



STREP

FP6-2005-IST-61-045410

MOBISERVE

**New mobile services at big events using DVB-H
broadcast and wireless network**

Deliverable:

**MOBISERVE rich-media middleware specification for the Olympic trials
D4.2**

Due date of deliverable: M6
Actual submission date: 27/04/2007

Start date of Project: 01 September 2006

Duration: 24 months

Responsible WP: Streamazzo

Revision:

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Service)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (excluding the Commission Services)	

0 DOCUMENT INFO

0.1 Author

Author	Company	E-mail
Pierre Plevén	STZ	pierre.pleven@streamezzo.org
Philipp Steckel	TU-BS	steckel@ifn.ing.tu-bs.de
Marius Spika	TU-BS	spika@ifn.ing.tu-bs.de
Jianzhong LI	FTB	ljianzhong.ext@orange-ftgroup.com
Joep van Gassel	PRLE	joep.van.gassel@philips.com

0.2 Documents history

Document version #	Date	Change
V0.1	17/04/07	Starting template, split of combined deliverable
V0.2.	20/04/07	Skeleton generation, MSCs added (TU-BS)
V0.3	25/04/07	Introduction added (STZ, TU-BS)
V0.5	27/04/07	Rich Media description added (STZ)
V0.6	26/04/07	Advanced architecture description added (TU-BS)
V0.8	27/04/07	ISAP description added (FTB)
V1.0	27/04/07	Final revision (TU-BS)

0.3 Document data

Keywords	
Editor Address data	Name: Pierre Plevén Partner: Streamezzo Address: 21 BD Victor Hugo 75016 Paris France E-mail: pierre.pleven@streamezzo.org
Delivery date	

0.4 Distribution list

Date	Issue	E-mailer

Table of Contents

0	<u>DOCUMENT INFO</u>	2
0.1	AUTHOR	2
0.2	DOCUMENTS HISTORY	2
0.3	DOCUMENT DATA	2
0.4	DISTRIBUTION LIST	2
1	<u>INTRODUCTION</u>	4
2	<u>COMPARISON AND TECHNOLOGY DECISION FOR RICH MEDIA</u>	6
2.1	DEFINITIONS	6
2.2	A VISION ON RICH MEDIA SERVICES	6
2.3	KEY FEATURES THAT MAKE A DIFFERENCE WITH LASER BASED SOLUTIONS.	8
2.4	RECENT EVOLUTIONS AND LASER/DIMS CONVERGENCE	9
2.5	TECHNOLOGY DECISION	12
3	<u>APPLICATION LAYER DESCRIPTION</u>	13
3.1	OLYMPIC ARCHITECTURE	13
3.1.1	XRM APPLICATION	13
3.1.2	RICH MEDIA ENGINE	13
3.1.3	DATA MODEL PRESENTER	15
3.1.4	IP DATACAST ESG PRESENTER	16
3.1.5	ISAP INTERACTIVE SERVICE MANAGEMENT	16
3.1.6	ISAP SYNCHRONIZATION	17
3.2	ADVANCED ARCHITECTURE	20
3.2.1	JAVA ADAPTER	21
3.2.2	APPLICATION TYPES	21
4	<u>REFERENCES</u>	25

1 Introduction

This document highlights one of the novel aspects of the MOBISERVE architecture, in particular in the area of Rich Media technologies. In this document we shall analyze technological backgrounds which lead to decisions for particular solutions that work package 4 will implement. In addition, the interfaces and specification of the high-level (application) middleware stack is introduced.

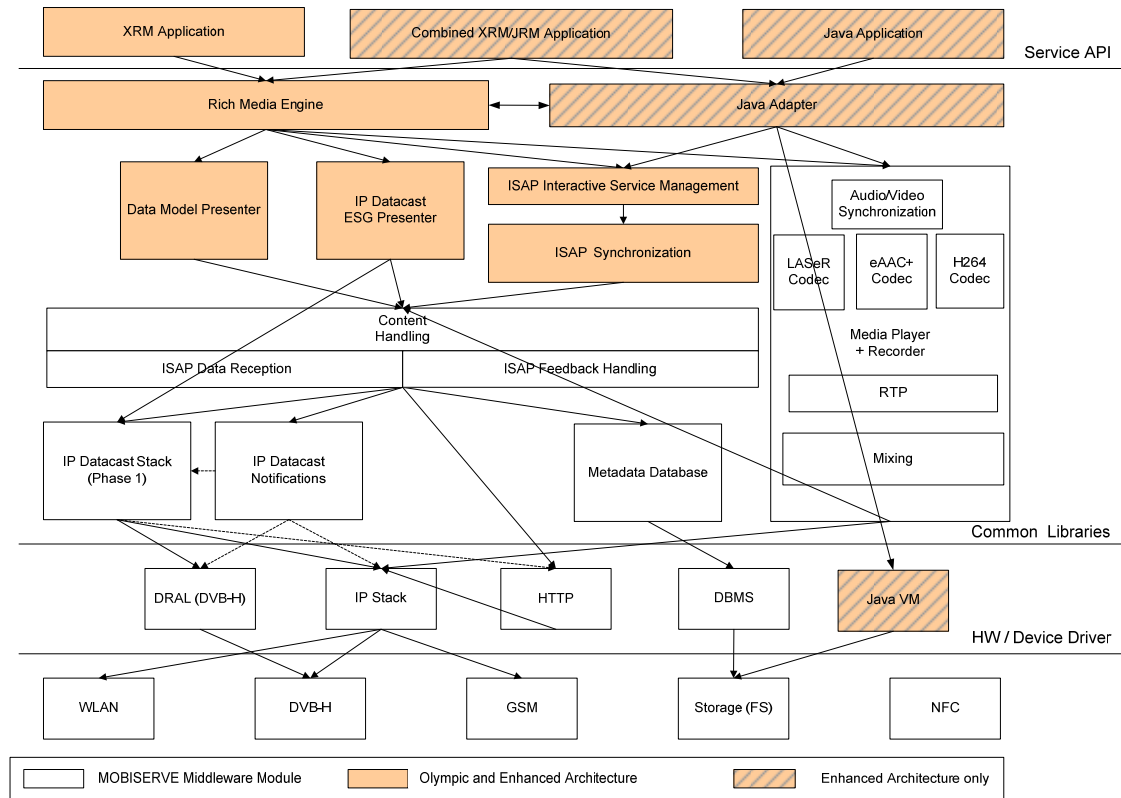


Figure 1 MOBISERVE terminal software architecture

Figure 1 depicts the terminal software architecture which was introduced in D4.1 [1]. This document focuses on the high-level application middleware which includes the Rich Media-related components and the respective support modules (i.e. Model Presenters, Synchronization logic). In the figure, the respective modules are marked in red. During the development of the terminal architecture, a two-profile approach has been agreed on and introduced in D4.1. The concept is to decouple the terminal software developments for the Olympic trials and research-oriented demonstrations on exhibitions and conventions. Accordingly, the “Olympic Architecture” is envisaged to be used for the trials during the Olympic games in Beijing whereas the “Advanced Architecture” is used during research-oriented dissemination activities. In the figure, the red non-hatched software modules are included in both architectures whereas the red hatched modules are available in the Enhanced Architecture only.

In both architectures the rich Media Engine is one of the important components. Therefore, the first chapter of this document discusses the positioning and the reasons for the choice of the MPEG4 Part 20 Laser implementation.

