An architecture for multimedia content management

A.Matellanes, A. May*, P. Villegas [†], F. Snijder, A.Kobzhev, E.O.Dijk [‡]

*{adrian.matellanes,tony.may}@motorola.com. Motorola Labs. Jays Close, Viables Industrial Estate, Basingstoke, UK

[†] paulo@tid.es. Telefónica Investigación y Desarrollo. Valladolid, Spain.

[‡] {freddy.snijder,esko.dijk,alexander.kobzhev}@philips.com. Philips Research. Prof. Holstlaan 4, Eindhoven, The Netherlands

Keywords: Software Architecture, Multimedia Content, Multimedia Analysis, Knowledge Extraction, Pro-active Content

Abstract

This paper[1] presents a software architecture for digital multimedia content management. The architecture presented is portable and allows complex and powerful content analysis techniques to be deployed on a wide range of devices. It enables the deployment of intelligent pro-active multimedia content and facilitates user tasks when dealing with multimedia content.

1 Introduction

This paper presents a system architecture that enables a novel approach to personal and commercial digital content management. This approach enables knowledge extraction from digital multimedia content, and realizes autonomous, pro-active digital content, capable of managing itself, independently from classical, static and device-dependent, content management applications.

This architecture is part of the aceMedia project (http://www.acemedia.org).

This paper describes the architecture from a high-level, component-like point of view, explaining the nature of the components and how they are integrated.

Since the end of the 1990s, both commercial and personal digital content have seen a major proliferation. Users are gathering an ever-increasing amount of digital content and need means to manage it, reducing human intervention and making content management an intuitive task.

Next to this proliferation, digital content is becoming more and more *nomadic*. For instance, digital content is nowadays easily carried, rendered and shown anywhere and anytime using small-form factor devices, such as mobile phones or MP3 players. Furthermore, moving, copying and sharing of digital content between devices and users is made easy due to the advancements in static and mobile networking technologies, methods and applications. This view of nomadic content has prompted the authors to rethink the current paradigm of static, device-dependent, content management applications.

It is proposed that nomadic content requires to be able to autonomously and pro-actively manage itself. This will enable the content to be, to a large extent, independent from the variable content management capabilities of the different devices it resides on over time.

Autonomous and pro-active content is accomplished by the introduction of the Autonomous Content Entity (ACE), which consists of three layers; the digital content itself, the metadata layer made of manual and automatically extracted semantic annotations, and a content management intelligence layer.

The content, metadata and intelligence of an ACE will move together wherever it is carried, copied or moved. This enables, for instance, an ACE to self-govern *who* can do *what* with the content, wherever it resides; the ACE can be responsible for gathering and managing its own metadata annotations wherever it is; depending on the context, it can consistently present and adapt itself in a way that might be unique to the content.

Extraction and inference of semantic metadata from digital content in combination with semantic reasoning is required to drive the autonomous and pro-active "self-management" of ACEs. It is further required to enable intuitive, semantic-level, management of ACEs by users. Knowledge processing is thus a hallmark of the system architecture presented in this paper.

2 Architecture overview

The presented system architecture consists of an *aceFramework,* which can be seen as a kind of generic content management middleware enabling the usage of ACEs.

The framework enables ACEs to run, and reuse base content analysis functionality, shared by all running ACEs. The framework further enables users to control and restrain the behaviour of ACEs. An application interface allows users to interact with individual ACEs and manage ACE collections. A networking component in the framework allows for

[1] This research was funded by the European IST project aceMedia - http://www. acemedia.org under contract FP6-001765.

communication among ACE peers for distributed content management applications and to enable the transfer of ACEs between systems implementing the aceFramework. An ACE repository system is described enabling storage, querying and retrieval of ACEs as coherent entities.

The overall aceMedia software architecture consists of a framework on top of which applications are executed. The purpose of this *aceFramework* is to provide a convenient and secure environment for ACEs to be run, content analysis modules to be plugged into, and content to flow from one device to another. For example, an application could use the aceFramework to browse ACEs, to execute an ACE, or to analyze the content inside an ACE using some of the tools provided by the aceFramework. An aceMedia system will then be composed of an instantiation of the aceFramework together with one or more applications on top of it and provides functionality, either locally to users interacting with the system, or remotely (via a remote aceFramework).

Figure 1 shows a high level overview of the components in a single aceMedia-enabled device. The aceFramework must be present on a device to make it aceMedia-enabled. On top of the aceFramework there can be any number of applications using it.

