Signal Processing: *Image Communication* 29 (2014) 150–166

Contents lists available at ScienceDirect

Signal Processing: *Image Communication*

journal homepage: www.elsevier.com/locate/image

MPEG-M: A digital media ecosystem for interoperable applications

Panos Kudumakis [a,*], Mark Sandler [a], Angelos-Christos G. Anadiotis [b], Iakovos S. Venieris [b], Angelo Difino [c], Xin Wang [d], Giuseppe Tropea [e], Michael Grafl [f], Víctor Rodríguez-Doncel [g], Silvia Llorente [h], Jaime Delgado [h]

[a] *School of Electronic Engineering and Computer Science, Queen Mary University of London, London, UK*
[b] *School of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece*
[c] *CEDEO.net, Torino, Italy*
[d] *Huawei, Santa Clara, USA*
[e] *Consorzio Nazionale Interuniversitario per le Telecomunicazioni, Parma, Italy*
[f] *Institute of Information Technology, Alpen-Adria-Universität Klagenfurt, Klagenfurt, Austria*
[g] *Universidad Politecnica de Madrid, Madrid, Spain*
[h] *Universitat Politècnica de Catalunya, Barcelona, Spain*

ARTICLE INFO

Article history:
Received 5 April 2013
Accepted 23 October 2013
Available online 30 October 2013

Keywords:
MPEG
IPTV
Middleware
Interoperability
Digital media
Multimedia architectures

ABSTRACT

MPEG-M is a suite of ISO/IEC standards (ISO/IEC 23006) that has been developed under the auspices of Moving Picture Experts Group (MPEG). MPEG-M, also known as Multimedia Service Platform Technologies (MSPT), facilitates a collection of multimedia middleware APIs and elementary services as well as service aggregation so that service providers can offer users a plethora of innovative services by extending current IPTV technology toward the seamless integration of personal content creation and distribution, e-commerce, social networks and Internet distribution of digital media.

1. Introduction

With the deployment of broadband networks enabling new ways to deliver and exchange multimedia services and the improvement of hardware performance allowing many service aspects to be implemented as web-service software, businesses related to media services are facing significant changes. These changes are opening new business opportunities for multimedia services, such as those generated by the recent introduction of IPTV services for which several standards have been or are being developed. Examples of already developed standards are: ITU-T Q.13/16, Open IPTV Forum, Alliance for Telecommunications Industry Solutions IPTV Interoperability Forum, Digital Video Broadcasting IPTV, Hybrid Broadcast Broadband TV and YouView.

However, most of the current IPTV efforts stem from rather conventional value chain structures thus standing in stark contrast with the buoyant web environment where new initiatives – sometimes assembling millions of users in a

* Corresponding author. Tel.: +44 20 7882 6152.
E-mail address: panos.kudumakis@eecs.qmul.ac.uk (P. Kudumakis).

0923-5965/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.image.2013.10.006

fortnight – pop-up almost daily with exciting new features, such as Apple's and Google's Application Programming Interfaces (APIs) enabling third parties to develop and provide applications and services [1,2].

At the same time we are witnessing cases where the closed delivery and content bundles offered by some operators are being either abandoned (e.g. mobile phone brands linked to a particular content service) or complemented with the possibility offered to users to freely access services (e.g., broadband, mobile and IPTV) of their choice. The latter becomes more eminent by the appearance of new operators offering service components (e.g., cloud services) and the need for these to be interoperable.

MPEG has been the provider of some enabling technologies and has developed a large portfolio of standards that can be assembled to provide multimedia services [3]. Continuing its approach of providing standards for the next generation of products, services and applications, MPEG has developed MPEG-M, a standard for advanced IPTV services. MPEG-M is based on a flexible architecture capable of accommodating and extending in an interoperable fashion many features that are being deployed on the web for delivering and consuming multimedia content (e.g., Hulu, Netflix or Apple TV), next to those enabled by the recent standard MPEG technologies (e.g., High Efficiency Video Coding and Dynamic Adaptive Streaming over HTTP [4,5]).

Thanks to the MPEG-M suite of standards, aimed at facilitating the creation and provisioning of vastly enhanced IPTV services, it is envisaged that a thriving digital media economy can be established, where *developers* can offer MPEG-M service components to the professional market because a market will be enabled by the standard MPEG-M component service API; *manufacturers* can offer MPEG-M devices to the global consumer market because of the global reach of MPEG-M services; *service providers* can set up and launch new attractive MPEG-M services because of the ease to design and implement innovative MPEG-M value chains; and *users* can seamlessly create, offer, search, access, pay/cash and consume MPEG-M services.

The MPEG-M suite of standards extends the devices capabilities with advanced features such as content generation, processing, and distribution by a large number of users; easy creation of new services by combining service components of their choice; global, seamless and transparent use of services regardless of geo-location, service provider, network provider, device manufacturer and provider of payment and cashing services; diversity of user experience through easy download and installation of applications produced by a global community of developers since all applications share the same middleware APIs; and innovative business models because of the ease to design and implement media-handling value chains whose devices interoperate because they are all based on the same set of technologies, especially MPEG technologies.

A brief overview of the MPEG-M suite of standards can be found in [6]. In contrast, this paper is focused on the detailed description of the MPEG-M digital media services ecosystem and its components providing all the necessary technical information (including access to the reference software) needed by developers who would like to build MPEG-M compliant applications and services; and, why not attract millions of users in a fortnight, too!

The rest of the paper is structured as follows: In Section 2 the scope and objectives of the MPEG-M digital media services ecosystem are explained. In Section 3 each of the individual MPEG-M standards described in detail including the functionalities offered by each of them. In Section 4 a number of MPEG-M related developments offering various digital media applications and services are presented. In Section 5 the critical decisions and choices that had to be made by the MPEG-M ad hoc group during the MPEG-M's life cycle development followed by related future developments are discussed. Finally, in Section 6 the conclusions are presented by highlighting the major MPEG-M achievements.

2. Scope and objectives

The scope of the MPEG-M is to support the service providers' drive to deploy innovative multimedia services by identifying a set of Elementary Services (ESs) and defining the corresponding set of protocols and APIs to enable any user in an MPEG-M value chain to access those services in an interoperable fashion. Note that an MPEG-M value chain is a collection of users, including creators, end users and service providers that conform to the MPEG-M standard.

Assuming that in an MPEG-M value chain there is a Service Provider (SP) for each ES, a User may ask the Post Content SP to get a sequence of songs satisfying certain Content and User Descriptions (metadata). The “mood” of a group of friends could be a type of User Description.

With reference to Fig. 1, the End User would contact the Post Content SP who would get appropriate information from both the Describe Content SP and the Describe User SP in order to prepare the sequence of songs according to the friends “mood” by using, for example, a semantic music playlist generator [7]. The End User would then get the necessary licenses from the Manage License SP. The sequence of songs would then be handed over to the Package Content SP, possibly in the form of an “MPEG-21 Digital Item”, the latter being a container for Resources, Metadata, Rights and their interrelationships [8]. The Package Content SP will get the Resources from the Store Content SP and hand over the Packaged Content to the Deliver Content SP who will stream the Packaged Content to the End User.

In many real-world MPEG-M value chains, service providers would not be able to exploit the potential of the standard if they were confined to only offer ESs. Therefore service providers will typically offer bundles of ESs, known as Aggregated Services (ASs). In general, as shown in Fig. 2, there will be a plurality of service providers offering the same or partially overlapping ASs, for example, a SP offering User Description Services, may offer Content Description Services as well.

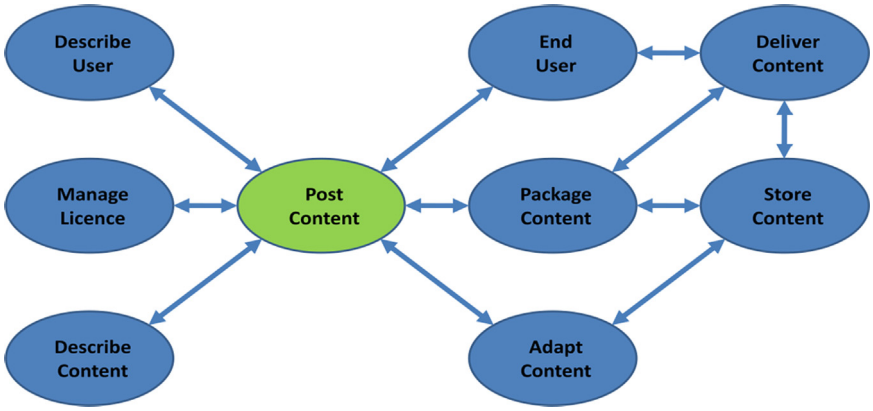

Fig. 1. A possible chain of Services centered around Post Content Service Provider.