The second part of the document gives a detailed overview about the upper layer Rich Media-based Application Middleware. For the sake of comprehensiveness, Message

Sequence Chart diagrams depict the high-level communication relationships of the particular modules.

2 Comparison and technology decision for Rich Media

2.1 Definitions

A Rich Media service is a dynamic, interactive collection of multimedia data such as audio, video, graphics and text. It ranges from a movie enriched with vector graphics overlays and interactivity (possibly enhanced with closed captions), to complex multi-step services with fluid interaction and different media types at each step.

The demand for such Rich Media service is increasing at a high pace, spurred by the development of the next generation mobile infrastructure and the generalization of TV content to new environments. Despite long lasting deployments and significant investments that have been made by the various actors of their respective industries, mobile and more generally embedded interactive services (e.g. mobile internet, interactive mobile TV) have failed to reach the masses. There are certainly a number of reasons for this failure, some of them being conjectural (e.g. economical) or structural (e.g. lack of compelling business models). Still, from a pure user experience point of view, the technologies in place also suffer from major drawbacks:

XHTML-like approaches (e.g. WAP, I-Mode, XHTML ...), although successful on the Internet, have shown their limits to provide a simple, efficient and deterministic user experience on non-PC heterogeneous devices.

The transposition of PC technologies to constrained devices and networks without the requested adaptation to take into account the characteristics of these environments (e.g. network latency, bandwidth, devices capabilities, screen sizes ...) lead to very low browsing capacity and poorly usable services.

In addition, the current technical approaches did not integrate from their original design the integration of audio and video media. Although they then attempted to integrate multimedia when it was recognized as a key driver of growth of services, only poor results have been achieved so far. Even though technology is definitely not a sufficient condition of economical success, we think it is a pre-condition to the pervasive development of Rich Media services on non-PC devices:

Using Rich Media services on embedded devices is more challenging than on a PC where various interfaces are available and homogeneously implemented (e.g. mouse, keyboard, ...), and for which ergonomic concepts have been tested and validated for years. On the move, in situation where it is not always easy to interact, where time is limited, users expect to be one click away from the information they need. End-users have been accustomed to quality interfaces on the Web. To be successful on the embedded domain, service interfaces need to leverage the "on-line experience". Finally, since these services are expected to generate revenues, the users also expect a decent level of quality, efficiency and readability as a pre-condition to pay.

2.2 A Vision on Rich Media Services

Several technologies are competing to achieve the vision described in the previous section, among which Flash is the first to analyze.

Indeed, Flash is very successful on the PC. It is the current de-facto standard for distributing Rich Media content on the Internet but suffers from serious drawbacks to address efficiently other industries requirements:

Flash is not an open standard. This is a critical issue to get massive industry support, especially on mobiles. Content creators, services operators and device manufacturers would be tied to Macromedia for the creation, distribution and playback of Rich Media

content. Knowing that once deployed, a media infrastructure is hard to make evolve, the trend is to avoid proprietary solutions and promote open standards.

Flash is a technology designed for the PC. As such, it is not suitable for the mobile environment as demonstrated by the current development of the mobile version of Flash, called Flash Lite. To match the double constraint of being compatible with its existing PC format and to fit constrained devices requirements, Macromedia had to compromise a lot on technology. The first version of Flash Lite is a downgraded version of what is available on PC. In addition, the problem of having a single vendor and a proprietary format remains.

Following this analysis, an important requirement for a Rich Media solution for mobiles is to be open and allow easy conversion from the many existing types of Flash and other proprietary content into this new standard. Two standardization groups have tried to specify standards which would satisfy this requirement: MPEG and W3C.

MPEG-4 BIFS is the first attempt of MPEG in the field of composition coding. It features innovative tools that allow the creation of multimedia content mixing 2D and 3D graphics, introduces the notion of incremental updates of the scene, enabling streaming of long running scenes, and insures a tight synchronization between the different audiovisual elements of the scene.

We attempted at profiling MPEG-4 BIFS to create a small enough subset to be used on mobile phones, to no avail. The inherent content and binary encoding structure makes it inappropriate for the mobile. Instead of compromising on the technology performances, MPEG reached the conclusion that an optimum between feature richness/compression efficiency and device constraints needed to be found and decided to create a new standard for Rich Media for constrained devices.

As an alternative to the Flash proprietary solution, the W3C also made attempts to define languages for creating Rich Media content. The SMIL and SVG standards are among these languages. Both of them are getting traction in the mobile industry, where SMIL and SVG mobile profiles have been adopted by the 3GPP and OMA consortia.

However, both are XML languages, relying on the HTML model for content consumption: download-and-play, or progressive download and rendering. However, the streaming of SMIL or SVG content is not specified, making these models inappropriate for fast, dynamic and interactive and interoperable content. A strong requirement for the new standard for Rich Media content for mobiles is to leverage the adoption of SVG by providing means to extend its consumption scenarios to streaming and broadcast.

Creating Rich Media content services for the mobile is not only about composition coding. An important aspect in the success of a mobile service is the reactivity and fluidity of the user experience. Such characteristic is achieved through efficient delivery mechanisms. However, efficiency is difficult to achieve when distributing Rich Media content made of individual audio, video, and image content. Separate delivery of all these media streams to a mobile incurs a high latency unless an efficient aggregation mechanism is used: high-latency networks attach a specific penalty to multi-media content when consumed in download-and-play mode, because the waiting time of the content is the sum of the waiting time of each media requested separately. Instead, getting all streams in one single package would reduce waiting time to one single request delay.

The requirements for such aggregation mechanisms are simplicity of implementation and efficiency.

A natural candidate for this task is the ISO Base Media File Format because it is very popular and already adopted by the mobile industry. However, this file format was designed for storage of large amount of media data, easy editing or streaming operations. It is not efficient for the storage of small amounts of timed media data. A simple to implement yet efficient aggregation format for mobile is needed to complement the ISO Base File Format.

2.3 Key Features that Make a Difference with LAsER based solutions.

The LAsER standard has been designed to satisfy the end-users expectation listed above. Four key features can be highlighted that makes a real difference with existing technology:

1. Graphic Animations, Audio, Video and Text are packaged and streamed altogether.

Contrary to existing technologies on mobile that are mostly aggregation of various components, not necessarily well integrated together (e.g. XHTML + SMIL + SVG + CSS + EcmaScript + ...), LAsER grounds its design from what made the success of

Macromedia Flash on the Web: a single, well defined and deterministic component that integrates all the media. This integration ensures both the richness and quality of the end user experience.

2. Full screen and interactivity with all streams.

With the use of vector graphic technology, content can easily be made to fit the screen size. This feature enables to provide an optimal content display although the screen resolution is highly varying. In addition, virtually all pixels can be used as elements of the user interface. This allows the design of rich and user-friendly interfaces, similar to what people used to have when interacting with their devices.

3. Real-time content delivery.

LAsER has been designed for an efficient delivery over constrained networks. More specifically, Rich Media LAsER content can be delivered into packaged pieces, allowing display as soon one piece is received (as opposed to a download and play mechanism). This concept of "streaming", already into place for audio and video data, has been generalized to scene description and Rich Media. As such, services can be designed such that there is always some information of interest on the screen.

