The ASSET Architecture: Integrating Media Applications and Broadcasting Products in a Unified Model and Framework

Mathurin Body, Bernard Cousin, Paula Viana, Mario Cordeiro, Vitor Rodrigues, Damien Bommart, Guilia Ferrari, Massimo Strambini, Ingo Hoentsch, Tobias Marx, et al.

The ASSET Architecture: Integrating Media Applications and Broadcasting Products in a Unified Model and Framework

Mathurin Body, Bernard Cousin, INRIA, France
Paula Viana, Mario Cordeiro, Vitor Rodrigues, INESC Porto, Portugal
Damien Bommart, Compaq-HP, France
Giulia Ferrari, Massimo Strambini, SHS, Italy
Ingo Hoentsch, Tobias Marx, IRT, Gernamy
Walter Bernet, Edgar Müller, Dalet A.N.N., Germany
Serge Daulard, THOMSON, France
Bernard Algayres, Marc Laurentin, FPDI, France

Abstract

To design and develop an Architectural Solution for Services Enhancing digital Television is the goal of the ASSET project. This European funded project gathers eight partners (Compaq-HP Group, THOMSON, Dalet a.n.n, INESC Porto, INRIA, Institut fuer Rundfunktechnik, Front Porch Digital International, SHS Multimedia) to create a universal and unified system architecture that shall greatly improve interoperability in broadcasting environments. The implementation of this solution will indeed make integration of different proprietary systems and media applications much easier. This paper outlines the ASSET architecture, describes some key-concepts of the ASSET framework and presents the prototype currently under development.

I. Introduction

Digital systems for TV content creation offer facilities that cover the complete operational workflow, including acquisition, creation, editing, control, storage, broadcasting publishing and archiving of media assets.

Technical solutions available on the market are limited due to the lack of interoperability between equipment and applications. They are vertically integrated or proprietary and rely typically on a single manufacturer or system integrator. Hence they are not compliant with the broadcaster requirements. Application providers should be able to implement platform independent and device independent programs for increased portability.

The objective of the ASSET project [1] (IST-2001-37379 Architectural Solutions for Services Enhancing digital Television) is to identify solutions and define a software architecture that will overcome these limitations. The project identifies technologies, APIs, and software tools that enable both users and vendors to use/create media applications integrated in systems independent of device manufacturer, programming language and the underlying middleware platform.

The project approach takes advantage of open standards and emerging technologies (like MXF [2], standard data models for describing essences, XML [3] and distributed system technologies) for defining the ASSET architecture. The structural design of the ASSET solution wraps the standard software layered architecture into a middleware approach that provides:

- the abstraction of broadcast software and hardware devices as logical resources,
- generic, openly defined and extensible interfaces to control devices, data distribution and data flows,
- added value for system logic: optimized decisions to configure devices, convert/move data and handle workflows.

II. ASSET System Architecture

A. *Architecture overview*

The ASSET architecture is based upon a software framework – *the ASSET framework* – composed by a set of three standard interfaces and protocols for applications and products working together in an integrated environment. Applications communicate with the framework using the *ASSET Public API*, which allows them:

- to communicate with any other application,
- to access the public services provided by the framework,
- to use additional functionalities implemented by third party integrators as aggregated services.

Products from different manufacturers are integrated in the ASSET framework either natively (using an ASSET agent) or via an ASSET proxy. They are controlled and managed through the *Media ASSET Bus API*, based on XML service schema definitions. This API ensures an easy and seamless integration of devices and products in the framework.

An *ASSET Private API* is defined to access core services of the framework (common services) that enable the workability of this integrated environment. This API is only used internally in order to guarantee the overall system integrity.

Figure 1 illustrates the different components of the ASSET architecture.