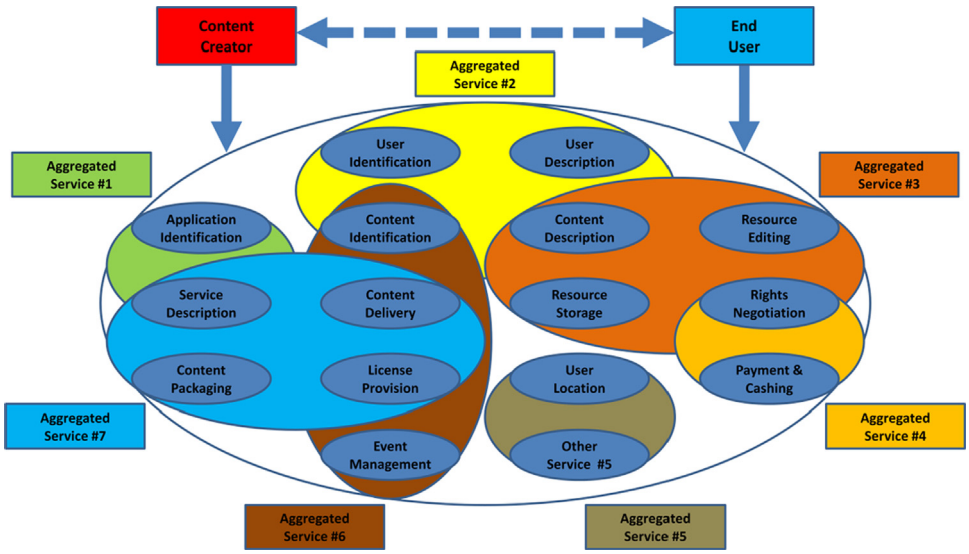

Fig. 2. MPEG-M standard-enabled digital media services eco-system underpinning and supporting the activities of content creators and consumers.

Starting from MPEG-M elementary services, the aggregation of services can put together a certain amount of services generating a complex MPEG-M value network, having different topologies and associating services in several ways. For example, the Payment and Cashing and Rights Negotiation ESs are aggregated to create AS#4, while Content Delivery and License Provision ESs are both shared between AS#6 and AS#7.

3. Technology

MPEG-M (ISO/IEC 23006) is a suite of standards that has been developed under the auspices of Moving Picture Experts Group (MPEG).

ISO/IEC 23006 is referred as MPEG Extensible Middleware (MXM) *in its first edition*, and it specifies an architecture (Part 1), an API (Part 2), a reference software (Part 3) and a set of protocols which MXM Devices had to adhere (Part 4).

MPEG-M (ISO/IEC 23006) is referred as Multimedia Service Platform Technologies (MSPT) *in its second edition*, and it conserves the architecture and design philosophy of the first edition, but stressing its Service Oriented Architecture character. Furthermore, it specifies how to combine elementary services into aggregated services (Part 5).

More specifically, the second edition of MPEG-M is subdivided into five parts:

- Part 1 Architecture: specifies the architecture that is part of an MPEG-M implementation;
- Part 2 MPEG Extensible Middleware (MXM) Application Programming Interface (API): specifies the middleware APIs;
- Part 3 Conformance and Reference Software: specifies conformance tests and the software implementation of the standard;

- Part 4 Elementary Services: specifies elementary service protocols between MPEG-M applications;
- Part 5 Service Aggregation: specifies mechanisms enabling the combination of elementary services and other services to build aggregated services.

These five parts are described next.

3.1. Part 1 – architecture (23006–1)

The first part of the standard describes the MPEG-M architecture, its elements and APIs that enable MPEG-M compliant devices to be interoperable albeit made by different manufacturers.

A general architecture of an MPEG-M device is given in Fig. 3, where MPEG-M applications running on an MPEG-M device could call, via an application-middleware API, the Technology Engines (TEs) in the middleware to access local functionality modules, and the Protocol Engines (PEs) to communicate with applications running on other devices by executing elementary or aggregated service protocols among them. The role of the orchestrator engine is to set up a more complex chain of TEs and PEs.

Typical technology engines include MPEG technologies such as Audio, Video, 3D Graphics, Sensory Data, File Format, Streaming, Metadata, Search, Rendering, Adaptation, Rights Management and Media Value Chain Ontologies [3].

Typical protocol engines include those implementing the elementary services, as described earlier in the music "mood" example, such as Describe User, Describe Content, Manage License, Package Content and Deliver Content.

The elements of the MPEG-M architecture are MPEG-M engines, MPEG-M engine APIs, MPEG-M orchestrator engine, MPEG-M orchestrator engine API, MPEG-M device, and MPEG-M application. *MPEG-M engines* are collections of specific technologies that can be meaningfully bundled together; the *MPEG-M engine APIs* can be used to access functionalities of MPEG-M engines; an *MPEG-M orchestrator engine* is a special MPEG-M engine capable of creating chains of MPEG-M engines to execute a high-level application call such as "Photo Slide Show"; the *MPEG-M orchestrator engine API* can be used to access the MPEG-M orchestrator engine; an *MPEG-M device* is a device equipped with MPEG-M engines; and an *MPEG-M application* is an application that runs on an MPEG-M device and makes calls to the MPEG-M engine APIs and the MPEG-M orchestrator engine API.

In general an MPEG-M device can have several MPEG-M applications running on it such as an audiovisual player or a content creator combining audio-visual resources with metadata and rights information. Some applications may be resident (i.e., loaded by the MPEG-M manufacturer) while some may be temporary (i.e., downloaded for a specific purpose).

When an MPEG-M application is executed, there may be "low-level" calls directly to some MPEG-M engines through their APIs and "high-level" calls like, say, "Photo Slide Show", which will be handled by the orchestrator engine. The MPEG-M orchestrator is capable of setting up a chain of MPEG-M engines for handling complex operations, orchestrating the intervention and send/receive data to/from the particular chain of engines that a given high-level call will trigger, thus relieving MPEG-M applications from the logic of handling them.

3.2. Part 2 – application programming interface (23006–2)

The second part of the standard specifies a set of Application Programming Interfaces (APIs). These APIs are the gateway to the MPEG-M middleware – providing access to its technology engines as specified in Part 1 – for any application running on an MPEG-M device.

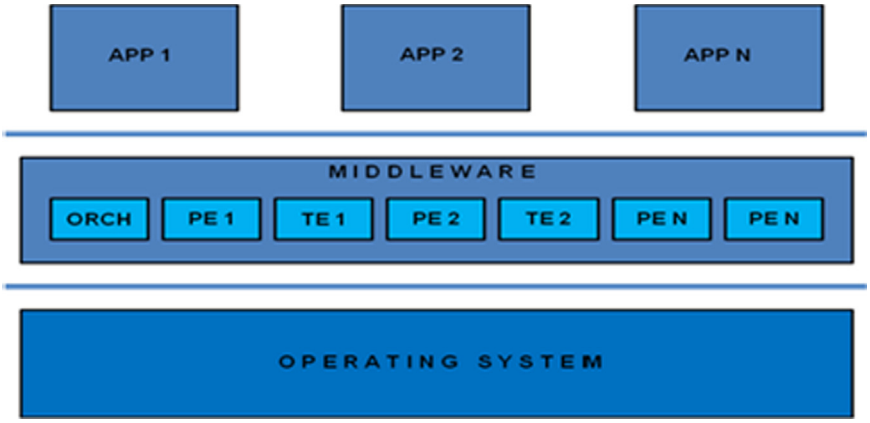

Fig. 3. MPEG-M device architecture; the middleware is populated by Technology Engines (TEs), Protocol Engines (PEs) and one or more Orchestration Engines (ORCH).

These APIs are designed in such a way so that an individual MPEG-M engine, which provides access to a single MPEG technology (e.g., video coding), can be orchestrated with other engines. In this way, chains of MPEG-M engines to execute "high-level" application calls can be created. That is, calls to a group of MPEG technologies (e.g., creating a "Photo Slide Show") become feasible.

Conceptually, these APIs are divided in four categories, namely creation APIs, editing APIs, access APIs and engine-specific APIs. *Creation APIs* are used to create data structures, files and elementary streams conforming to the respective standards; *Editing APIs* are used to modify an existing data structure, file, elementary stream in order to obtain a derived object still conforming to the respective standard; *Access APIs* are used to parse data structures, files, decode elementary streams in order to retrieve the information contained within; and *Engine-specific APIs* are those that do not fall into the above categories, such as APIs for license authorization and content rendering.

Furthermore, Part 2 of the standard contains the description and the API specification of MPEG-M engines, which are classified into three types: (a) Protocol Engines (PEs), (b) Technology Engines (TEs), and (c) Orchestrator Engines.

3.2.1. Protocol engines

The protocol engines are instantiating the communication protocol of the elementary services and, therefore, there is a one-to-one relationship between them. Their APIs have been designed in a unified way by providing interfaces for creating and parsing protocol requests and responses, as they are specified by the corresponding elementary services in MPEG-M Part 4, as well as for performing the requests and receiving the responses.