4. Low bandwidth.

Last but not least, LAsER has been designed to deliver Rich Media Service starting from 10 Kb/s. The key technology used here is vector graphics compression and dynamic updates of the scene. This feature enable to drastically limit the waiting time of the end users as opposed to a standard Web-like approach where the complete page is re-sent even though only small changes had been made. Needed for low bitrate networks such as GPRS, this functionality is also useful on higher bit rate network where Rich Media services can be sent at low rate, therefore preserving bandwidth to improve audio and video quality.

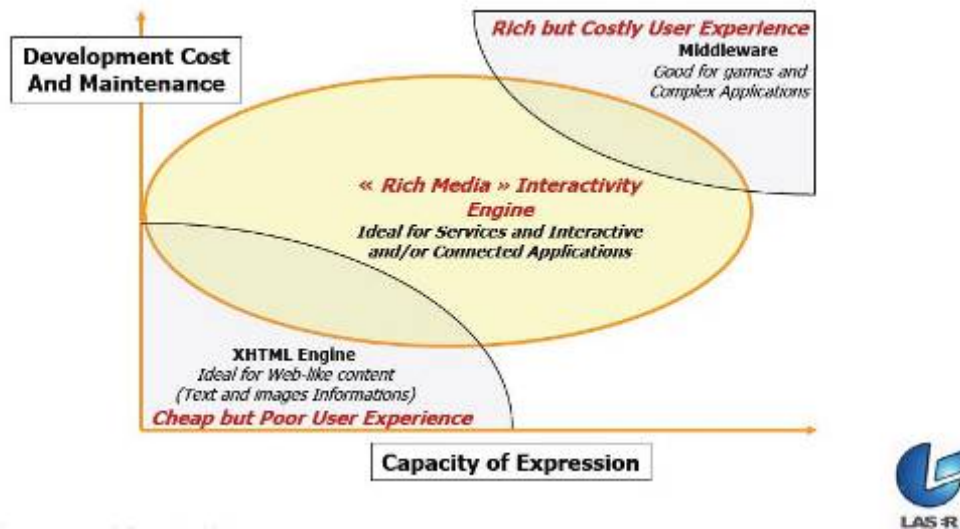


Figure 2 Cost vs. Expression Capacity

2.4 Recent evolutions and Laser/Dims convergence

One of the key components for a non linear and active media experience relates to the presentation navigation and interactivity that delivers to the user access to “Rich Media”. In the Rich Media definition here we include all techniques allowing a strong interactivity between the user and the desired contents, These contents includes texts, graphics, animations, still images, audio and video and the application logic for the interactivity needs. All these components, taken as a whole can be described as Rich Media. The presentation engine is the software component in the terminal middleware that manages the relationship between the user and the visible part of the display (frame buffer)

During the MMC Workshop in Berlin on October 20, 2006, Alfred Baier and Martin Richartz from Vodafone summarized the different types of interactivity solutions as given in Figure 3 distinguishing between

- Generic browsers
- Rich-Media streamed solutions
- Downloadable Rich Media applications (“Java”)

Generic browser-based solutions allow for low-cost applications whereas downloadable applications potentially provide a larger set of functionality.

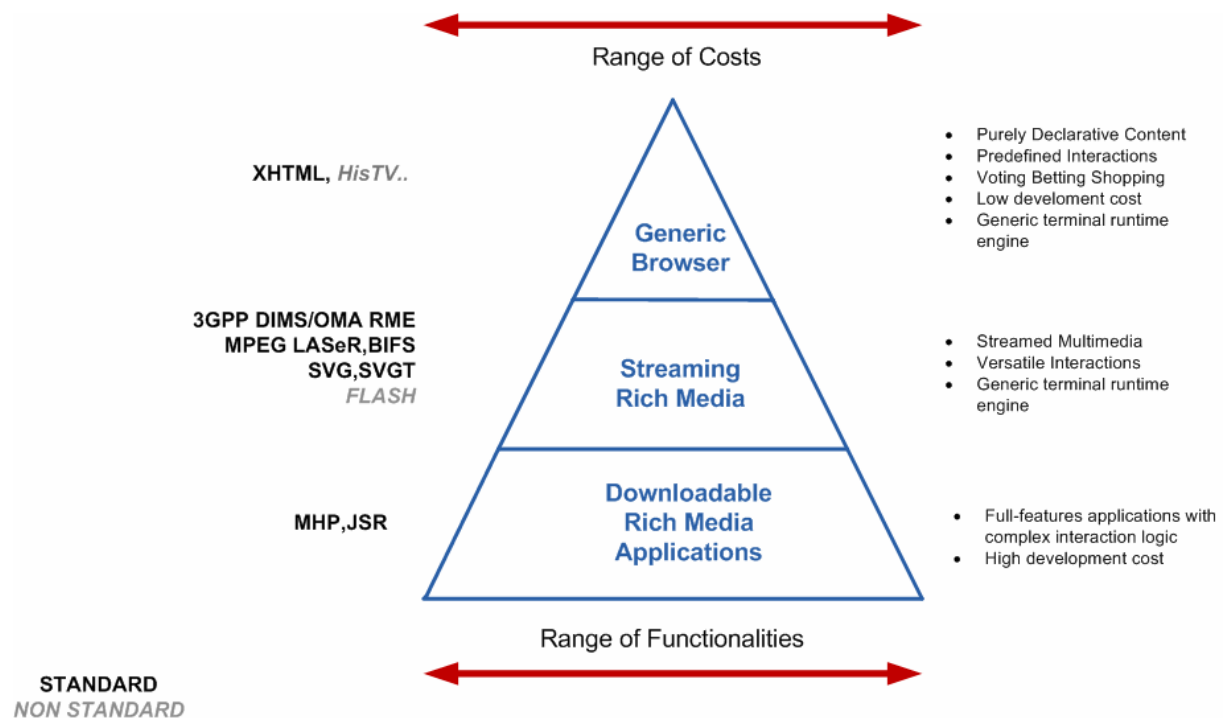


Figure 3 Three types of interactivity solutions

Currently different declarative formats exist already that provide some sort of abstraction of the user interface. Examples are UIML, XAML (Microsoft), MXML (ADOBE), XUL, ...

As of today solutions for Rich Media standard browser are few to cover both broadcast and unicast environments.

ISO/IEC 14496-20 defines LAsER (Lightweight Application Scene Representation) and SAF (Simple Aggregation Format) designed for devices needing low footprint rich media browsers. In a nutshell, LAsER is a binary encoding of SVG Tiny 1.2 with dynamic updates added, and SAF is a very simple configuration of the MPEG-4 Sync Layer, allowing aggregation and synchronization of elementary streams, secure streaming on HTTP, and streaming with RTP through the mapping defines in RFC 3640.

The requirements for LAsER and SAF include footprint, code size and performance in addition to the usual MPEG criteria of compression, so as to cater with the specific needs of constrained devices such as mobile phones. LAsER and SAF allow rich-media mobile services to be designed and streamed to a majority of today's phones with the current network capabilities and protocols. The specification will be completed as soon as SVGT1.2 goes final.

LAsER and SAF have been proposed as work items to OMA and 3GPP. A 3GPP work item called DIMS based on a subset of LAsER is currently finalized and will be adopted before summer 2007. OMA has adopted a rich-media environment (RME) work item which is equally based on a subset of LAsER, and conceived as a superset of DIMS. It will be ready by the end of 2007.