The mission of the aceFramework is to provide a set of standardised services to aceMedia applications, so that those applications are developed more easily and more consistently, facilitating reusability. Since aceMedia systems must run in a number of devices, e.g. PCs, mobile handsets and set top boxes, the aceFramework has been implemented in Java to maximise portability.

Inside the aceFramework a number of components provide all its functionality which can be categorised according to the type of service each component provides:

a) Administrative services are base utilities, needed for any aceMedia system to work, and are provided by the *acecore* subsystem. For interfacing purposes, this is in turn wrapped by a utility layer that provides developers with easy and standardised access to its functionality.

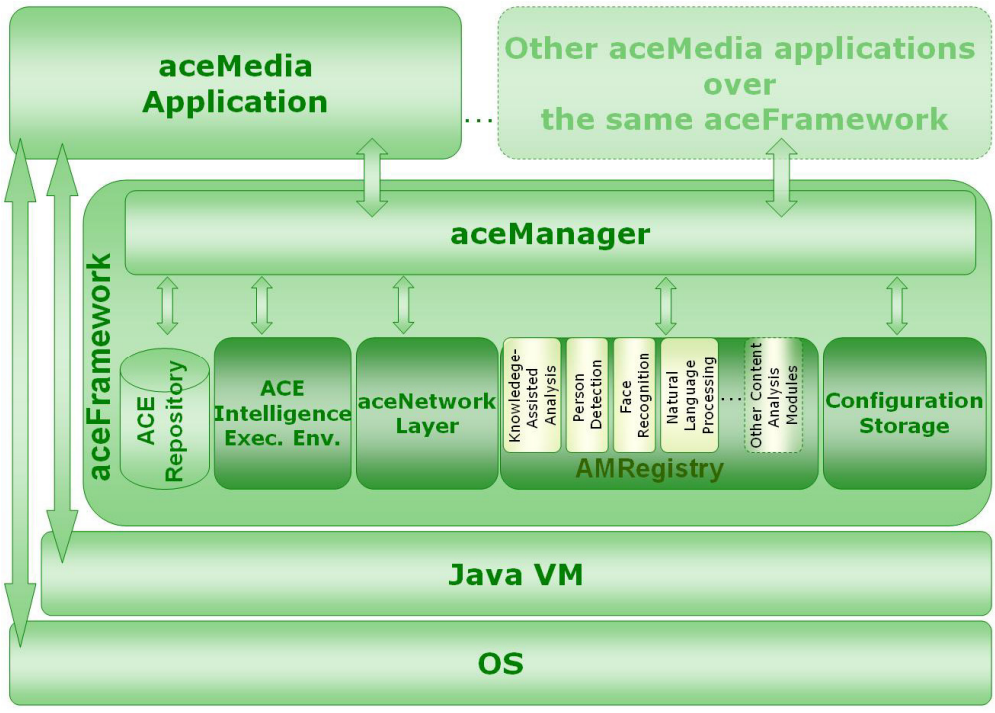
b) Media processing services are then provided by independent components called *Application Modules* (*AMs*). The use of AMs enables aceMedia systems to be customised by including in them just the functionality needed, and also making it possible to upgrade them. AMs are further discussed in section 4.

This modular architecture in aceMedia makes it possible to adapt the system to a great number of environments, from resource-constrained devices to full-fledged systems. While the acecore is required to be local on an aceMedia-enabled device, application modules may reside on the same system or can be located on remote aceMedia systems and be accessed via the network facilities mentioned in section 3, thus effectively creating a distributed system for multimedia processing.

The acecore component consists of six connected software components, as Figure 1 shows:

1) The central one is the *aceManager*, which coordinates and controls the communication between the other components within the aceFramework on the one hand and between the other components and the outside world on the other hand. It is akin to a kernel in an Operating System.

Figure 1 - aceMedia architecture



Additionally it provides base functionality such as user session creation, authentication and logging.

2) The *aceRepository* provides storage and retrieval of ACEs. Its interface hides the actual implementation of the repository, which could be a simple file system, but also a smart database with fast indexing capabilities. This provides the scalability to handle multiple aceMedia platforms, for example mobile phones, set-top boxes and high-end servers at a content provider.

The aceRepository is

the place where both media content (the content layer of ACEs) and knowledge about the content (the metadata layer) resides with the ACE intelligence layer. The aceRepository provides convenient access functions to the ACEs and their layers. It is further discussed in section 5.