A fundamental abstract protocol engine is the Base PE, corresponding to the base protocol of MPEG-M Part 4, which contains the methods and interfaces for creating and parsing the MPEG-M base schema. The Base PE cannot exist alone; all the other protocol engines must extend the Base PE, in order to ensure the consistency between MPEG-M Parts 2 and 4.

The schema handler of a protocol engine is used to manipulate the schemata dictated by the corresponding elementary services, where the technology handler provides the means to implement the call to the orchestrator (local or remote) which manages the technology engines that perform the actual elementary service operation. A typical protocol engine of the form $<\text{Action}><\text{Entity}>$Engine (e.g., CreateContentEngine) has a specification similar to the one shown in Listing 1. This specification is neither exhaustive nor aligned to all the engines; it is just given as a typical example.

Listing 1: Typical protocol engine (PE) form.

```
<Action><Entity>Engine:
  <Action><Entity>SchemaHandler:
    <Action><Entity>Request extends ProtocolRequest
    <Action><Entity>Response extends ProtocolResponse:
      <Action><Entity>Success extends ProtocolSuccess
      <Action><Entity>Failure extends ProtocolFailure
  <Action><Entity>TechnologyHandler:
    <Action><Entity>Protocol
```

3.2.2. Technology engines

The technology engines are responsible for carrying out the actual operation of an elementary service (even though a technology engine can be used by other components of the middleware or application layer, their primary scope is the support of elementary services operation). They are also organized in terms of schema and technology handlers. The schema handler is mainly used for managing the schemata dictated by the standards used for implementing the corresponding technology. For example, the Security TE schema handler provides interfaces which allow for creating and parsing objects based on XML security schemata provided by W3C (e.g., digital signatures) and OASIS (e.g., Security Assertion Markup Language). On the other hand, the Digital Item TE is based on MPEG-21 technologies, such as the MPEG-21 DIDL schema and, hence, its schema handler provides the interfaces to access this schema transparently, thereby stimulating the interoperability features of MPEG-M implementations.

The technology handler of the technology engines is responsible for exposing the API that allows handling of the underlying technology. With respect to the aforementioned Security TE example, these APIs enable developers to use encryption and signature capabilities, which could be provided by different implementations (e.g., smart card and/or other software security solutions).

Table 1 provides a list with the technology engines supported by the standard. These engines can be deployed by several elementary services and can be accessed either directly or through an orchestrator engine.

3.2.3. Orchestrator engines

Orchestrator engines are managing the execution flow of technology engines in the context of an elementary service, an aggregated service or an application. The standard contains one protocol engine per elementary service. In contrast, one or more orchestrator engines can be specified based on service or application requirements. Given that orchestration engines

are specific to applications, the focus of their APIs is in providing a standardized way to their instantiation, rather than providing access to their functionalities which are, anyway, wrapped in generic methods.

The standardized orchestrator engines APIs require at least one method that wraps the orchestration. In the case of the elementary services orchestrations, this method corresponds to the protocol request/response. All the orchestrator engines are accessible through a basic *OrchestratorEngine*, which instantiates and returns to them, taking also into account any possible requirements that they may have in technology engines. The *OrchestratorEngine*, on the other hand, is initialized as any other protocol or technology engine.

3.2.4. Engine load and initialization

MXM has been designed to be modular following an engine-level granularity. The use of standard APIs enables applications development regardless of the middleware implementation. Moreover, different engines' instances can be provided in order to support different functionality, provided by each engine. This is achieved through the runtime engine loading feature of MXM, which is based on a configuration file, where the MPEG-M application developer indicates the implementation that should be used for each involved engine. This configuration file is given as an XML document, which is based on a standard schema. An example configuration file is given in Listing 2.

Listing 2: An example of a configuration file for loading and initialization of TEs.

```
<?xml version="1.0" encoding="UTF-8"?>
<mxm:MXMConfiguration xmlns:mxm="org:iso:mpeg:mxm:configuration:schema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="org:iso:mpeg:mxm:configuration:schema mxmconfiguration.xsd">
 <MXMParameters>
  <entry key="username">JohnDoe</entry>
  <ComplexParameter>
   <bar:CustomMXMParam xmlns:bar="urn-x:bar">
    A complex MXM configuration parameter
   </bar:CustomMXMParam>
  </ComplexParameter>
  <MXMEnginesFolder>/usr/bin/MXMEngines</MXMEnginesFolder>
 </MXMParameters>
 <mxm:MXMEngine id="0" type="MPEG21FileTE">
  <ClassName>org.iso.mpeg.mxm.test.MPEG21FileEngine</ClassName>
 </mxm:MXMEngine>
 <mxm:MXMEngine id="1" type="MetadataTE" isDefault="true">
  <ClassName>org.iso.mpeg.mxm.test.GenericMetadataEngine</ClassName>
  <EngineParameters xsi:type="mxm:MXMGenericParameterType">
   <entry key="verbosity">ALL</entry>
  </EngineParameters>
 </mxm:MXMEngine>
 <mxm:MXMEngine id="2" type="MetadataTE">
  <ClassName>org.iso.mpeg.mxm.test.MPEG7SMPMetadataEngine</ClassName>
  <EngineParameters xsi:type="mxm:MXMGenericParameterType">
   <entry key="verbosity">DEBUG</entry>
  </EngineParameters>
 </mxm:MXMEngine>
 <mxm:MXMEngine id="3" type="DigitalItemTE">
  <ClassName>org.iso.mpeg.mxm.test.MPEG7SMPMetadataEngine</ClassName>
  <EngineParameters xsi:type="mxm:MXMGenericParameterType">
   <entry key="verbosity">DEBUG</entry>
  </EngineParameters>
  <EngineDependencies>
   <DependentMXMEngine id="1"/>
  </EngineDependencies>
 </mxm:MXMEngine>
</mxm:MXMConfiguration>
```

Table 1
List of Technology Engines (TEs).

Digital item engine	The Digital Item Engine interface defines the APIs for handling ISO/IEC 21000-2 and ISO/IEC 23000-7 Digital Item Declaration (DID) data structures and providing functionalities, such as: creation of Digital Items and data retrieval from them; and, management of Item, Statement, Descriptor, Component, Resource, License, Metadata and Event Report Requests in Digital Items.
MPEG-21 File format engine	The MPEG-21 File Format Engine interface defines the methods for operating over ISO/IEC 21000-9 MPEG-21 File Format files and providing functionalities, such as: creation of MPEG-21 files and accessing data contained in them.
REL engine	The REL Engine interface defines the methods for handling ISO/IEC 21000-5 Rights Expression Language (REL) expressions and providing functionalities, such as: creation of Rights' Expressions, accessing data contained in them and users' authorization to exercise rights.
IPMP engine	The IPMP Engine interface defines the methods for operating over ISO/IEC 21000-4 and ISO/IEC 23000-5 Intellectual Property Management and Protection (IPMP) data structures and providing functionalities, such as: creation of IPMP data structures and accessing data contained in them.
Media framework engine	The Media Framework Engine is a high level MXM Engine, grouping together several media specific engines, such as: Video, Image, Audio and Graphics Engines. It implements common functionalities (independent on the media type) such as resource loading and saving. The following media specific engines are currently supported: VideoEngine, AudioEngine, ImageEngine, Graphics3DEngine; each of them exposes APIs for creation (encoding) and accessing (decoding) elementary streams. The Media Framework Engine provides functionalities, such as: creation and accessing data to be provided in the aforementioned supported engines.
Metadata engine	The Metadata Engine interface defines the methods for handling creation and management of metadata structures.
Event reporting engine	The Event Reporting Engine interface defines the methods for operating over ISO/IEC 21000-15 Event Reporting data structures.
Security engine	The Security Engine interface defines security-related methods providing functionalities, such as: creation of new credentials and management of public-key based certificates; generation of symmetric keys and encryption/decryption of data; storing of confidential information such as licenses and keys in the secure repository; certification of tools integrity; and, enabling complex authentication protocols.
Search engine	The Search Engine interface defines the methods for operating over metadata structures and providing functionalities, such as: creation and parsing of MPQF query structures.
CEL engine	The CEL Engine interface defines the methods for handling ISO/IEC 21000-20 Contract Expression Language (CEL) expressions and providing functionalities, such as: creation of Contract Expressions and accessing data contained in them.
Overlay engine	The Overlay Engine specifies a minimum set of interfaces that should be implemented by any device participating in a Content Delivery Network. Emphasis is given in peer-to-peer networks, that are quite popular with users for content discovery.

The configuration given above contains some global MXM parameters and listing four MPEG-M Engines: a *DigitalItemEngine* (implemented by class org.iso.mpeg.mxm.test.DIDEngine); an *MPEG21FileEngine* (implemented by class org.iso.mpeg.mxm.test.MPEG21FileEngine); and, two *MetadataEngines* (implemented by classes org.iso.mpeg.mxm.test.MPEG7SMPMetadataEngine and org.iso.mpeg.mxm.test.GenericMetadataEngine).