Today Rich Media based services are widely deployed in the mobile 2.5/EDGE/3G environment, with customers like SFR in France, H3G in Israel, T-Mobile in Germany, Starhub in Singapore and M-Stars in Indonesia.

First commercial implementations will be carried out in the first half of 2007 in the domain of broadcasting to mobile terminals (DVB-H, T-DMB, MediaFLO); and several research projects (FP6 Mobiserve and National French project Mobimages) are trialling broadcast / connected network cooperation for rich media delivery based on LAsER.

Another research project (National French project InterLight) is experimenting LAsEr in the DVB-T environment associated to MPEG-2 as well as MPEG-4 (SD&HD) audiovisual content.

The standard 3GPP DIMS (Dynamic interactive Multi-media Scene) and OMA RME (Rich-Media Environment) will be compatible and be based on the same technological core. This core will be adapted in the context of 3GPP for environments like MBMS, PSS and MMS, and also in the context of OMA with environments like OMA-BCAST, OMA-DCD, etcetera.

It is important to notice that this technological rich-media core is common to both standards 3GPP/OMA and is portable on any broadcast bearer (MPEG-2, TS, DVB, MediaFLO, DMB...)

In DVB activities of CBMS phase 2.0 are about to start with extensive discussion with OMA for potential alignment

Progress beyond the state-of-the-art of LAsEr solution for the Rich Media

The main advance will be to bring the benefits of streamable Rich media solutions to the proposed architecture. Based on scene representation principles, the main progress beyond the state of the art will be a seamless presentation, navigation and interactivity software module, allowing to optimise browsing, streaming, progressive downloads and fully downloaded applications .

Implementing the “Rich Media” runtime engine as one of the components of MOBISERVE Middleware, will create the necessary abstraction layer to bring to the content creator and end user the consistency they need across heterogeneous networks and terminals.

MPEG-LAsEr and DIMS/RME:

DIMS/RME refers in a normative way to the technologies defined in MPEG-LAsEr. The figure below illustrates the compatibility and the complementarities of these various standards.

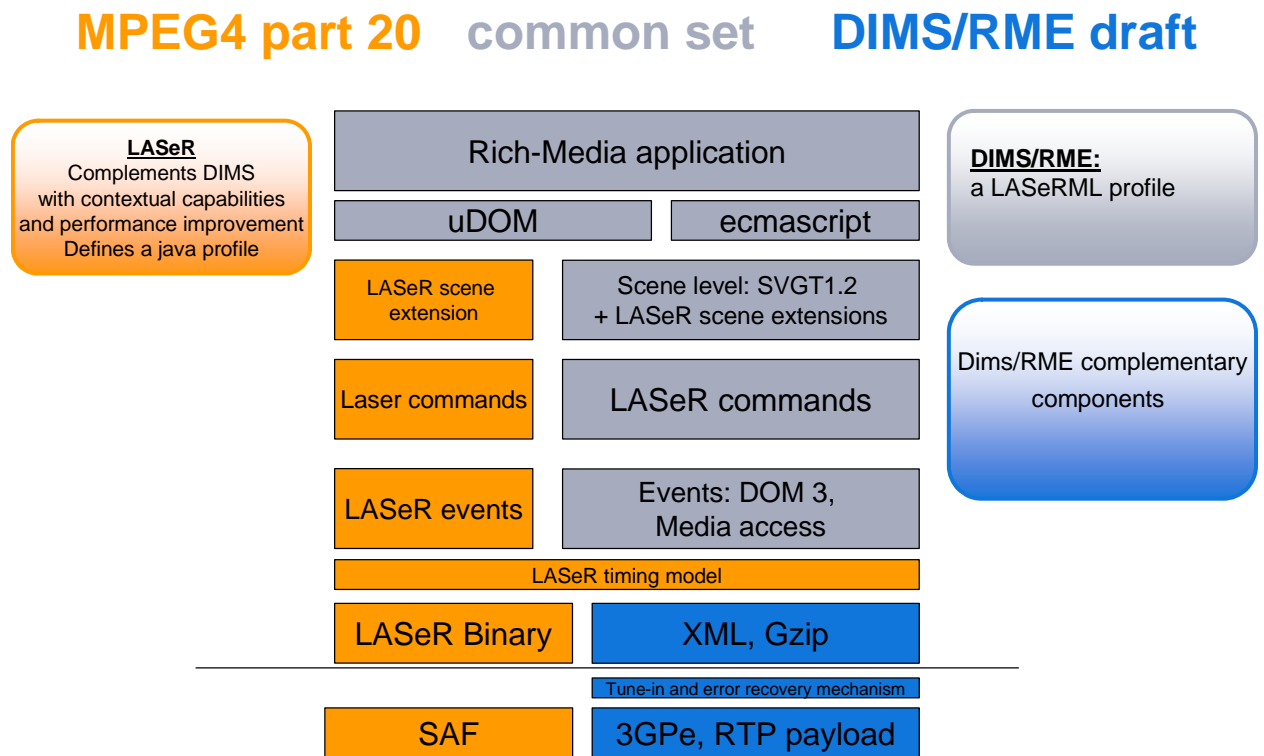


Figure 4 Comparison and complementarities between the two initiatives

The complementary elements in LAsER are mainly dedicated to the optimization of the resources. (Compression, detection of the changes in the environment of the user: level of battery, band-busy, validated data...). A MPEG-LAsER profile relating to DIMS is under study.

2.5 Technology decision

In addition to the foregoing analysis of rich media technologies the figure below shows a comparison of various leading rich media technologies. In consequence of the results work package 4 partners decided to choose LAsER technology to support the developments in MOBISERVE.

Key Differences	Laser	BIFS Core2D	FlashLite	SVG Tiny 1.2
Audio and Video support <i>Tight integration</i>	😊	😊	😞	😊
Device input sensitivity <i>Fine tuning to device characteristics</i>	😊	😊	😞	😊
Readability (fonts) <i>High quality small fonts</i>	😊	😊	😞	😞
Progressive download <i>Immediate, continuous, dynamic context eg televoting</i>	😊	😊	😊	😞
Dynamic Updates <i>Compressed, Package in a single file,</i>	😊	😊	😊	😞
Synchronisation (frame accurate) <i>Rich-media, audio, video...</i>	😊	😊	😞	😞
Interactivity <i>eg remote control for mobile TV</i>	😊	😊	😊	😊
Optimized in size <i>Optimized data traffic over the air</i>	😊	😞	😊	😞
Client-server data management <i>e.g. Continuous refresh of data with no memory saturation</i>	😊	😊	😞	😞
Decoder Footprint	😊	😞	😊	😞
Integration within infrastructure <i>e.g. within XHTML browser, uDom,</i>	😊	😞	😊	😊
CPU resources needed	😊	😞	😊	😞
Content development tools <i>Development tools, leverage current tools</i>	😊	😞	😊	😊

