SOMMIT: Software Open MultiMedia Interactive Terminal

GOALS AND TECHNICAL APPROACH
The aim of the SOMMIT project was to develop a software architecture for interactive multimedia terminals independent of hardware, operating system, network, and delivery medium, and furthermore extensible, scalable, upgradable. These issues, related to the easy portability of both the software architecture implementation as well as the applications developed to operate on it, were solved with the combination and integration of commercial available solutions. Next, the project continued into the area of actually developing enriched applications. It was derived from market studies that an evolutionary approach should be developed by first utilising the “traditional” broadcast model. This will bring the end-users closer to, and become familiar with, the “Interactive digital multimedia services”—the SOMMIT area of interest—staying within a well known and everyday used environment. The selected approach is alternative to that one of immediately pointing to an entirely new bidirectional interactive broadband service, much more demanding in terms of investments for infrastructure, servers, contents and application development due to the need of introducing a return channel. In addition, it is possible to build on existing equipment, at least for new simple applications.

MPEG, DAVIC/DVB and OPIMA were the Standardisation/Specification Bodies considering the issues relevant to the SOMMIT goals and thus the project activities were made consistent with their work programmes.

During its research activity, SOMMIT submitted significant contributions to MPEG, DAVIC and OPIMA.

Concerning the platform, a solution incorporating a (Java) Virtual Machine, was considered suitable to solve some of the problems that arose in these standardisation fora.

In MPEG, in more detail in MPEG-4 Version 2, the project has contributed significantly to the MPEG-J part that is aiming at providing higher level of flexibility and interactivity to the MPEG-4 application domain. In DAVIC, SOMMIT has submitted some Java APIs to Part 9 that are suitable for the broadcast environment. For OPIMA the project has made a significant contribution to the architecture development process based on the experience gained w.r.t. conditional access systems and the application of virtual machine technology.

SOMMIT has realised a demonstrator that can host different kinds of applications, as it will be described in the following chapters. This implementation actually has anticipated part of the specifications that were issued by the Advanced Interactive Content Initiative (AIC-I). The approach adopted by the AIC-I enables the use of MPEG-4 applications carried over the MPEG-2 TS in broadcast-based environment. The role of SOMMIT in this international initiative was to drive the implementation of the specifications, and to form consensus about the way of exploiting the MPEG-4 standard.
The analysis of the situation at the end of the Project allows us to state that the Consortium was right in its choices and the Project produced successful results at the various levels of its activity: consensus forming, API specifications, technology implementation and service definition. In this way the Project has contributed to the development of knowledge and solutions that are beneficial to the entire European Community.

**Contributions of SOMMIT in MPEG-4**

The contribution of the Project to the MPEG standardization activity is twofold: on one hand the implementation of complex applications using MPEG-4 (EPG and MPEG-2 enhancement service) proved the soundness and the applicability of the specification. On the other hand, the specification of some API for the Version 2 of the standard (MPEG-J part) allowed programmatic content to be delivered with the MPEG-4 stream as a Java applet to control the MPEG terminal.

**SOMMIT Applications Based on the MPEG-4 Standard**

Here we are going to describe some of the applications that were developed by the Project. To exploit MPEG facilities SOMMIT realized an EPG in Java, a C++ application for the download of the MPEG-4 scenes, and the integration of an existing MPEG-4 player with an MPEG-2 decoder.

- **The SOMMIT EPG Application**

  This application implements an Electronic Program Guide (EPG) and runs on the terminal that has been prototyped by the Project.

  This application extracts TV program information from an MPEG-2 Transport stream [1] and allows the selection of one of its programs.

  The EPG GUI (Graphic User Interface) provides the user with one 3D tool for the navigation among the available TV programs and for the sorting of them according to the user preferences. For example, the user interacts with a 3D scene to select only the sport programs that are broadcast the same day. In this case MPEG-4 is used to build the 3D scene representing the GUI. Then, the requested information is displayed in 2D format, along with a textual description of the event and a thumbnail picture associated to it. The picture can be either still or animated, according to the bandwidth of the channel.