3) The *aceExEnv* is the ACE execution environment. It is a confined environment to run the intelligence layer of an ACE using Virtual Machine (VM) technology. A confined environment is needed to maintain system integrity, as ACEs may contain untrusted code. The execution of an ACE can, for example, be requested by an Application or another ACE. However, the ACE execution in the aceExEnv is always initiated and controlled via the aceManager.

4) The *aceNetworkLayer* handles all communication with third parties. Communication is needed in the aceFramework for several purposes, for example:

   - To receive an ACE from another aceMedia-enabled device, or to send an ACE to such a device;
   - To share collections of ACEs between several households over the internet (peer-to-peer networking);
   - To allow ACEs to communicate with other ACEs (ACE peer-to-peer communication);
   - To let ACEs contact a designated trusted web service from which they can download additional information, content and/or metadata.

   Generic networking functions are provided by the *aceNetworkInterface*, independent of the actual protocols that are being used to communicate with third parties over the network.

5) The Application Module registry (*AMRegistry*) is shown in Figure 1 containing several Application Modules (AMs). All AMs are registered in the Application Module registry (AMRegistry). The AMRegistry provides a general interface to register, request and get access to AMs. It keeps track of all AMs on the local device. If an ACE or an Application wants to use an AM, it has to request access to the AMRegistry via the aceManager first.

   AMs themselves can make use of other AMs. Take for example a Person Recognition AM that can identify which people are present on photos. This module could make use of a Face-region Detection AM if that is available. If needed, the Person Recognition AM can make a request to *aceManager* for permission to communicate with the Face Detection module, and then set up a fast communication link between them.

6) The *Configuration Storage* component safely stores user's preferences, network profiles system settings and system defaults that relate to the local aceMedia device. Such settings can be set by the user via an application on top of aceFramework. The Configuration Storage ensures that settings can only be changed by the component that set them in the first place. Also, ACEs are generally not allowed to read or change the configuration settings.

3 Networking

Networking Requirements

The aceMedia network layer is responsible for handling all communications between implementations of the aceMedia framework over an IP-based network. IP networks can be complex, requiring the aceMedia network layer to communicate through network proxies and firewalls. The aceMedia network layer is required to provide both peer-to-peer network services and client-server network services so that ACEs can be shared between users, and centralised ACE repositories and services can be offered to aceMedia users. The aceMedia network layer has to support many terminal types, e.g. PC, set-top box, and mobile devices, and connection types, e.g. GPRS, dial-up, 3G, broadband.

Terminal and Network Profiles

To allow aceMedia applications, ACEs and application modules to adapt ACEs to different networks and terminals, the aceMedia network layer is required to retrieve information about the capabilities of a remote terminal and the network that links the remote terminal to the local terminal.

The terminal profile contains information such as the processor speed, the display characteristics, available codecs and the battery status. The network profile contains information such as the maximum and minimum network data rates and the available transport protocols. The profiles are based on the MPEG-21 DIA (Digital Item Adaptation) Network Characteristics and Terminal Capabilities[1]. To support both static and dynamic profiles and hence support varying network conditions and device handover, the aceMedia Network layer provides functions to retrieve all or only a part of a profile.

Network Middleware

As explained, the network requirements are very complex, so several alternative middleware technologies were studied, including JADE [14], [15] and uPnP [16], [17], before it was decided that JXTA [3], [13] provided the best basis for the aceMedia network layer. JXTA provides a virtual network layer which eases the setup of aceMedia peer-to-peer networks. This allows connections to be made between two peers via another peer when a direct IP connection is made impossible e.g. due to firewalls or NATs (network address tables). JXTA also provides support for managing changing network topologies which allows the aceMedia network layer to support mobile devices.

JXTA provides unique peer IDs for each platform which allows the aceMedia network layer to uniquely identify remote platforms. JXTA provides an advertisement protocol that allows aceMedia peers and servers to be discovered. The protocol also allows the services and ACEs provided by aceMedia terminals to be advertised.

Protocols such as SOAP [7], JMS [8], XML-RPC [9] and RMI [10] are available for use over JXTA virtual networks. JXTA based implementations of these protocols use the unique peer ID instead of an IP address. JXTA provides support for encrypting data sent over the JXTA virtual network and provides an authentication mechanism (username and password) that can be used directly to authenticate an aceMedia user.

Since JXTA provides a large collection of protocols to support peer, service and ACE discovery, the aceMedia network layer provides an additional aceMedia centric interface that provides aceMedia application modules, applications and ACEs a simple interface to discover and communication with other aceMedia systems.