Figure 5 Comparison matrix of various rich media technologies

3 Application layer description

3.1 Olympic architecture

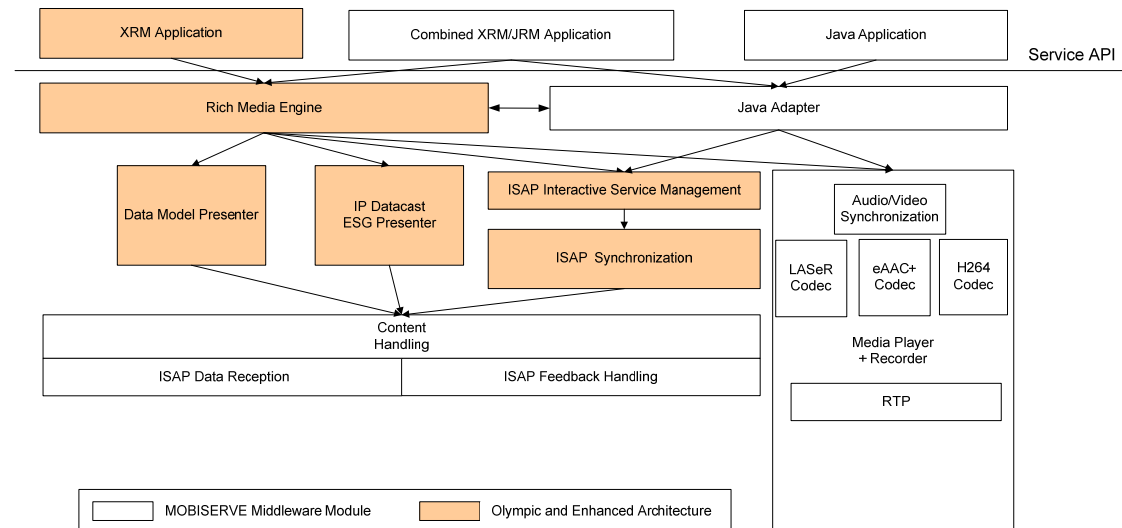


Figure 6 MOBISERVE application middleware stack

3.1.1 XRM Application

Description

For each service the consortium will develop an application which corresponds to the specifications in terms of drawing of screen, sequence of the screens and more generally in the rules of ergonomics specified in WP2

If these applications are developed in Rich Media, the author has a software tool of design and simulation (the Studio), which runs on a workstation of the type PC.

The application thus generated will be adapted to the target terminal according to its particular characteristics (size of the screen, physical ergonomics (function keys or not), and obviously embarked OS). This adaptation is facilitated by the presence of the RME (Rich Media Engine) which plays the part of a “middleware” specific to the “Rich Media” applications

Interfaces

The software component “application” needs the RME (Rich Media Engine) to function on the terminal, and can be also brought to be interfaced with AV player

3.1.2 Rich Media Engine

Description

The RME is a software component which ensures graphic rendering of the presentation layer of the multi-media application. This engine takes into account the graphic, audio and video elements, the fixed images or animations, and the texts with their police to compose

the “scene” seen by the user. This one will be able to interact on the “scene” in using the interaction mechanism proposed on the the terminal (keys, joystick, touch screen, stylus). The scene can be static or dynamic with the temporal dimension envisaged by the author. The scenes can be transferred in the form of “streams” or locally be downloaded.

Interfaces

The Interfaces of the software component “Rich Media Engine” are:

1. **The applicative layer** where the RME, according to received information will collect the contents and will take into account the actions of the user:

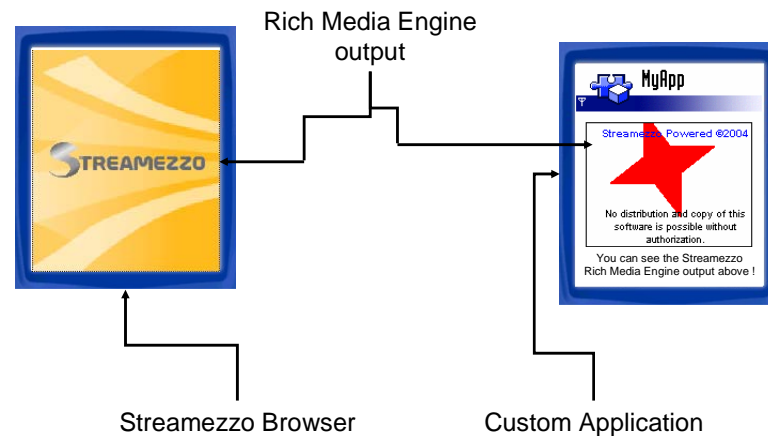


Figure 7 Rich Media control flow

2. **Audio and video codecs:** the composition of the scene requires that the RME off has access to the video sources within a buffer “screen” the RME will be responsible to compose the various Graphic Text and Audiovisual objects and to generate the images on the screen, a precise synchronization being implemented The composition includes the Space-Time positioning of the audio visual content

- The “Clipping” of the video, (Trimming)
- The scaling and the rotation of the video

The RME is available for the majority of OS and of the platforms, this makes the development of the services easily portable on the referred platforms (Symbian, Linux, Microsoft, JAVA (with limitations specific to the virtual machine such as access to the screen buffer for alpha mixing of the various objects components.)

3.1.4 IP Datacast ESG Presenter

This module interprets data not from the IPDC stack but from the Content Handler ESG data format (the format should abstract away from the exact ESG/EPG format used in different channels).

Inside the project the presenter will be developed as a Middleware agnostic DVB-H component

The IPDC data is organized in 7 data table

- Service Bundle
- Service
- Schedule event
- Acquisition
- Content
- Purchase
- Purchase Channel

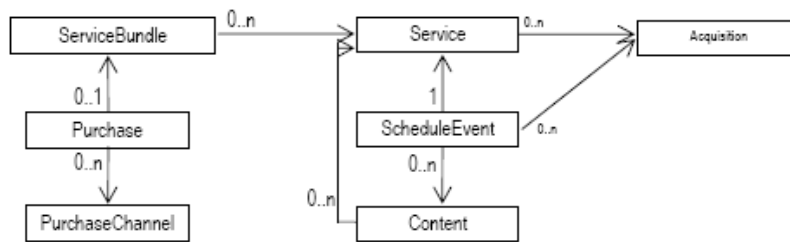


Figure 10 IP Datacast ESG structure

In the transformations which are necessary to interface with the Rich Media Engine the relationship reads as follows

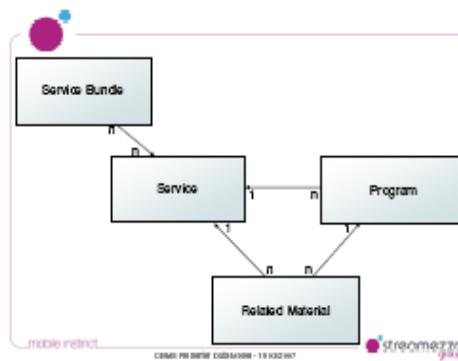


Figure 11 RME IP Datacast ESG instantiation

The detailed interfaces will be done at API level.

3.1.5 ISAP Interactive Service Management

ISAP Synchronization component is responsible for handling the timing information related both to received data and feedback requests.

The raw timing information related to the received data is received with it from the ISAP Data Reception component (Content handling module). After handling, this timing

information is sent to ISAP Interactive Service Management component in order to let the latter handle the priority of the related data.

Concerning the feedback requests, ISAP Synchronization component checks first the time validity of the feedback (for instance, as a kind of feedback, a vote may have a deadline over passing it implies the invalidity of the vote) and then send it back to the ISAP Feedback Handling component together with the appropriate timing information.