It should be noted that the "id" attribute of an MPEG-M Engine indicates the identifier of a specific MPEG-M Engine. If two MPEG-M Engines of the same type, as in the case of *MetadataEngine*, are listed in the MXM Configuration File, the "id" attribute is mandatory as it is needed to differentiate the two implementations of the same engine. The default value of the "id" attribute is zero indicating that it is the default engine of that type.

3.3. Part 3 – conformance and reference software (23006–3)

The third part of the standard is about the conformance and reference software. The reference software, a Java implementation of the engines, is licensed under a Berkley Software Distribution (BSD) type of license. The software can be downloaded via Subversion (SVN) software versioning and revision control system from http://wg11.sc29.org/mxmsvn/repos/JAVA/trunk using "*mxmpubro*" as username and "*mpegmxmro*" as password. The APIs and the ESs are given in MPEG-M Part 2 and MPEG-M Part 4, respectively.

3.3.1. MXM project structure

The MXM project structure is dictated by Maven, a tool that provides flexibility in organizing a large project in a modular way along with sophisticated dependency management. The MXM implementation consisted of the following basic modules: (a) MXM Core Module (*mxm-core*); (b) Engines Contained Module (*mxm-engines*); (c) Schemata Container Module (*mxm-dataobject*); (d) Elementary Services Module (*mxm-es*); and, (e) Applications Container Module (*mxm-applications*). In the following these modules are described.

3.3.2. MXM core module

The fundamental (and the only normative) module of MXM reference software is the *mxm-core*, which includes the implementation for the instantiation and dynamic engine loading through the configuration file, whose schema is specified in MPEG-M Part 2. The *mxm-core* is the part of MXM that essentially enables the interoperability between the various implementations of MXM Engines and guarantees the conformance of the software to the standard. This is achieved by using a general container to carry the MXM data units and by automatically instantiating the engines and providing hooks to them only via their standard APIs. Moreover, *mxm-core* is responsible for dependency management between the engines. This is achieved by checking the MXM configuration file for possible dependency circles along with taking care for proper engine initialization so as to make sure that all engines are loaded in the right order. The *mxm-core* module contains all the abstract classes and interfaces that correspond to the engines' APIs specified in MPEG-M Part 2. Any further engines, outside the

MPEG-M context, or extensions of MPEG-M engines in terms of APIs is suggested to be provided in a respective, project-specific, module. For example, the *xyz* middleware, which extends MPEG-M will have a *xyz-core* module, inheriting basic MXM functionality, such as initialization, which will also contain the APIs for any custom engines of *xyz*.

3.3.3. Engines container module

The *mxm-engines* (informative) module is the container for all protocol, technology and orchestrator engines implementation. This implementation should be in line with the APIs specified in MPEG-M Part 2; in particular the engines should implement the full or part of the functionality defined in the corresponding engine APIs. Each engine implementation must implement the corresponding interface, so that the engine can be instantiated by the MXM Core Module. Moreover, an engine may have several different implementations, each included as a different module in *mxm-engines* and identified with a different identifier in the MXM configuration file, used for loading. The application developer can select the engine implementation by means of that identifier provided in the MXM configuration file. The *mxm-engines* module also contains any custom implementations of third-party engines or MPEG-M engines extensions.

3.3.4. Schemata container module

The *mxm-dataobject* (informative) module contains the implementations for handling the XML schemata needed by the MXM engines. As explained in MPEG-M Part 2, the engines include a schema handler module, which provides the APIs for managing any schemata related to the engines. This implementation can be engine-specific and, therefore, the *mxm-dataobject* module is not considered a mandatory component of MXM. However, in the reference software, a JAXB based implementation is provided, which allows easy schema handling, with a similar API to the one of the schema handlers. Moreover, the *mxm-dataobject* module includes the implementation of a generic XML schema creator and parser that enables middleware developers to create interoperable data objects out of the XML elements of the corresponding schemata.

3.3.5. Elementary services module

The *mxm-es* (informative) module includes the implementation of the elementary services, specified in MPEG-M Part 4. MXM provides a reference elementary services implementation using Web Services and the Enterprise Java Beans (EJB3) framework. This, however, is not binding, since other emerging standards may be used as well, such as Representational State Transfer (REST). The reference implementation is application-centered, in the sense that each elementary service makes a call to the orchestrator engine that carries out the corresponding operations to fulfill the application. The elementary service is invoked by the remote module of the protocol engine, which runs on the client side.

3.3.6. Applications container module

The *mxm-application* (informative) module contains a set of reference implementations for applications, which are based on MXM. These applications are provided as examples for MPEG-M application developers to get easier accustomed to the use of MXM reference software, especially with basic features, such as its instantiation and the use of the MXM configuration file.

3.4. Part 4 – elementary services (23006–4)

The forth part of the standard is concerned with the specification of Elementary Services (ESs) and their protocols. The latter being the key elements in achieving services interoperability in the MPEG-M ecosystem.

The holistic view of MPEG-M Part 4 starts from a comprehensive list of Entities and the Operations that can be performed on them, and identifies an ES to be specified whenever a meaningful combination of them exists. ESs are understood as Services of atomic nature, which cannot be usefully divided into smaller ones, but which can be combined to form more complex services. The latter known as Aggregated Services are actually defined in MPEG-M Part 5 (as described in the next section) and realize the concept of MPEG-M Services, which are an integral part of the MPEG-M architecture.

Some of the ESs are considered as regular ESs, that is, when specific Operations are performed on a specific kind of Entity (e.g., Search Content); some as generic ESs, with specified Entity but generic Operations (e.g., Process Content); and, some as abstract ESs, with specified Operation but generic Entities (e.g., Search Entity). A particular implementation of a service is known as a Service Instance, as illustrated in Fig. 4. It is typically described in terms of its provider, connection end-points, and usage conditions. Furthermore, two Service Instances implementing a single ES are shown in Fig. 4; their connection end-points are http://example-a.com and http://example-b.com, respectively. Each Service Instance is described by a Service Instance Declaration (SID) XML document comprising the aforementioned information. Among others, it may contain usage conditions specifying the terms and availability of the Service Instance.

Each of the ESs is described by a narrative description, the protocol specification as an exchange of XML messages, and the syntax (in XML Schema) and semantics of the messages and their parameters. As a formal description of the Service's workflow, the Business Process Model and Notation (BPMN) 2.0 XML (see also section 3.5) representation of the collaboration and process is also given, as well as the extension to the Service Instance Declaration. The machine-readable representation of the Elementary Service workflows is further utilized in Part 5 for the aggregation of Services.

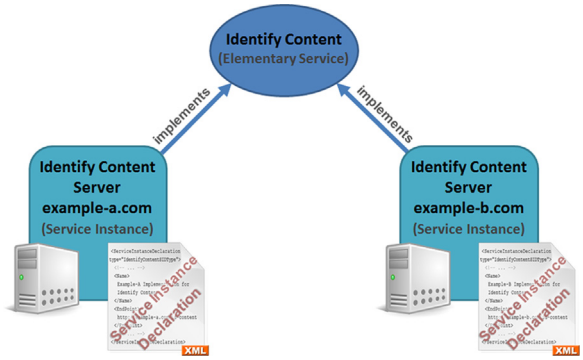

Fig. 4. Multiple Service Instances can Implement the same Elementary Service.

Table 2
Elementary Services classified by Operations and Entities.

	Content	Contract	Device	Event	License	Service	User
Authenticate	X	X					X
Authorize							X
Check With		X			X		
Create	X	X			X		
Deliver	X	X					
Describe	X		X			X	X
Identify	X	X	X		X		X
Negotiate		X			X		
Package	X						
Post	X						
Present		X			X		
Process	X				X		
Request	X	X	X	X	X		
Revoke	X	X			X		
Search	X	X	X		X	X	X
Store	X	X		X	X		
Transact	X				X		
Verify		X	X		X		

In order to remain as open as possible, the standard does not specify the transport protocol, which has to be given in the Service Instance Declaration of each particular implementation, although the use of some HTTP responses is informatively described.

3.4.1. *Table of elementary services*

The list of MPEG-M Entities is restricted to seven elements: Entity (as an abstract generalization), Content, Contract, Device, Event, License, Service and User. The list of Operations comprises eighteen items: Authenticate, Authorize, Check With, Create, Deliver, Describe, Identify, Negotiate, Package, Post, Present, Process, Request, Revoke, Search, Store, Transact and Verify. These Entities and Operations cover the whole spectrum of elements and possible actions in the MPEG-M ecosystem.

MPEG-M Part 4 describes thus a total number of 51 Elementary Services, as shown in Table 2.

Opening the standard to future extensions, Entities and Operations are foreseen to be further specialized generating in turn new Services. In fact, some exemplary extensions as Service Types are given for the Process Content: recognize and synthesize speech, process and translate language, extract sensory information, making a content adaptation or a resource/stream transcoding.