  The 3D tool offered to the user is not an immersive 3D world. This kind of metaphor was rejected for two reasons:

  - it may be difficult for a non accustomed person to use it with a simple remote control;
  - in terms of usability, the time spent to select a TV program could be felt too long and not really efficient.

  Hence, another 3D metaphor has been selected: the Rubik Cube metaphor. The users can rotate the blocks in the 3D world in order that the faces in front of them display their choice.
The MPEG-4 3D world is controlled by a Java applet through a set of API (called Scene Graph API) that has been standardized in MPEG.

The selection of search criteria allows the user to find a set of real-time data coming from the MPEG-2 transport stream. These data comply with the DVB-SI (Digital Video Broadcast – Service Information) standard [6] and are composed of the following tables: NIT (Network Information Table), SDT (Service Description Table), BAT (Bouquet Association Table), EIT/PF, EIT/Schedule (Event Information Tables).

This information is displayed as a list of available programs. Then, the user can select one entry of the list and have further details, sometimes with a moving thumbnail of the program, and the possibility of switching to the program, if this is broadcast at the same moment.

The implementation of this application required a big effort also for the definition and implementation of the terminal on which it runs, in particular to ensure that the prototype was compliant to the recommendations of the standardization bodies.
• **MPEG-4 Application Downloading over MPEG-2**
This was one of the first applications the Project prototyped to show how MPEG-2 content can be enhanced with MPEG-4 applications. The MPEG-4 content is supposed to carry additional information related to the one that is displayed in the MPEG-2 main movie.

While the user is watching an MPEG-2 A/V stream, a private data stream in a format defined by the Project carries standard MPEG-4 applications. If the user wishes, he/she can be notified as soon as a new application has been completely downloaded. At some time the user can decide to launch one of the MPEG-4 applications that were downloaded. Depending on the profile this application is compliant to, either an MPEG-4 2D player or an MPEG-4 3D player is launched.

The players are outside the scope of the Project; the 2D player is taken from the IM-1 group in the MPEG community, while the 3D player is the result of another project.

• **MPEG-4 Application Streaming over MPEG-2**
This application shows how existing digital TV (MPEG-2 A/V carried over MPEG-2 TS) can be enriched by overlaying interactive information (MPEG-4 carried by the same MPEG-2 TS), that is synchronized with the main program.

The program starts receiving the MPEG-2 TS from the network and decodes in software MPEG-2 video and MPEG-1 audio.

A specific DMIF instance of the MPEG-4 player retrieves from the MPEG-2 TS some interactive information in MPEG-4 format and overlays it over MPEG-2. The user cannot easily tell what the usual information is and what the enhancement content offers (the
button “Click to view the goal”). This is what we call “transparent integration”.

If the user has a normal (non-enhanced) MPEG-2 STB, the decoding process is not affected by the additional information in the stream: MPEG-4 packets are simply ignored and the user does not see the button “Click to view the goal”. This is what we call “transparent pass-through”.

Enhancement information is presented to the user at appropriate times: MPEG-4 content is synchronized with MPEG-2.

While we are watching this volley match, a football match is running on another channel. We have subscribed to any important event that may occur on the other football channel (how we subscribed is out of the scope of this demonstration). This is one of the results of our subscription: Roberto Baggio has just scored a goal and you can see the goal as a picture-in-picture (PIP), if you want, just by clicking on the Baggio’s face.

Here the stream has been multiplexed off-line, while in live events someone should monitor the other channel for special events (such as goals) and insert the enhancement information as soon as the event occurs.

Figure 42: The MPEG-2 stream with the enhancement information
On request we can watch the goal as a picture-in-picture, without losing any moment of the volley match.

After the end of the goal, both the PIP and the button disappear leaving full view of the volley match.

With exactly the same mechanism we can provide additional information for example about one of the players of the volley match, on user request. This information may be offered by a sponsor, for example.