3.1.6 ISAP Synchronization

ISAP Interactive Service Management component is the central component for implementing interactive service applications based on the information received (received data) along with the A/V program (application session). Its main functions are as follows:

- Receiving the data from ISAP synchronization component with the timing information, namely the starting time, the ending time and the valid time;
- Providing the Rich Media Engine with the received data, the type of data, the kind of data and most importantly the status (new, ongoing, invalid, invalidated);
- Receiving from the Rich Media Engine the feedback request
- Checking the validity of the feedback request
- Sending the feedback to ISAP synchronization component
- Maintaining the status for the purpose the related applications and for allowing the back and forward between application sessions.
- Handling the priority of the data.

Bootstrap ESG use case

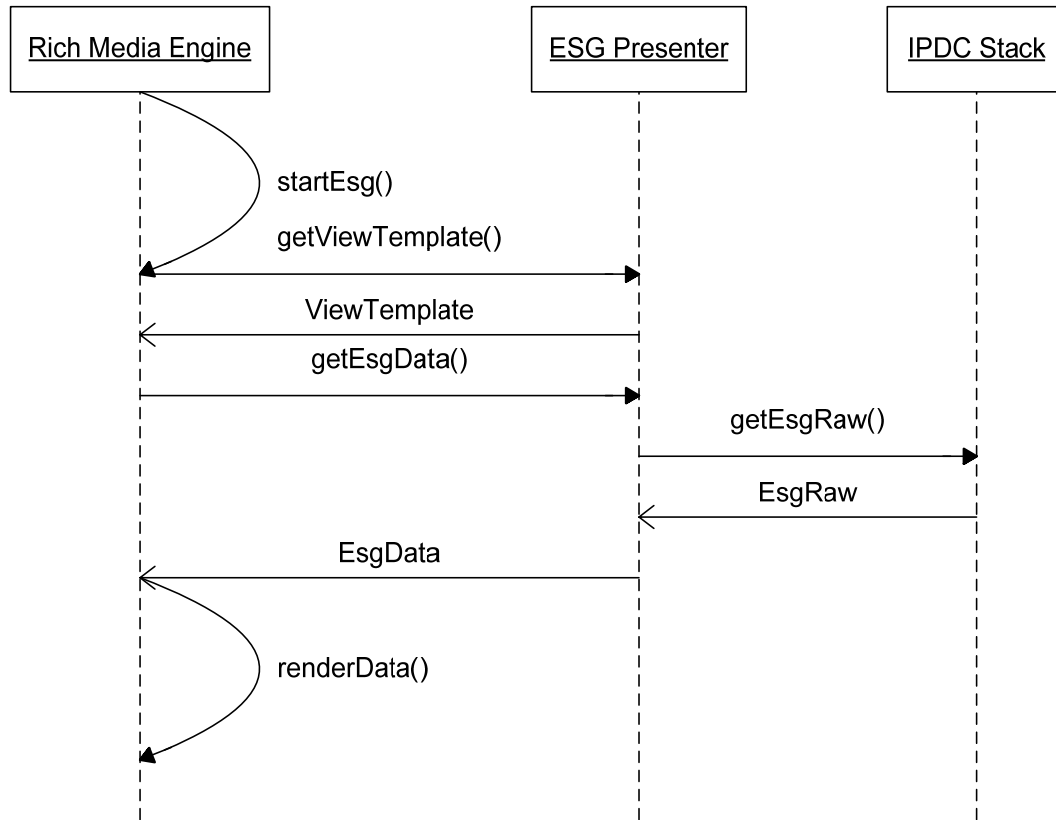


Figure 12 Bootstrap ESG MSC

Generic data acquisition use case

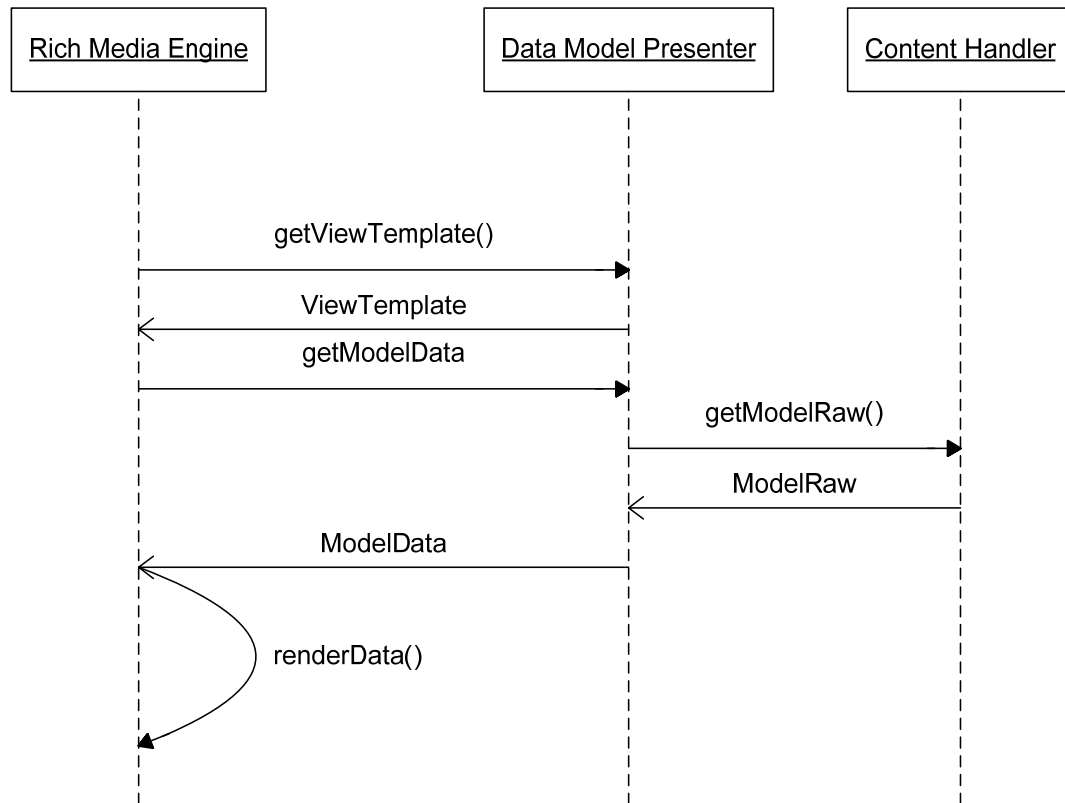


Figure 13 Generic data acquisition MSC

Voting use case (ISAP)