Third parties may define their own Elementary Services based on the MPEG-M architecture and technologies. MPEG-M Part 4 specifies the mandatory information that a third-party ES definition has to exhibit, such as the XML Schema definition of the protocol messages and a description of the behavior of the ES.

3.4.2. *Elementary services in action*

In the following, the usage of an Elementary Service is described based on an example. Assume a content producer wants to register his newest work at a dedicated registration authority in order to have an identifier assigned to this work, a song in this case.

The creator uses the Identify Content ES for this purpose. As most Elementary Services, Identify Content comprises a single request message from the client to the service provider and a response message. The messages can be sent via various

protocols as discussed later. In this example, it is assumed that the request message is sent as an HTTP POST request and the response message is sent as the corresponding HTTP response.

The creator sends a content request to an Identify Content service provider as shown in Listing 3. Note that XML declaration tags and namespace definitions in the XML snippets are omitted for readability reasons.

Listing 3: XML snippet of an Identify Content request message.

```
<IdentifyContentRequest>
 <mpegmb:TransactionIdentifier>60f717d62a3f7589</mpegmb:TransactionIdentifier>
 <dsig:Signature>
  <!-- Signature of the message... -->
 </dsig:Signature>
 <IdentificationContent>
  <didl:DIDL xsi:type="mpegm-didl:DIDLType">
   <mpegm-didl:Item>
    <mpegm-didl:Descriptor>
     <didl:Statement mimeType="application/xml">
      <mpeg7:Mpeg7>
       <mpeg7:DescriptionUnit xsi:type="mpeg7:CreationType">
        <mpeg7:Title>Raining Today</mpeg7:Title>
        <mpeg7:Creator>
         <mpeg7:Role href="urn:mpeg:mpeg7:cs:RoleCS:2001:SINGER">
          <mpeg7:Name>Singer</mpeg7:Name>
         </mpeg7:Role>
         <mpeg7:Agent xsi:type="mpeg7:PersonType">
          <mpeg7:Name>
           <mpeg7:GivenName>Sunny</mpeg7:GivenName>
           <mpeg7:FamilyName>Sunshine</mpeg7:FamilyName>
          </mpeg7:Name>
          <mpeg7:ElectronicAddress
           xsi:type="mpeg7:ExtendedElectronicAddressType">
           <mpeg7:InstantMessagingScreenName service="http://twitter.com">
            @Sunshine_Singer
           </mpeg7:InstantMessagingScreenName>
          </mpeg7:ElectronicAddress>
         </mpeg7:Agent>
        </mpeg7:Creator>
        <mpeg7:CreationCoordinates>
         <mpeg7:Date>
          <mpeg7:TimePoint>2012-11-02</mpeg7:TimePoint>
         </mpeg7:Date>
        </mpeg7:CreationCoordinates>
       </mpeg7:DescriptionUnit>
      </mpeg7:Mpeg7>
     </didl:Statement>
    </mpegm-didl:Descriptor>
    <mpegm-didl:Component>
     <mpegm-didl:Resource mimeType="audio/mp4" encoding="base64">
      <!-- base64-encoded audio file... -->
     </mpegm-didl:Resource>
    </mpegm-didl:Component>
   </mpegm-didl:Item>
  </didl:DIDL>
 </IdentificationContent>
</IdentifyContentRequest>
```

The message contains the content to be identified, which is represented as an MPEG-21 Digital Item (DI) [8]. The DI describes the content by means of a title, its creator, and creation date. Of course, the DI also contains the actual song as an inline MP4 audio bitstream.

The message further has a *TransactionIdentifier*, which ensures that the client can unambiguously match a later response to its request, regardless of the transport protocol. The client also signs the message via a Digital Signature to ensure its authenticity to the service provider.

The service provider receives the request and assigns a new identifier to the content. In the example, the identifier is *doi:10.9999/123.456*. The service provider sends this identifier via a response message to the client as shown in Listing 4.

Listing 4: XML snippet of an Identify Content response message.

```
<IdentifyContentResponse>
 <mpegmb:TransactionIdentifier>60f717d62a3f7589</mpegmb:TransactionIdentifier>
 <dsig:Signature>
  <!-- Signature of the message... -->
 </dsig:Signature>
 <IdentifyContentSuccess>
  <IdentifiedContent>
   <dii:Identifier>doi:10.9999/123.456</dii:Identifier>
  </IdentifiedContent>
 </IdentifyContentSuccess>
</IdentifyContentResponse>
```

Now, that this simple protocol run has been completed, here are some details to explain how the client knows which transport protocol to use for message exchange and how it discovers the service provider in the first place. The Service Instance Declaration (SID), shown in Listing 5, describes a particular implementation of a Service, providing a name and identifier, as well as a communication end point for request messages and which transfer protocol to use for those messages. Two transfer protocols can be currently defined via the *ProtocolBinding* element: XML over HTTP and SOAP [9]. Similar to protocol messages, the authenticity of the SID can be ensured via a Digital Signature. Note that the *xsi:type* of the *ServiceInstanceDeclara*tion element is *IdentifyContentSIDType*, which tells the client that this is an implementation of Identify Content. Depending on the requirements of a Service, the SID may contain further configuration information for the client, e.g., which metadata formats are supported by the service provider.

The client can obtain the SID either from outside the MPEG-M framework, e.g., from a website, or it may use the Search Service ES to search inside a directory for any kinds of MPEG-M Service Instances.

Listing 5: XML snippet for a Service Instance Declaration of a particular implementation of Identify Content.

```
<sid:ServiceInstanceDeclaration xsi:type="sid:IdentifyContentSIDType"
 strictMetadataSupport="true">
 <mpegmb:SIDIdentifier>
  urn-x:example:identify-content:version-1.0
 </mpegmb:SIDIdentifier>
 <sid:Name>Example Implementation for IdentifyContent</sid:Name>
 <sid:Provider xsi:type="mpeg7:OrganizationType">
  <mpeg7:Name>Example Content Registration Authority</mpeg7:Name>
 </sid:Provider>
 <sid:EndPoint>http://example.com/identify-content</sid:EndPoint>
 <sid:ProtocolBinding
  href="urn:mpeg:mpegM:cs:03-es-NS:2012:ProtocolBindingCS:1">
  <mpeg7:Name>XML over HTTP</mpeg7:Name>
 </sid:ProtocolBinding>
 <dsig:Signature>
  <!-- Signature of the SID... -->
 </dsig:Signature>
</sid:ServiceInstanceDeclaration>
```

Similar to Identify Content, most Elementary Services are specified in a straight-forward manner, with a request and a response message. However, there are some Elementary Services that may require more time and protocol steps to complete, such as Negotiate License. Negotiate License allows two parties to reach an agreement on the terms of a license through offers and counter offers. Another example is Store Content, which allows uploading content to a storage device. As this ES is designed in particular for content of large size, the uploading process might be time consuming. For such Services with long-running sessions, two meta protocols exist for (a) session control (i.e., to pause, resume, or abort a session) and (b) status polling.

3.5. Part 5 – service aggregation (23006–5)

The fifth part of the standard specifies how Elementary Services (ESs) and perhaps other existing Aggregated Services (ASs) should be combined to build new ASs. To do so, it provides a methodology which defines the basic steps for the definition of ASs. A set of representative examples are also provided in Part 5 to illustrate new AS definitions. These are briefly summarized next. The "Multimedia Content Registration and Sell" example shows how a content creator can register her multimedia content in order to release it. The "Multimedia Content Search and Purchase" example shows how a user can search some multimedia content, purchase and consume it. The "Create and Identify Content" example shows how a content creator can create and identify some multimedia content. The "TV-Multimedia Processing" example shows how a user can process (e.g., adapt) content. The "Video On Demand Service via Speech Interface" example shows how a user can access some multimedia content using a speech interface. These examples are based on real business scenarios with the aim of demonstrating the applicability of service aggregation.

Part 5 also describes the procedure for registration of both new ESs – not currently present in Part 4 – as well as new ASs. For this purpose, an MPEG-M ESs and ASs Registration Authority (RA) has been set up. Its aim is twofold. On the one hand, services' developers may submit their ESs or ASs compliant information to the RA for registration, while on the other hand, other developers may utilize the RA services to find registered ESs or ASs for reuse.

During ESs and ASs registration, the RA will verify their syntactic correctness and will perform regular checks to verify that the registered information and related links are valid.

The methodology for the ASs definition comprises the following steps:

(1) Provision of a narrative description of the actions that a new AS would perform; in other words, the use case or scenario to be implemented as AS.
(2) Identification of ESs and ASs that are needed by a new AS in order to be implemented. These ESs and ASs could be classified as those: (a) described in Part 4 and Part 5, respectively; (b) registered with the corresponding RA; and, (c) external ESs specifically required by the new AS. The external ESs could also be registered with the RA for further use by third parties.
(3) Provision of a textual description of the AS workflow describing the interactions between the Client and Service Provider.
(4) Resulting AS service workflow formal description provision. It should describe both protocol and service by including the service workflow graphical representation and optional its XML serialization.
(5) Optional registration of the resulting AS with the RA. The RA syntactically validates each registered AS.