4 Creating and using Analysis Modules

Application Modules (AMs) are the workhorse of the aceMedia system; they provide services for the benefit of ACEs, other Application Modules or an aceMedia application. These modules can perform tasks, triggered by a user, by the ACEs themselves or by the aceManager.

Application Modules are managed through the AMRegistry, which enables seamless installation and addition of AMs. Any aceMedia component (applications, other AMs) can query the AMRegistry to check for the presence of AMs providing the desired functionality and then can call them to perform the desired processing tasks.

AM architecture

The AM definition and requirements have logically motivated the need to use an architecture based on software components. To enable the functionality required for AMs we have based their structure (and by extension the whole aceFramework) on the OSGi architecture. The OSGi™ specifications [2] define a standardized, component-oriented, computing environment for services, in particular network services. Adding an OSGi Service Platform to a networked device (embedded as well as servers), adds the capability to manage the life cycle of the software components in the device. Software components can be installed, updated, or removed on the fly without having to disrupt the operation of the device. Software components are libraries or applications that can dynamically discover and use other components. Software components can be bought off the shelf or are developed in house.

In OSGi terminology the components offering functionality and supporting dynamic deployment and execution are called *bundles*. aceMedia AM*s* are managed as OSGi bundles and have to be registered to be known by the other aceMedia system components. An aceMedia AM is very close to the concept of an OSGi Service/Bundle, having a number of desirable properties:

- An AM is a separate piece of code that can be added to the aceFramework. The procedure for the definition of an AM and its public interfaces are clearly specified, which helps developers to deliver and integrate an AM.
- An AM has a life cycle. AMs are started, stopped, installed, updated, and removed on the fly, without stopping the system.
- Several implementations of the same AM can be accessible on the same system at the same time, possibly with different properties, e.g. different languages, or different locations.
- An AM should be available to any other parts of the system, via registration and search and retrieval mechanisms.

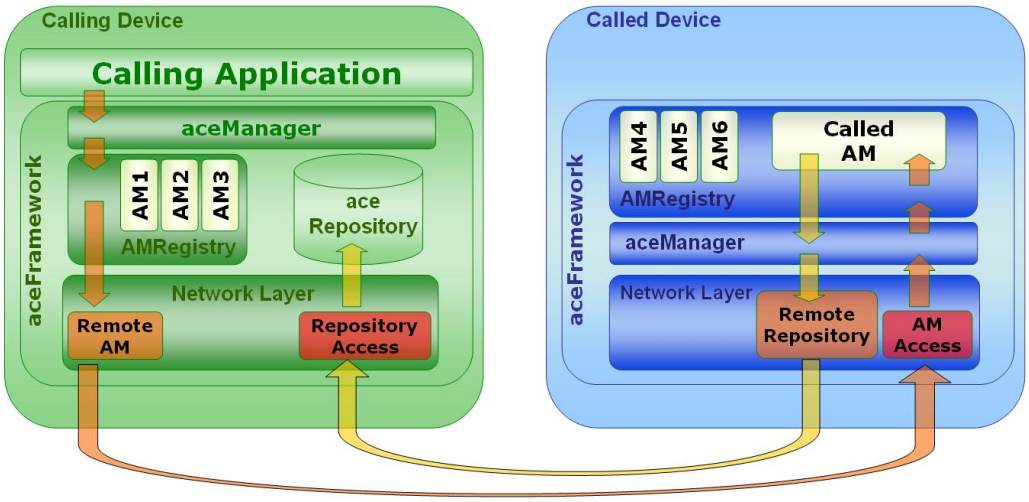
Remote invocation

An additional important feature of the aceMedia architecture is the ability for an aceMedia system to call remote AMs, requesting specialised processing not available in the local aceFramework. This uses the networking layer described in section 3 to locate and access aceMedia peers with the desired functionality.

Figure 2 summarises a remote invocation process, in which a local component requests the help of a remote AM to aid in the processing of a local ACE. It roughly involves two steps:

a) First the local system finds out a remote system providing that functionality (by using the discovery tools made available to the AMRegistry by the networkLayer)

Figure 2 - Remote invocation of AMs



and starts the operation of the remote AM, giving it precise instructions to carry out.

b) Then the remote AM uses the network access to obtain the needed media and metadata components of the ACE from the local system. The remote AM performs the required operation and accesses the local aceRepository to update the ACE with the results.