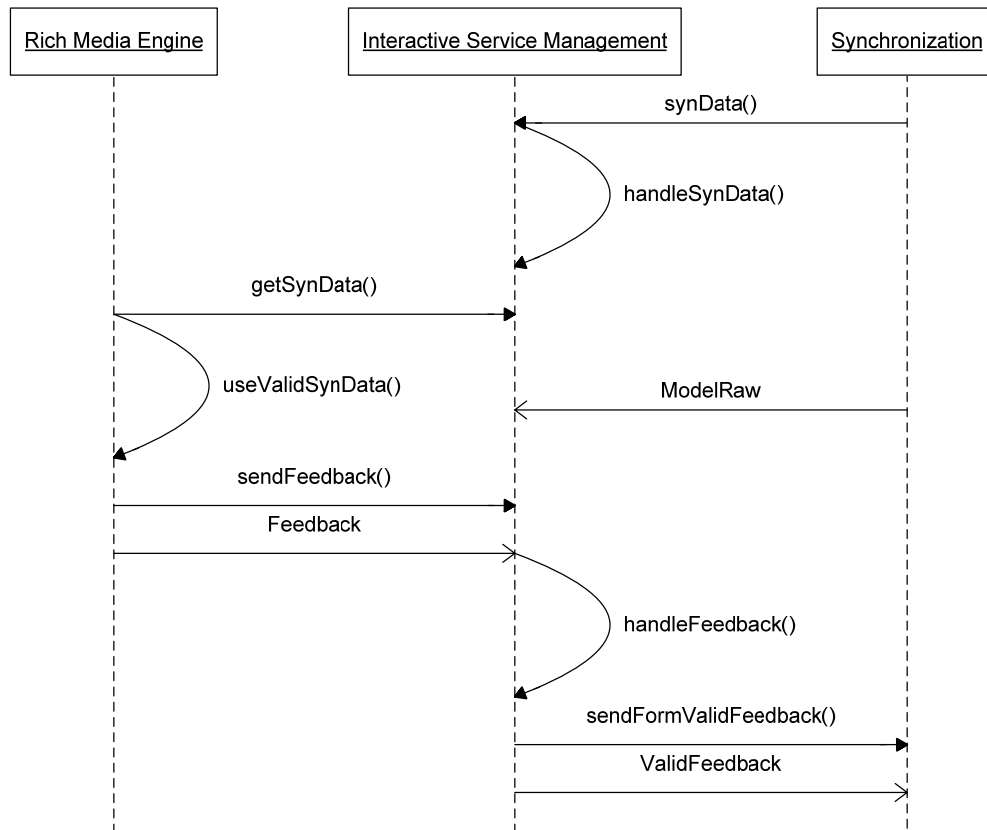


Figure 14 Voting MSC

3.2 Advanced architecture

The Advanced Architecture contains additional software modules in comparison to the Olympic Architecture. These components are a Java VM (as described in [1]), a Java Adapter with specific Java APIs and the new Application Types “Java Application” and “Combined XRM/JRM Application”. In other words, the Advanced Architecture adds a complete Java execution environment that runs parallel to the XML based Rich Media Engine. An analysis of Java-based Rich Media (XRM) and XML-based Rich Media (XRM) solutions [2] has identified limitations and benefits of both technologies. In consequence XML-based Rich Media technologies and Java technologies might perfectly complement each other. Therefore, work package 4 decided to follow this research-oriented approach and to implement and analyze this more complex architecture. As it might exceed the capabilities of the Olympic trial device this architecture is intended to be used for demonstrations and exhibitions and will integrate pure research-oriented middleware developments. It allows coexistence of Rich media streamable solutions with Java compatible applications. By doing this, innovative and complex services may be implemented and demonstrated. The core architecture is similar to the Olympic trial device. In the figure below the dashed boxes show the extensions that are needed for the advanced device. The specific modules are described in the following sections.

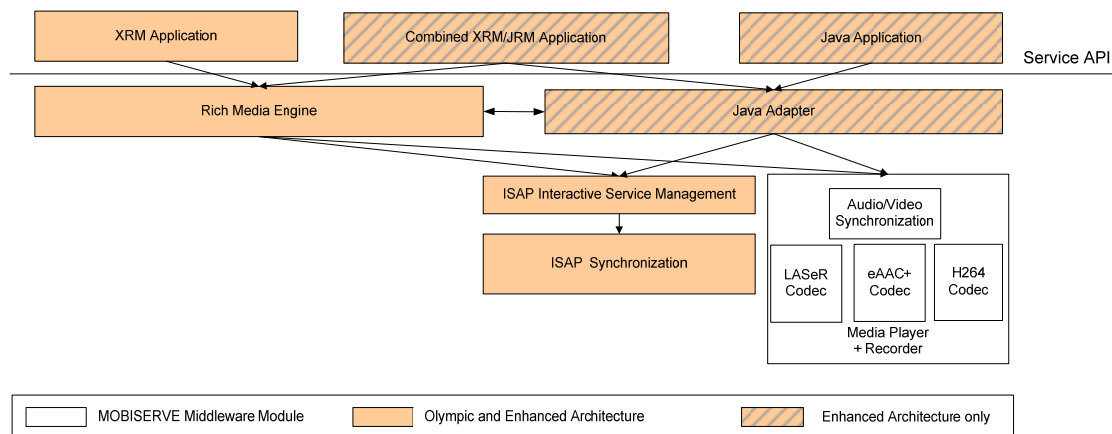


Figure 15 Component description and interfaces

3.2.1 Java Adapter

As already shortly described in Deliverable 4.1 the Java Adapter represents the interface between native-based and Java-based software components. It is based on the Java VM component and adds fundamental Java modules (i.e. the Java configuration) and mobile handset domain-specific APIs. For API functionality that already exists on the device in the form of native C-based APIs, the Java Adapter component provides Java adapter classes which directly map the functionality to Java APIs. All other API functionality is directly implemented by the Java Adapter component. Possible Java-APIs implemented by this application middleware component are:

- Mobile Multimedia API for AV-Access (JSR 135)
- File Access API (JSR 75)
- Messaging API for sending and receiving SMS and MMS (JSR 205)
- Mobile Broadcast Service API for Access to broadcast data (JSR 272)
- Further MOBISERVE terminal specific APIs

3.2.2 Application Types

The availability of both execution engines (Java and XRM Engine) enables three different types of applications on the device.

These are XRM Applications, Combined XRM/JRM Applications and JRM Applications. All of them use specific APIs offered by the two engines (see Figure 16). The Java Adapter provides specific Java APIs that were already mentioned in the previous section. The APIs provided by the XRM Engine are twofold. On the one hand it offers the typical LAsER-XRM API. On the other hand it provides specific Java APIs that may be accessed by Java-based applications. A list of possible APIs is described in the section related to the Combined XRM/JRM Application.

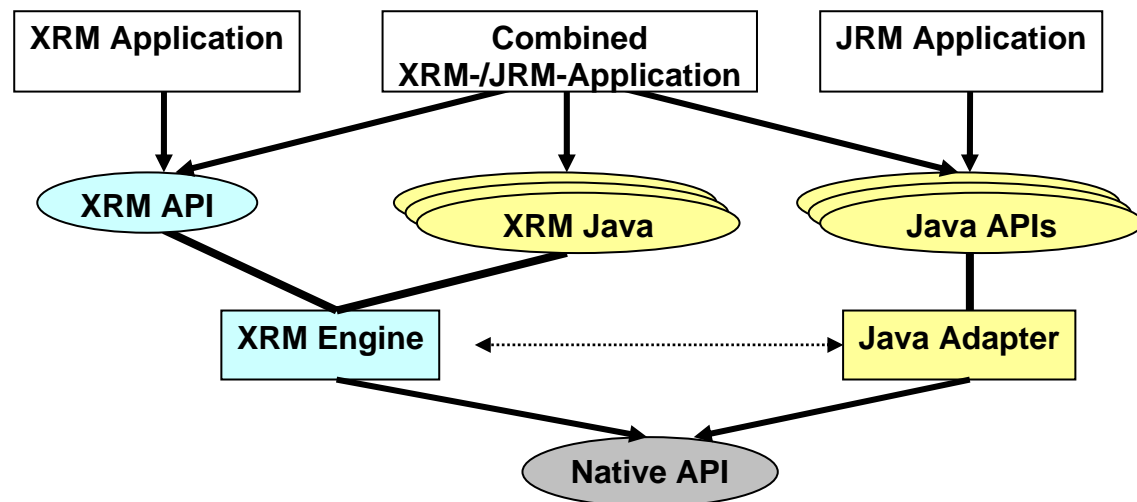


Figure 16 APIs of the Advanced Architecture

XML based Rich Media applications

XML-based Rich Media applications (XRM) are interpreted directly by the XRM Engine. Within MOBISERVE these will be LAsER applications. In particular, these are rather presentation-based services in contrast to the other application types.