For the definition of the AS workflow, the Business Process Model and Notation (BPMN) has been used [10]. It provides a standard notation that is readily understandable by all business stakeholders, including the business analysts who create and refine the processes, the technical developers responsible for implementing the processes, and the business managers who monitor and manage the processes. Consequently, BPMN is intended to serve as common language to bridge the communication gap that frequently occurs between business process design and implementation. It is defined by the Object Management Group (OMG).

Thus, in Part 5, the use of two different service description types (*bpmn:collaboration* and *bpmn:choreography*) based on BPMN has been adopted for illustration purposes regarding the AS definition. However, it should be noted that Part 5 does not impose any restriction for the service workflow representation (step 4 of the methodology) and any graphical workflow notation could be used.

In the following an example is given on how an AS could be defined using the aforementioned methodology:

Step 1. Narrative description
In this use case, a User asks for a sequence of songs satisfying certain "mood", as described in Section 2.
Step 2. Identification of ESs and ASs required by the AS in question
The ESs to be used are: Post Content, Describe Content, Describe User, Manage License, Package Content, Store Content and Deliver Content. All of them are defined in Part 4. Manage License is an AS which makes use of Create License and Store License ESs.
Step 3. Workflow textual description

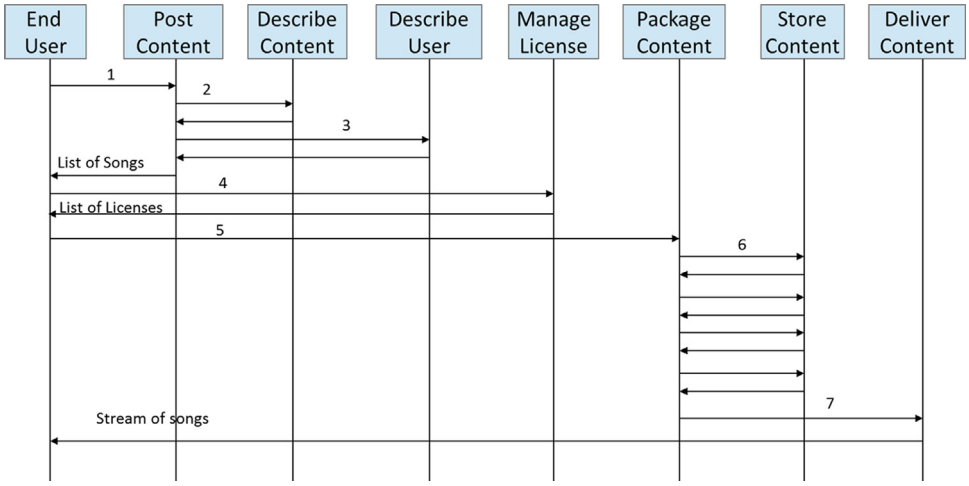

Fig. 5. Music "mood" AS workflow.

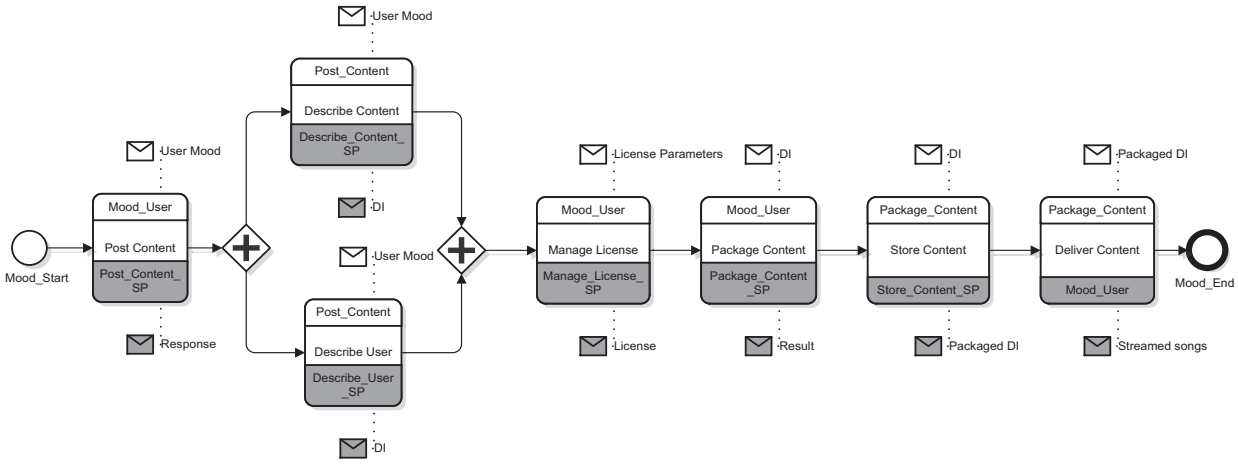

Fig. 6. Music "mood" BPMN diagram.

In the following, the service workflow associated to music "mood" AS is described. Its graphical representation is given in Fig. 5.

- End User contacts Post Content SP to ask for a list of songs according to the friends "mood".
- Post Content SP contacts Describe Content SP to get descriptions.
- Post Content SP contacts Describe User SP to get descriptions. After that, it can prepare a list of songs, which is sent back to the End User.
- End User gets necessary licenses from the Manage License SP.
- The sequence of songs is handed over to Package Content SP.
- Package Content SP will get the Resources from Store Content SP.
- Store Content SP hands over the Packaged Content to the Deliver Content SP who streams the Packaged Content to the End User.

Step 4. Formal description of the service workflow

In this example, BPMN 2.0 is used for the formal description of the proposed AS. The corresponding diagram is shown in Fig. 6.

4. Related developments

- Open Connected TV (OCTV) has been set up by the Digital Media Project (DMP). The outcome of OCTV is not a complete product or a running service, but a commercial-grade implementation of MPEG-M software that may be used direct by DMP members who implement commercial products and services. [Online]. Available: http://octv.dmpf.org/
  In particular, using OCTV platform, a Creator operates a *Source* to upload videos and corresponding metadata to a *Server.*

The *Server* prepares the videos for MPEG-DASH streaming and presents them together with their corresponding metadata to a web page so that an End User can operate a *Sink* to view information on videos and possibly request MPEG-DASH streaming of a specific video. The *Source, Server and Sink* OCTV devices make use of the following MPEG-M engines: Digital Item, Rights Expression Language, Media Framework, Metadata and MPEG-21 File Format.
Furthermore, WimTV (Web/Internet/Mobile TV) is an ICT platform to support a digital media ecosystem that enables diffuse trading and distribution of video content. Media move from one user to the next with associated terms of use and payment conditions. The WimTV server implements a number of MPEG-M engines and exposes a set of application-oriented API. [Online]. Available: http://www.wim.tv/