**MPEG-J API**

The idea to have a kind of “intelligence” associated with a basic or even complex multimedia application is not new. You can view over the Internet a lot of pages with video, audio, graphics and VRML scenes that can also perform “actions” even in a complex way. Mainly all these functionality are made possible by using a suitable program language to instruct the Web browser (or and independent execution engine). Two program languages are mainly used to associate to Web pages a sort of “intelligence”: Java and ECMAScript (was JavaScript) [2] and [3].

**Figure 43:** Additional interactive information about the volley players
In its unifying effort, MPEG-4 took this idea and these tools to describe a powerful framework for a multimedia system suitable both for the broadcast environment (service-driven) and for the Internet (application-driven environment).

To realize the Java based MPEG-4 framework, an Ad Hoc Group was created within MPEG-4 and its name is MPEG-J (MPEG-Java) AHG. [5]

To allow the Java application/applet to interact with the MPEG-4 world, application program interfaces (API's) to the various components of an MPEG-4 terminal are defined. Furthermore, MPEG-J describes the delivery of application programs.

Why Java? The choice of the Java language as a means of putting intelligence in MPEG-4 application was driven by a set of considerations that relates to the industrial usability of the MPEG-J system. Features of portability (platform independence) and object orientation have been carefully taken in consideration, but also the Java security model has been appreciated. Moreover, Java is a complete language, providing multithreading capabilities and interfaces to the native platform, a set of features that distinguish it from a basic scripting tool.

Modification of the IM-1 2D player to decode MPEG-J information was in the scope of the Project. We added the MPEG-J decoder which instantiated a Java Virtual Machine in the same address space of the 2D player.

The MPEG-J 2D player implements most APIs (Network, Capabilities, Decoder) defined in the MPEG-J subgroup of MPEG, proposed by the Project and accepted, and allows to play both local Java applications and applications contained in the MPEG stream, called MPEG-let.

The MPEG-J 3D player implements also the EAI (External Authoring Interface) API also used in MPEG.

**The Streaming of the Java Applications in the MPEG-4 Terminal**

The byte code, compressed in a zip archive, is streamed as Access Units with time stamps. After the multiplexing of the components in an MPEG-4 stream, this can be delivered to the MPEG-4 terminal to be played. The basic steps for the execution of the MPEG-J Application are described in the following paragraph.

If an application program is delivered as an MPEG-4 elementary stream, the MPEG-4 Player is notified by the reception of an MPEG-J stream Object Descriptor.

An MPEG-J session can thus be initiated as follow:
- The MPEG-4 player starts an MPEG-J session. This will involve:
- Instantiation of the Java Virtual Machine;
- Instantiation of the Class Loader,
- The MPEG-J decoder reassembles all the received packets containing MPEG-4 data in order to have the complete MPEG-J application ready to by executed.
- The MPEG-J decoder loads the application/applet, and starts it. There can be more than one class with an entry point within one MPEG-J stream. Each time such a class
containing an entry-point is received (a “run” method), execution will start there as a new thread.
The following set of API specifications was submitted to the MPEG-J ad-hoc group on their request for the definition of a minimal set of APIs to control the terminal. Now these API are part of Version 2 of the MPEG-4 standard or the DAVIC/DVB specifications.

- **The Section Filter API**
The objective of the Section Filter API is to provide a general mechanism allowing access to data held in MPEG-2 private sections. A mechanism is provided for inter-operable access to data, which is too specialized to be supported by -for instance- a high level Service Information API or which is not actually related to service information. The aim of the package is to provide a platform-neutral interface, which allows access to the MPEG-2 private sections present in an MPEG-2 Transport Stream. The package allows an application to create section filters, to connect section filters to a section source (Transport Stream) and to manage connecting and disconnecting of resources.

- **The Service Information API**
The Service Information API allows retrieval and monitoring of SI (Service Information) data. It retrieves information on the network and on the available services from the resident network and service database module. Actually, the API should filter this information from the transport stream when requested, but the data is considered stable enough that using the resident database information is acceptable. The SI API does filter the other tables from the transport stream when requested, by using the PSI/SI database module. It should be mentioned that the Service Information API, the access to the Service Information database, is a specialization of the Section Filter. It can be considered as being implemented on top of the Section Filter. Performance considerations in systems that only require Service Information however might suggest implementing the Service Information directly on the MPEG-2 transport Stream. The Service Information API could on one hand be considered as an example of a specialized interface to the private section and on the other hand as a recommendation for access to the Service information data when required.