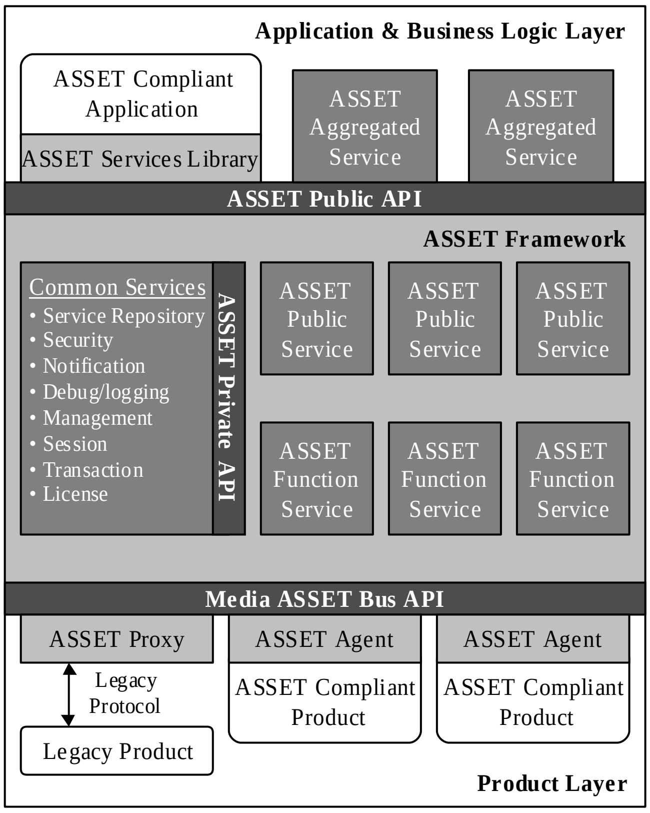

Fig. 1 – ASSET Architecture

Adding new applications or products to the framework simply requires the writing of a small software adapter to conform either to the ASSET Public API or to the Media ASSET Bus API.

B. *ASSET Architecture components*

The ASSET Architecture defines a number of concepts, components and functions that enable the implementation of an ASSET Compliant Framework:

The core of the ASSET Framework consists of three main components:

- The *ASSET Public Services* expose the mandatory services of the ASSET framework to the applications and the aggregated services through the ASSET Public API. These services provide a minimum set of multimedia functionality, sufficiently rich and extensible to not limit the system efficiency. They ensure the consistency and integrity of the system by handling the internal logic (e.g. access right, resource allocation, etc.) required for each operation.
- The *ASSET Common Services* provide implementation of key infrastructure requirements such as security, logging, notification, resource management, etc. This allows a uniform and single implementation of these services throughout the solution. They expose a private API that can be only used by ASSET Public Services.
- The *ASSET Function Services* provide an abstraction of functionalities (encoder, recorder, player, etc.) to the ASSET public services. They hide the specificities of the different interconnected products (e.g. for a public service, a VTR output and a Video Server output are considered as two system-wide logical output ports).

At the Application and Business Logic layer, three components are introduced:

- The *ASSET Compliant Applications* are the top level ASSET software components. They use the ASSET Public API to access services provided by the ASSET Framework and optionally, by ASSET aggregated services.
- The *ASSET Services Library* is a software component included in (or linked with) an application, which makes it compliant to the ASSET framework and gives it access to the ASSET public services.
- The *ASSET Aggregated Service*s implement additional business logic on top of public services (or even other aggregated services). They register in the framework as new services available for ASSET compliant applications (e.g. complex workflows may be specified as aggregated services and then, be available for other connected applications or aggregated services).

At the product layer, we call *product* a manageable hardware or software component that implements one or several common functions (e.g. most of the Video Server products implement a Recorder function, a Player function and a Storage function) and logical components (logical ports, repositories, etc.). Two ways exist to make a product compatible with the ASSET framework:

- An *ASSET Compliant Product* is a product that is managed by the framework through a built-in *ASSET Agent*.
- A *Legacy Product* is a product that has not (or cannot have) a built-in ASSET agent. The ASSET framework can nonetheless managed such a product through an external software module called an *ASSET Proxy* (e.g. a VTR cannot include a built-in ASSET agent and has to be connected through an ASSET Proxy).

Products communicate with the framework using the API provided by the Media ASSET Bus, which is introduced in the next subsection.

C. *Media ASSET Bus*

Most of the ASSET framework is built above a software bus called the Media ASSET Bus or MAB.

The goal of the MAB is to provide support for the integration of the widest range of products within the ASSET framework, independently of the underlying operating system environments and protocols.

The MAB defines a set of standard interfaces and synchronization process, which ensure a seamless interoperability between ASSET components and products. These interfaces allow any third party media application or product to integrate into the Media ASSET Bus by developing a simple software adapter in an agent or a proxy. A MAB Software Development Kit (SDK) is provided in order to facilitate this task. It gives a uniform way of connecting devices, registering services and exchanging messages within the ASSET framework.