- Another MPEG-M compliant implementation for secure management and distribution of multimedia content, known as MIPAMS, has been developed by DMAG-UPC [11]. This platform provides several MPEG-M Elementary Services that are accessible by third parties to further develop their own Aggregated Services. [Online]. Available: http://dmag.ac.upc.edu/mipams
  In particular, the MIPAMS platform has been successfully used for the development of a mobile application for the "Multimedia Content Search and Purchase" Aggregated Service, mentioned in Part 5 of the MPEG-M standard. This Aggregated Service enables a user to search, purchase and consume premium multimedia content on a mobile device. In this case, the mobile application is the front-end interface to the MPEG-M middleware providing the user with the requested functionality (i.e., content search, licensing capabilities and event reporting). To do so, the mobile application accesses several ESs in the MPEG-M middleware, such as: Authenticate User, Search License, Present License, Create License and Authorize User.
  The challenge faced by implementing this mobile application was to demonstrate that MPEG-M ESs were able to run over the existing MIPAMS platform. This has been achieved by implementing a *mapping process* between MPEG-M ESs and MIPAMS operations. Integration was made on the data layer, as MIPAMS is based on the same MPEG-21 standards as MPEG-M. However, since the format of MPEG-M ESs' parameters and the respective MIPAMS operations' parameters can be different in certain cases, the *mapping process* is needed. For instance, in an MPEG-M ES, an MPEG-21 REL license alone may be included as parameter, while, in the corresponding MIPAMS operation, the MPEG-21 REL license together with its version number, may be both required as parameters, before the MIPAMS operation is called and executed. In this case, the *mapping process* adapts the MPEG-M ES's parameters to match the MIPAMS operation's parameters, that is, the license together with its version number. Similarly, the *mapping process* is also applied to transform the responses of MIPAMS operations back to those required by MPEG-M, when needed.
  It should also be noted that the aforementioned mobile application has been developed for the Android mobile platform. However, the same *mapping* approach can also be extended to other mobile or web based platforms providing – through MIPAMS platform – to third party application developers MPEG-M compliant ESs and ASs.
- The EC funded project ALICANTE (FP7-248652), deploys MPEG-M Elementary Services on the Service Provider side of the media handling chain to manage the distribution of content to end-user devices. It also defines additional Elementary Services for managing multimedia service offerings. [Online]. Available: http://ict-alicante.eu/
  In particular, the ALICANTE project developed a new media ecosystem with a strong collaboration between service/content providers and end users. This media ecosystem relies on content-aware networking and context-awareness towards multimedia service offerings in order to enable adaptive media delivery with advanced Quality of Experience (QoE) management. Multimedia service offerings (which are called *Services* in the ALICANTE architecture), can range from different streaming techniques such as MPEG-DASH or RTP multicast to interactive media usage or even peer-assisted streaming scenarios.
  The challenge faced by the project was the interoperable registration and management of multimedia service offerings for underlying content. In particular, the type of offering and any associated configuration parameters that should be made available to the end user.
  This challenge was addressed by adopting MPEG-M Elementary Services for Post Content and Search Content, as well as by defining counterparts for managing multimedia service offerings (which may comprise one or more multimedia resources). Based on these protocols, multimedia service offerings from multiple sources can be enlisted by a common registration entity. End users can search for these offerings, filter the results by the appropriate type of offering, retrieve the configuration parameters, and – based on these parameters – establish the connections for accessing the content. The content is then dynamically adapted to the usage context (e.g., display resolution) and available resources during media delivery.
- The EC funded project CONVERGENCE (FP7-257123), offers an MPEG-M based platform with additional Aggregated Services supporting publish, control, search for, and content usage, where users are able to define their own policies on it, thus, enabling new business models on content usage. [Online]. Available: http://www.ict-convergence.eu/
  In particular, the project faced the challenge of deploying a publish/subscribe architecture on top of an Information Centric Network layer (ICN): users can subscribe to content that matches complex semantic queries (i.e., get movies tagged '*romantic*' may also return movies tagged 'sentimental' and 'love', that is, the synonyms of '*romantic*'), and peers participating in a collaborative network fetch it from the ICN and deliver it to end-devices. The MPEG-M and MPEG-21 standards have been adopted and extended to elegantly achieve this goal by:
  Firstly, CONVERGENCE introduced an enhanced version of the DI concept, named VDI for Versatile DI. It exploits RDF/OWL semantically typed relationships between DIs, and DI versioning, building upon the Digital Item Engine.

Secondly, it uses a gossiping protocol over an overlay of peers to exchange and store state about publications and subscriptions. The project started by specifying the *OverlayEngine*, a generic framework able to support both structured and unstructured protocols, and then provided reference software of an *OverlayEngineOrchestrator*, implementing a gossip protocol that exploits the underlying ICN infrastructure and offers *PublishContent* and *SubscribeContent* aggregated services. The information to be gossiped is gathered from the Resource VDI, packaged into a transport package and gossiped from peer to peer. The transport package is also a VDI, the *Publication VDI (P-VDI). Subscription VDIs (S-VDIs)*, issued by users searching for specific resources, play a role symmetrical to that of Publication VDIs and also carry a license (REL Engine) and an expiry date. The introduction of Publication and Subscription VDIs decouples the functionalities of the middleware and the network level; the two levels are independently managed and this makes the system more robust.

Thirdly, peers can perform matches between P-VDIs and S-VDIs and communicate any match to specified peers in the form of ER (Event Reports), depending on licenses and ERRs (Event Report Requests). S-VDIs contain one or more representations of the semantic subscription to a set of resources, in the form of a SPARQL query or a list of requested metadata, embedded in MPQF (thus bridging with yet another MPEG standard). Two new MPEG-M compliant engines, the Match TE and CDS TE (Common Dictionary Service) have been developed to perform the matching of semantic constructs and their expansion by means of additional, external, well-known or custom supplied supporting taxonomies.

- A collaborative music analysis, visualization, annotation and repurposing service is also currently under development based on Sonic Visualiser [12] integrated into MPEG-M digital media trading platform. Sonic Visualiser supports multi-track audio with volume sliders for DJ mixing and lyrics for Karaoke applications, thanks to MPEG-A: Interactive Music Application Format (IM AF) [13–15], as well as chords and melody extraction, automatic audio tracks alignment and audio effects. [Online]. Available: http://www.isophonics.net/SonicVisualiser

  Sonic Visualiser is an open source cross-platform framework for analysis of music and audio. It has been developed to assist musicologists, music information retrieval and signal processing researchers and anyone else looking for a friendly way to study at what lies inside an audio file. An essential strength of Sonic Visualiser is its ability to support third-party plug-ins, typically referred to as *audio analysis plug-ins* or *audio feature extraction plug-ins*. Its native VAMP plug-in format has been adopted by several other influential projects, including Audacity and Marsyas. Typical VAMP plug-ins include chords and melody extraction, automatic tracks alignment and audio effects.

  The MPEG-A: Interactive Music Application Format (IM AF) by providing multi-track audio with volume sliders for DJ mixing and lyrics for Karaoke applications, enables an environment that activates and stimulates passive listeners, to: (a) enjoy enhanced listening experience, (b) develop musical/voice instrument skills through active learning, and, last but not least, (c) become creative music producers.

  The integration of IM AF codec in Sonic Visualiser has recently been achieved – thanks to a number of students' projects [16] – allowing the combination of their features (i.e., automatic chords extraction aligned with lyrics). However, the challenge of Sonic Visualiser integration to MPEG-M digital media trading platform it is still work under progress. The latter, would not only attract music fans but also create new revenue opportunities for all parties involved in the music value chain.

5. Discussion and future developments

MPEG-M origins may even be traced back to couple of EC funded projects [17], whose evolution, heavily influenced by DMP's vision and coordination efforts, as well as contributions – among others – from a number of PhD theses [18–21], led to OCTV and MIPAMS platforms. Start-ups influenced by MPEG-M have also been emerged, such as: http://www.wim.tv, http://www.mediatg.com, http://www.bitmovin.net and http://www.netmust.eu.

This diversity of contributions by projects and people involved in MPEG-M standardization process has imposed several requirements and challenges, which the MPEG-M ad hoc group aspired to address in their whole.

5.1. The plurality of metadata formats challenge

A key challenge faced by the MPEG-M ad hoc group was the handling of metadata encapsulated in a Digital Item (DI), since they can be used in several contexts; from content discovery to even more practical cases of providing details useful to media players or other applications rendering the content. Given the variety of different metadata formats available (i.e., MPEG-7, TV-Anytime and EBUCore) requiring different technologies to manipulate them, it has been decided to decouple the DI from metadata handling, by providing separate engines; moreover, the Metadata TE has been designed considering the generic functionality that should be provided by a standard, so that it can be extended to support any metadata format and related handling technology, while also the other engines can still make use of the standard API.

5.2. The inter-dependencies of technology engines challenge

The fine granularity of MPEG-M engines, even though it has advantages – the aforementioned case of decoupling the Digital Item TE from the Metadata TE being a representative example – also introduces dependencies between them, which can lead to the creation of inter-dependency cycles in their implementation. This challenge has been addressed by the MPEG-M ad hoc group by defining the notion of an engines' *orchestrator*, which is capable of providing complex

functionality by combining a number of individual engines; as a result, information and functionality overloading in individual engines' implementations is avoided. Moreover, MXMConfiguration file schema allows the definition of engines' dependencies, which is used by the reference software to automatically perform engine loading in the proper sequence, while also tracing and providing feedback on dependency cycles.

5.3. The service aggregation workflow language challenge

This granularity has not only been followed by the MPEG-M engines but also in the elementary services specification. For this reason, the standard has devoted a specific part on specifying how service aggregation should be performed by combining a number of elementary ones. However, even though there are several service aggregation workflow languages, it has been decided by the MPEG-M ad hoc group to not rely on any specific one, due to that each one comes with its own pros and cons [26]. Thus, the selection of the service workflow language could be decided by the application developers, separately, depending on the application domain. Therefore, only the guidelines for service aggregation are provided by the standard, while BPMN [10] is used for informative examples.

MPEG-M evolution is ongoing since standards are living 'organisms' that need to be adapted to emerging industry needs. Thus, couple of ISO/IEC MPEG activities closely related to MPEG-M are worth to be mentioned. These activities will produce new MPEG-M engines and APIs to be used by applications developers, for (a) media recommendation, and (b) virtual worlds services. In the following, these are briefly explained.