  This API is in the DVB/DAVIC standard and is referenced in MPEG.

- **The Network API**
The MPEG-J Network API allows simple control and statistical monitoring of the network component of the MPEG-4 player (Access Layer).

  Through these APIs Java applications have the possibility of interacting with the Access Layer. This layer manages all the network resources and all the communication channels of the player toward the network (such a module uses the services provided by a particular subsystem of the MPEG-4 player called DMIF—Delivery Multimedia
Integration Framework.) [1].

Because the level of abstraction provided by the MPEG-J Network APIs, the applications are unaware of the kind of connections that they are using (connections to LAN, WAN, Broadcast channels, local disks and so on) to access to a service.

The functionality provided by the architectural model that we proposed for the MPEG-J APIs can be split in two major groups:

• Network query: the ability to perform requests to the network module in order to get statistical information about the DMIF resources used by the MPEG-4 player has been recognized as an important feature.

• Channels control: a simple channel control mechanism is also provided. Using this feature an MPEG-J application can temporarily disable or enable existing Elementary Stream channel without any negative influence on the rest of the player. This feature fits with one of the general requirements of MPEG-J, that is, the capability to allow graceful degradation under limited or time varying resources [4].

The Terminal Capability & Profiles API

Program execution may be contingent upon the terminal configuration and its capabilities. An MPEG-J application may need to be aware of its environment, so that it can adapt its own execution and the execution of the various components, as they may be configured and running in the MPEG-4 terminal. The Terminal Capability API is responsible to provide access to dynamic and static terminal resources. A separation between static and dynamic terminal capabilities has been reflected in the API. As applications need to be notified when terminal capabilities change, an additional interface for that purpose has been defined (terminal capabilities could change dynamically when non MPEG-4 related applications run with the MPEG-4 terminal at the same time). Depending of the hosting machine it is possible that other running applications exist beyond the MPEG-4 one (for PC platform as an example but for Set Top Boxes this may not be true).

The purpose of the Profile API is to provide a facility that allows applications to find out what is the profile/level supported by the terminal where the application runs. Because profiles/levels are defined in the MPEG-4 standard and an application knows the terminal profile/level it can decide how to behave and what capabilities can operate in the terminal environment.

Conclusions and Future Work

In this short paper we have described the contributions to the MPEG standard mainly based on Java technology. A complex EPG application has successfully validated the requirements for the architecture to implement new service scenarios.

The contribution made by the SOMMIT project to MPEG-4 and DAVIC/DVB, jointly with the efforts provided by all the others companies involved in the standardization
groups, allowed the achievement of concrete results in multimedia standardization.

During this work we learnt that a lot of effort is still needed to reach a complete and more powerful multimedia framework. New tools and services will be developed and the existing ones re-targeted with a suitable standard that allows full interoperability among different environments (e.g.: broadcast vs. point-to-point) and products (developed by whatever industry).

Future developments of this work involve studies on applications and services that could benefit from the described terminal architecture. SOMMIT has taken into account other applications for its terminal, and has made plans to work further on the integration in a single environment of the HTML browser, MPEG-2 and MPEG-4.

Another important topic in this area is the study of the business models that are associated with this kind of enhancement content. In fact, while technology is becoming mature for any kind of 2D/3D applications, the revenue paths associated with this kind of applications are not well understood yet.

The present implementation is based on a PC platform that is indeed the most suitable environment for new developments: according to the SOMMIT workplan we have however started the porting of some of the new capabilities (namely MPEG-4) on a STB prototype. It is, in fact, the STB consumer equipment category that will ultimately provide the above applications to the users.

REFERENCES

ISO/IEC IS 13818-1 Information technology- Generic coding of moving pictures and associated audio information- Part 1: Systems
ETSI standard ETS 300 468 “Service Information”