The integrated environment that the MAB offers is based on a Transport Abstraction Layer (TAL), which takes care of message exchanges. Messages have XML format that ensure flexibility and adaptability of the ASSET solution.

The concept of the Media ASSET Bus is illustrated in Figure 2. Different ASSET components and products are interconnected via software adapters on top of the MAB SDK.

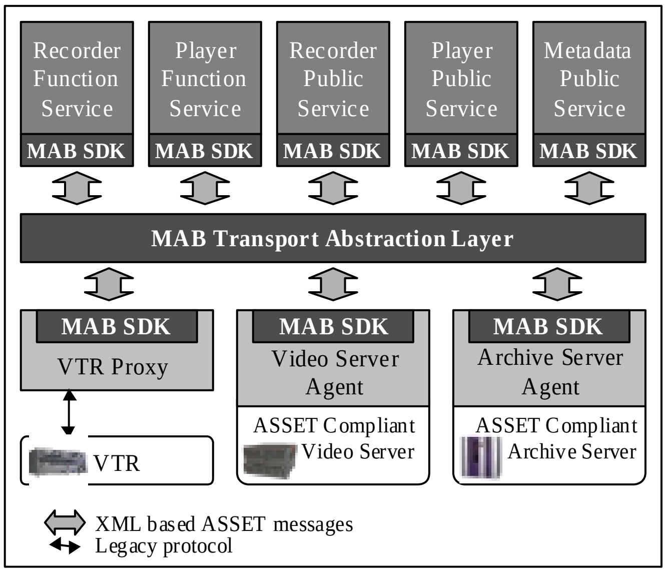

Fig. 2 – Media ASSET Bus

III. ASSET Data and Metadata Models

Within the ASSET framework, three data models were defined:

- The *Structural and Control data model,* which embraces all data dedicated to the system description (services, functions, security and access rights, etc.).
- A *Descriptive metadata model,* which provides the description of essences available within the framework.
- A *Content Management metadata model,* which describes the location of the media assets and their possible relationships.

A. *Structural and Control data model*

The information stored in the Structural and Control data model is distributed among several common services. The private API provided by the common services allows the creation and update of the data describing the past, present and eventually future state of the framework. For example, the session common service manages the information regarding the past and still active sessions, created by applications to access the framework. The Structural and Control data model is strictly designed to the ASSET framework and embraces the global description of the system and resources.

B. *Descriptive metadata model*

Almost every broadcaster or production company may have its own workflow and its own proprietary set of metadata describing essences, workflows and processes. Hence, it would not be possible to introduce a descriptive data model coping with these different requirements. As a result, we distinguished technical metadata (e.g. compression type, image size, etc.) from the other descriptive metadata.

Some technical metadata, which are required for the ASSET framework, are handled by the technical metadata common service. The other descriptive metadata are handle by descriptive metadata public and function services, based on a default model, which is not compulsory but may guide developer implementation. DMS-1 [4][5] is proposed as the default descriptive metadata model.

C. *Content Management metadata model*

The goal of the Content Management metadata model is to address the need of describing the location and the relationships between files and Media assets. For example, a media asset may have several representations (e.g. same clip with different compression types) each having different structures (e.g. a Media asset may be composed of one physical file or structured as a collection of audio, video and metadata files).

IV. Content Exchange

A. *Content Exchange Overview*

Associations of public broadcasters like the EBU-SMPTE Task Force [6] have highlighted the requirements to exchange media content in digital television environments, using non-proprietary formats. Different aspects for interoperability were identified. Some of them are:

- multiple users accessing simultaneously and independently the same content;
- various and adaptable speed transfer across scalable network areas;
- common container for data and metadata organized in data models;
- simple, direct and eventually partial access to the content through standardized network protocols and interfaces;
- unified formats for manipulating, managing, storing, sharing and distributing essences and metadata.

Emerging content exchange technologies shall satisfy these requirements.

Recently, the Pro-MPEG Forum [7] and the Advanced Authoring Format (AAF) Association [8] have developed an open standard that ensures the interoperability among the different vendor systems involved in production environments.

B. *MXF – Material eXchange Format*

The Material eXchange Format [2][9] is an Open Standard file format for the interchange of audio-visual materials and associated metadata. This format is compression scheme independent and therefore facilitates the integration of systems using MPEG, DV or other compression strategies, even yet unspecified. This format has actually been designed to address the interoperability requirements and has naturally been applied in the ASSET project.