- The first one, so called MPEG-UD (User Description) [22], is about enabling interoperable recommendation services. For example, Smart TVs offering a large number of both internet and broadcasting channels to the consumer. In such an environment, TV manufacturers consider including recommendation services within the TV set, in order to help their customers select a program (Service Description) according to their preferences (User Description) and context (Context Description). In such a scenario, there are benefits for the recommendation engines to be offered by third parties. However, this could only be achieved if standardised APIs exist supported by TV manufacturers. Then, a plurality of recommendation engine providers would be available for the consumer to choose from. In particular, the standardised APIs ensure the following benefits: (a) TV manufacturers are not constrained to a specific recommendation engine provider neither are needed to build one by themselves; (b) consumers have the option of selecting a recommendation engine that better fits their needs; and (c) recommendation engine providers would compete to provide better services, without being tied to a specific TV manufacturer.
- The second one, so called MPEG-V (Media Context and Control) [23–25] provides the architecture and specifies the associated information representations to enable interoperable multimedia and multimodal communication within *Virtual Worlds* and the real world. For example, it may be used to provide multisensory content associated to traditional audio/video data to enrich multimedia presentations with sensory effects created by e.g., lights, winds, sprays, tactile sensations and scents; or it may be used to interact with a multimedia scene by using advanced interaction paradigms such as hand/body gestures; or to access different virtual worlds with an avatar which looks similar in all of them. As a result, in MPEG-V a piece of digital content is not limited to an audio/video asset, but may be a collection of multimedia and multimodal objects forming a scene, having their own behavior, capturing their context, producing effects in the real world and interacting with one or several users. Moreover, MPEG-V addresses the need for interoperability between virtual worlds – and between any of them and the real world – by describing virtual objects, and specifically avatars to be '*teleported*' from one virtual world to another.

At last but not least, MPEG-M is a standards-enabled digital media ecosystem facilitating interoperable applications and services for the benefit of all stakeholders in media value chain: creators, service providers and consumers. Innovative services and business models could take advantage of MPEG-M middleware APIs and service aggregation mechanisms. With respect to the latter – perhaps as an ultimate challenge to be addressed by an existing or a forthcoming start-up, that in turn it could significantly affect MPEG-M's widespread use – of particular interesting is the seamless and hassle free service aggregation using graphical notation; that is, enabling not only application developers but any actor in the MPEG-M media value chain – *with no need for prerequisite knowledge of programming languages* – to build their innovative business models on the fly by linking boxes (representing predefined TEs and ESs) with their finger on a tablet's touch screen!

6. Conclusions

Streaming audio and video now dominate net traffic, constituting around half of all data on tablets and smart phones. Advances in compression, the spread of Wi-Fi, mobile connectivity, and the proliferation of devices have created commercial potential but also pose threats to some existing creative businesses. As a result of this, cross-platform production is becoming increasingly central to content businesses. Furthermore, with the continued emergence of new platforms, products and services, technical and service interoperability will be increasingly important for creative businesses. MPEG-M ensures that services and content work across different devices and environments to suit consumer's demands and expectations. MPEG-M by addressing interoperability, help businesses develop new technologies, products and services in

response to the challenges and opportunities arising from convergence: authoring, producing and distributing media, as well as advanced IPTV services, across multiple platforms and devices.

7. Standard

ISO/IEC JTC1/SC29/WG11 Information Technology – Multimedia Service Platform Technologies (23006):

1. ISO/IEC 23006-1:2013 Information technology – Multimedia Service Platform Technologies – Part 1: Architecture.
2. ISO/IEC 23006-2:2013 Information technology – Multimedia Service Platform Technologies – Part 2: MPEG Extensible Middleware API.
3. ISO/IEC 23006-3:2013 Information technology – Multimedia Service Platform Technologies – Part 3: Reference software and conformance.
4. ISO/IEC 23006-4:2013 Information technology – Multimedia Service Platform Technologies – Part 4: Elementary Services.
5. ISO/IEC 23006-5:2013 Information technology – Multimedia Service Platform Technologies – Part 5: Service Aggregation.

Acknowledgments

This work was partially supported by the European Commission under contracts FP7-257123 (CONVERGENCE project) and FP7-248652 (ALICANTE project); and, by the Spanish Ministry of Economy and Competitiveness under contract TEC2011-22989 (PBInt project). Panos Kudumakis would also like to acknowledge that this work has been partially done during his visit at the University of Malaga in the context of the program Andalucía TECH: Campus of International Excellence.

References

[1] iOS, Apple (Online), Available: ⟨http://developer.apple.com/⟩ (last accessed 14.03.13).
[2] Android, Google (Online), Available: ⟨http://developer.android.com/⟩ (last accessed 14.03.13).
[3] L. Chiariglione, MPEG Technologies (Online), Available: ⟨http://mpeg.chiariglione.org/technologies⟩ (last accessed: 14.03.13).
[4] ISO/IEC 23008-2:2013 MPEG-H Part 2 and ITU-T H.265 – High Efficiency Video Coding (HEVC).
[5] ISO/IEC 23009:2012 Information Technology – Dynamic Adaptive Streaming Over HTTP (DASH).
[6] P. Kudumakis, X. Wang, S. Matone, M. Sandler, MPEG-M: multimedia service platform technologies, Signal Process. Mag., IEEE 28 (6) (2011) 159–163, http://dx.doi.org/10.1109/MSP.2011.942296. (Nov.).
[7] SoundBite: Semantic music playlist generator, Centre for Digital Music, Queen Mary University of London. (Online). Available: ⟨http://isophonics.net/content/soundbite⟩ (last accessed 14.03.13).
[8] ISO/IEC 21000-2:2005 Information Technology – Multimedia Framework (MPEG-21) – Part 2: Digital Item Declaration.
[9] SOAP Version 1.2 Part 1: Messaging Framework, Second ed., W3C Recommendation, Apr. 2007, (Online), Available: ⟨http://www.w3.org/TR/soap12-part1/⟩ (last accessed 14.03.13).
[10] Business Process Model and Notation (BPMN), Version 2.0, Object Management Group, Jan. 2011, (Online), Available: ⟨http://www.omg.org/spec/BPMN/2.0/⟩ (last accessed: 14.03.13).
[11] S. Llorente, E. Rodriguez, J. Delgado, V. Torres-Padrosa, Standards-based architectures for content management (Online), MultiMedia IEEE; (99) (2012) http://dx.doi.org/10.1109/MMUL.2012.58. (1-1).
[12] Sonic Visualiser, Centre for Digital Music, Queen Mary University of London. (Online). Available: ⟨http://www.isophonics.net/SonicVisualiser⟩ (last accessed: 14.03.13).
[13] ISO/IEC 23000-12:2012 Information Technology – Multimedia Application Format (MPEG-A) – Part 12: Interactive Music Application Format (IM AF).
[14] I. Jang, P. Kudumakis, M. Sandler, K. Kang, The MPEG interactive music application format standard, Signal Process. Mag., IEEE 28 (1) (2011) 150–154, http://dx.doi.org/10.1109/MSP.2010.939073. (Jan.).
[15] G. Herrero, P. Kudumakis, L.J. Tardon, I. Barbancho, M. Sandler, An HTML5 Interactive (MPEG-A IM AF) Music Player, in: Proceedings of the 10th International Symposium on Computer Music Multidisciplinary Research (CMMR), Marseille, France, 15–18 Oct. 2013.
[16] P. Kudumakis, MPEG developments (Online), Available: ⟨https://code.soundsoftware.ac.uk/projects/mpegdevelopments⟩ (last accessed 11.10.13).
[17] C. Serrão, JMS. Dias, P. Kudumakis, From OPIMA to MPEG IPMP-X: A standard's history across R&D projects, Signal Process: Image Commun. vol. 20 (9-10) (2005) 972–994.
[18] Carlos Serrão, iDRM – Interoperability mechanisms for open rights management platforms (Ph.D. thesis), Universitat Politècnica de Catalunya, Barcelona, Spain, 2008.
[19] Filippo Chiariglione, Technologies and platform to manage rights and value of digital media (Ph.D. thesis), Università degli Studi di Firenze, Florence, Italy, 2009.
[20] Víctor Rodríguez Doncel, Semantic representation and enforcement of electronic contracts on audiovisual content (Ph.D. thesis), Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, 2010.
[21] Michael Grafl, Scalable media delivery chain with distributed adaptation (Ph.D. thesis), Alpen-Adria-Universität, Klagenfurt, Austria, 2013.
[22] Call for Proposals on MPEG User Description, ISO/IEC JTC1/SC29/WG11/N13879, Vienna, Aug. 2013.
[23] ISO/IEC 23005:2011 Information technology – Media Context and Control (MPEG-V).
[24] Marius Preda, Francisco Morán Burgos, Christian Timmerer, (Eds.), MPEG-V, Signal Process: Image Commun. 28 (2) (2013).
[25] WD 1.0 of MPEG-V Engines API, ISO/IEC JTC1/SC29/WG11/N13802, Vienna, Aug. 2013.
[26] M. Koukovini, E. Papagiannakopoulou, G.V. Lioudakis, N. Dellas, D.I. Kaklamani, I.S. Venieris, An Ontology-Based Approach towards Comprehensive Workflow Modelling, IET Software, 2013.