Java Application

Java-based Rich Media applications implement Rich Media functionality by using Java classes. Accordingly, this type of application is already known from mobile phone games in the market. Java applications run on top of a Java Virtual Machine (JVM) and access several device functionalities via JVM-APIs or additional JSR-API implementations.

Combined XRM/JRM Application

The availability of both execution engines provides a third application type, the “Combined Java and XML based applications”. This novel type of application is of particular interest within the MOBISERVE research activities. It enables the usage of both XML-RichMedia-based and Java-based mechanisms within one service. Accordingly, content creators are able to choose the type of implementation technology even for service fragments. Two different approaches of combined applications are possible. For instance, a Java class may represent the application core that accesses XRM functions via special APIs. On the other hand, an XRM-based application core is possible which accesses Java-based classes for certain tasks.

These two possibilities are described in the following sections.

XRM based Application core

Currently XRM-only services with logical requirements use scripting technology (e.g. JavaScript and ECMA-Script respectively). This technology is simple and easy to implement. The scripting code may be included in the XRM-description with low effort. However, a range of functions in the extend of those provided by the functional range that is provided by a JRM module can not be used. Furthermore, scripting language-based approaches may cause security issues compared to the Java-approach. The reason is that script interpreters are usually implemented in native C code. In contrast, the Java logic runs in a Virtual Machine Sandbox which provides certain security functionality by design (type safety, overflow prevention, etc.).

Consequently, including full Java classes executed by the JVM is the much more promising approach and therefore investigated within MOBISERVE.

It might be realized as depicted in the following figure. By defining specific XML tags within the XML document a Java class might be linked to the XRM core. As the XRM engine interprets the XML file, specific functions within the Java tags might initiate the start and thus the integration of a Java class. Notification and communication APIs provide mechanisms to access and control XRM service elements from JRM elements of a service and vice versa. Thus, the XRM Engine is able to initiate the Java class execution by the Java VM. Accordingly, the Java VM returns possible results to the XRM Engine. The latter incorporates and presents them on the screen. Consequently, an XRM-based service core approach with JRM-based logic packages is the best choice for presentation-oriented services.

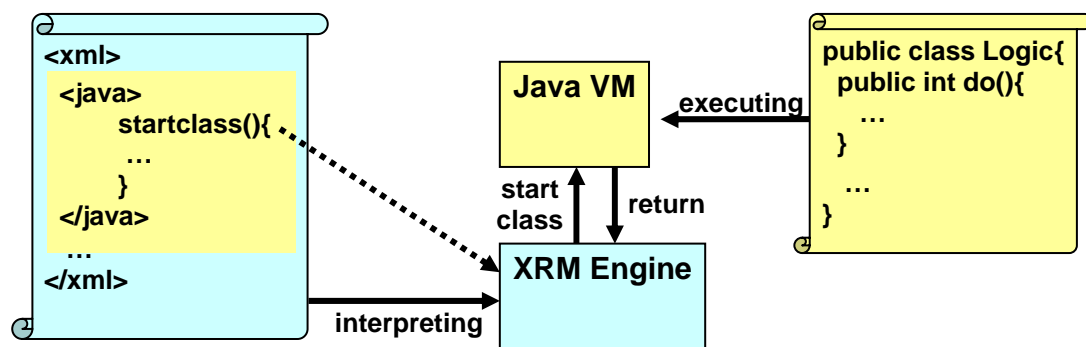


Figure 17 XRM core approach

Java based Application core

For services with complex application logic, the JRM-based package should form the core of the service. Additionally, XRM-based elements may be added to the service in order to avoid user interface-related Java implementations and to enhance its flexibility (see figure below). The JRM-based core will rely on a set of Java APIs provided by the XRM Engine. For instance, an SVGT-API for SVGT 1.1 (JSR 226) or SVGT 1.2 (JSR 287) might be specified. These APIs implement interfaces based on the “Document Object Model” (DOM) and “Micro Document Object Model” (uDOM) respectively as defined in the SVG specification. Beside vector graphics extensions using SVGT further XRM-APIs for Java are imaginable. They might provide some APIs to cover parts of the SMIL specification, a widgetset API like XUL or even a complete Java-API for MPEG-LASER. For future applications a 3D-API such as X3D (Extensible 3D) might be implemented as well.

These XRM-based APIs for Java are one of the major parts of the MOBISERVE middleware research activities. Obviously, this will extend the Java-based APIs and offer several benefits for both the application and its developers.

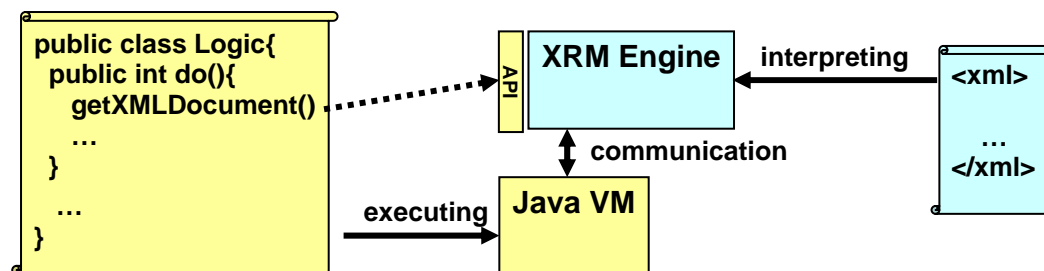


Figure 18 JRM core approach

Example of a “Combined XRM/JRM application”

Here an example of a “Combined XRM/JRM application” is given in order to exemplify the idea and to point out the advantages of the approach. In general, each application consists of various functionalities that might be separated into functional modules. These modules interact with each other and thus form the complete service.

An example of such an application might be a modular E-Shop Application that consists of the following functional modules:

- Application lifecycle
- Shop logic (flow control)
- Memory management
- Remote interactivity
- Scalable graphical output
- Widgetset

The first four modules are rather logic based while the latter two modules are related to the graphical user interface. As already mentioned in the sections above, particularly the JRM approach provides extended logical functionality that is necessary for realizing the first four modules. Some of the modules would even not be feasible with a XRM only solution. As XRM shows a lack of logical functionality, it offers excellent mechanisms for realizing a user interface. Therefore, the latter two modules might benefit from this advantage. Consequently we achieve an optimized “Combined XRM/JRM application” that takes advantage of both execution Engines, the XRM Engine and the Java VM.

4 References

- [1] MOBISERVE Deliverable 4.1: Terminal System Architecture Specification, for Olympic trial
- [2] P. Steckel, M. Spika: The hybrid Java and XML-based MOBISERVE Rich Media middleware for DVB IP Datacast, IST Mobile & Wireless Communications Summit, Budapest, Hungary, 1-5 July 2007