of the whole system.

The current organizations that participate in the development of the M3W technologies includes Philips Research, Information and Communication University – Korea, UHAPI consortium (includes Samsung, HP, nVIDIA, Philips, etc), ETRI, University of Florence – Italy. It is expected that more parties will join this activity as it is in the development phase.

V. Conclusion

M3W is designed to be a middleware that is dedicated for providing interoperable multimedia applications so that the multimedia applications can run on top of any platforms equipped with M3W at minimum modification. In future, it is expected that M3W will be widely deployed and/or embedded in various devices such as PC's, portable music players, mobile phones, etc so that it can support multimedia interoperability between homogeneous devices (within same kinds of devices, i.e., one mobile phone to another mobile phone, but maybe different manufacturer), as well as between heterogeneous devices (across different kind of devices, i.e., a PC to a mobile phone). Hence, ubiquitous consumption of multimedia applications can be possible.

The deployment of M3W will have significant impact to the industry, especially the device manufacturers and service/content providers since they can aim at bigger market for their products/services. An DRM-protected music services over mobile phones is a good example. Currently, the protected music from a content provider can be consumed only in a limited type (or

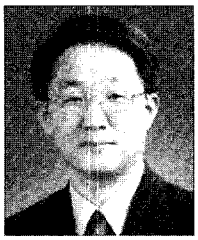
even only one type) of mobile phones. So there is a very tight relationship between the content provider and the mobile phone manufacturer. However, with the M3W embedded in mobile phones, the protected music can be processed and consumed in any mobile phone by utilizing its interoperability-guaranteed middleware environment. The content provider expands its market from only limited types of mobile phones that can play its music to all types of mobile phones. On the other hand, the mobile phone manufacturers also take benefits since their products can playback various kinds of protected music contents offered by different content providers.

Finally, considering the impact it brings, this M3W standardization is important. It is recommended that related organizations in the multimedia industry support and participate in this standardization activity.

REFERENCES

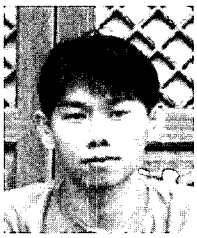
1. ISO/IEC JTCI/SC29/WG11 "MPEG-21 Vision, Technology, and Strategy", MPEG67/N6269, Hawaii, USA, December 2003.
2. ISO/IEC JTCI/SC29/WG11, "MPEG Multimedia Middleware: Context and Objective," MPEG68/N6335, Munchen, Germany, March 2004.
3. ISO/IEC JTCI/SC29/WG11, "MPEG Multimedia Middleware Requirements v.2.0," MPEG70/N6835, Palma de Mallorca, Spain, October 2004.
4. ISO/IEC JTCI/SC29/WG11, "Call for Proposals on Multimedia Middleware (M3W)," MPEG71/N6981, Hong Kong, China, January 2005.
5. ISO/IEC JTCI/SC29/WG11, "WD1.0 of MPEG Multimedia Middleware," MPEG72/N7254, Busan, Korea, April 2005.
6. ISO/IEC TC JTCI/SC29/WG11, "WD2.0 of ISO/IEC 23004-1~7," MPEG73/N7494~N7500. Poznan, Poland, July 2005.
7. ISO/IEC TC JTCI/SC29/WG11, "White Paper on Multimedia Middleware," MPEG73/N7510, Poznan, Poland, July 2005.

저자소개



김 문 철

1989년 2월 경북대학교 전자공학과 학사
1992년 12월 University of Florida, Electrical and Computer Engineering 석사
1996년 8월 University of Florida, Electrical and Computer Engineering 박사
1997년 1월~2001년 2월 ETRI 방송미디어연구부 팀장(선임연구원)
2000년 MPEG 2001년 ~ 현재 MPEG 포럼 운영위원 및 SC29-Korea 전문위원, 한국대표단 단장
2001년 2월~현재 한국정보통신대학교(ICU) 부교수
주관심 분야 비디오 압축, 비디오 트랜스코딩, 멀티미디어 DRM, 통방융합 방송 미디어 서비스, 지능형/개인형 멀티미디어 방송, MPEG-4/7/21, MPEG-A, MPEG-E



Hendry

2001년 7월 University of Indonesia, Computer Science 학사
2005년 2월 한국정보통신대학교(ICU) 공학석사
2005년 2월~현재 한국정보통신대학교(ICU) 박사과정
2005년 7월~현재 MPEG-E MPEG Multimedia Middleware (M3W) Coeditor
주관심 분야 멀티미디어 DRM, 대화형 멀티미디어 시스템, MPEG-21, MPEG-A, MPEG-E