MXF is used for file interchanges within the ASSET framework as well as for the export/import of both essences and metadata, to/from products or applications external to an ASSET framework.

V. Demonstration Platform

A simple demonstration platform is currently under development for IBC 2003. Its goal is to assess and validate the quality of the services and concepts introduced by the ASSET solution. This demonstrator will be a deployment example of the ASSET solution in a common example of TV production, where devices and applications are integrated in a unified system. The planned demonstrator does not focus on a particular workflow, but on how functionalities provided by the framework are supported and how interoperability and connectivity is achieved.

Figure 3 illustrates the demonstration platform currently specified for IBC 2003.

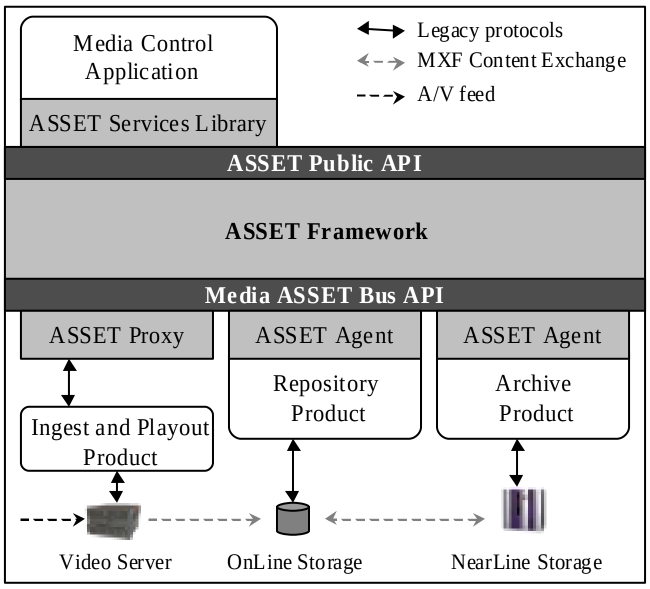

Fig. 3 – ASSET demonstration scenario

The demonstrator uses a client and several products of different manufacturers to emulate simple workflows of a news platform. The client is a media control application used to control ingest, archive, delete or play media content between connected devices. The products are a video server, which allows ingest and playout of clips, an online storage system (repository) and an archive server (nearline storage). Technical descriptive metadata and content management metadata are stored internally using ASSET common services.

In the demonstration platform, contents can be ingested into the framework via the A/V feed of a video server. Media asset may then be exchanged (e.g. for archiving) using the MXF file format. Descriptive metadata are extracted during the ingest phase and can be retrieved by the system through the ASSET agent of the online storage system. They are actually not permanently stored in a Media Asset Management system (as they should), in order to simplify the demonstration platform.

Using the framework within the above scenario provides a straightforward integration and collaboration of various proprietary applications and devices. The exchange of asset media and metadata becomes transparent through the aid of MXF.

The ASSET demonstrator may also serve as a reference system where application providers, administrators, and integrators are able to test and verify the implemented functionality against the specifications and the requirements.

VI. Conclusion

This paper gives an overview of the project approach and describes the work under development in the IST ASSET project. The software architecture is outlined and its components presented. APIs will be made available to manufacturers and applications providers, so that interconnection between equipment and media applications is greatly improved in digital TV environments. The partners have already defined the software architecture, concepts and demonstration scenario and are now working towards the development of the prototype that shall demonstrate the effectiveness of the ASSET solution.

VII. References

[1] ASSET web site – http://www.ist-asset.com
[2] Pro-MPEG Forum, "Material eXchange Format (MXF)", http://www.pro-mpeg.org/mxf.htm, October 16, 2002.
[3] Extensible Markup Language (XML) 1.0 (Second Edition). W3C Recommendation. World Wide Consortium, http://www.w3c.org/TR/REC-xml, October 6, 2000.
[4] Pro-MPEG Forum, "A guide to MXF Descriptive Metadata Schemes and their application", August 2002.
[5] Pro-MPEG Forum, "Material eXchange Format (MXF) – Descriptive Metadata Scheme (DMS-1)", October 2002
[6] H. Hoffman, "Networks and Transfer Protocols", EBU Technical Review, Autumn 1998.
[7] Pro-MPEG Forum – http://www.pro-mpeg.org
[8] AAF Association – http://www.aafassociation.org
[9] Bruce Devlin, "MXF – The Material eXchange Format", EBU Technical Review, July 2002.