This remote invocation and access effectively creates a distributed architecture, in which different systems can collaborate based on their capabilities and agreements.

Overview of currently existing AMs

The first aceMedia prototype is very much centred on personal content services, specifically images, and thus AMs have been devised to process such types of media information. The AMs defined until now in aceMedia are able to generate and manipulate the knowledge, metadata and content that will be embedded into the ACEs or displayed to the user.

This is a general list of what has already been developed:

1) Audiovisual processing and characterisation tasks, grouped in a connected set of tools named *aceToolbox* [11], which include low-level media descriptor production descriptors such as colour descriptors (histograms, dominant colours), texture descriptors, edge and shape definitions, etc, many of them based on the corresponding MPEG-7 image descriptors. This group also includes the production of image spatial segmentations.

2) Specialised image understanding tasks, such as high level feature detection (Face Detection and Person Detection), ontology-based image cataloguing (e.g. indoor-outdoor decision), and knowledge-assisted analysis, capable of assigning semantic labels to segmented region images according to a domain ontology e.g. sky, sand, beach, for a simple holiday ontology, or, in the future, more complex entities such as players and events in a more elaborated sport ontology [12], [18], [19], [20].

3) A search and retrieval AM, able to perform both text searches through semantic annotations (with the help of a natural language processing AM capable of extracting and matching concepts from natural language texts) and visual similarity searches, including user relevance feedback for enhanced results.

4) Content adaptation and personalisation. This includes a cross media engine (based on MPEG-21 DIA concepts [1]) able to automatically adapt media formats to terminal, network and user profiles, and personalisation tools such as personal filters to adapt search results to user preferences and profile managers.

As it can be seen, AMs can be quite heterogeneous in their functionality. The architecture is generic enough to allow all this different functionality to be encapsulated into it. Thanks to this an aceMedia system can be continually upgraded, as new content analysis AMs are developed in the future.

5 Content and knowledge storage

Present day content management and data sharing systems are unimaginable without heavy network communication. Adding intelligence to content extends both content and metadata interchange across the networks. Consequently, the metadata representation and the format of network messages have to be standardised in order to achieve true interoperability across various platforms. Over a number of years the Extensible Markup Language (XML) has become the de facto encoding for information exchange on the Net. It is the basis for software stacks implementing peer discovery, messaging, remote querying, and data transfer services (JXTA, UPnP, Gnutella etc.) The aceMedia project benefits from XML too. The interfaces and protocols of aceMedia software are based on XML.

The architecture of the aceRepository has been created with the primary goal of providing uniform, clear, and platform independent interfaces for metadata manipulation. The usage of XML as a metadata representation format allows higher flexibility and extendibility in comparison to the conventional relational model or the object-oriented approach. The aceMedia metadata schema is not strict so that an application can extend it at run time by any valid XML fragment. This flexibility, however, reduces the simplicity of implementation. The design of the aceRepository is a trade-off between flexibility and performance. The repository abstracts from the actual database technology that is used for a particular implementation. It provides only the interfaces and illustrates the data management approach. Therefore, different aceMedia enabled systems can have varying database implementations.

The ACEs stored in the repository are split into three parts – content, metadata, and intelligence. The content and its associated intelligence code are stored on the local or remote persistent storage. The aceRepository automatically maintains their associations with appropriate content metadata. The metadata itself is stored in a Relational Database Management System (RDBMS) serving as a proven backend solution for handling structured data. It is possible to map the XML-based metadata to a set of relations efficiently by analysing this data and its access patterns. Such a mapping allows an increase in the querying performance in comparison to a raw XML search. All data is stored in one “virtual” table but the most searched fields (e.g. AceID, Title, Description etc.) are mapped directly to the columns of that table. This technique enables the majority of the popular queries to be executed quickly and efficiently.

The XML metadata may contain fragments of semantic metadata, formatted using the RDF/XML syntax [6]. In this case it could be beneficial to include a reasoning knowledge base into the aceRepository, to enable semantic queries on ACEs using RDF queries.

The results of the aceRepository queries are represented as a DOM-document containing all metadata for the ACEs that matched the search criteria. This implementation of the query result allows the use of the JDOM API [4] for accessing the contents of the result set. Using JDOM makes an aceMedia implementation independent of any XML schema changes and subsequently leaves great possibilities for metadata extendibility. The XPath expressions [5] can be used to simplify browsing the query results. XPath support is natively provided by JDOM.

The aceRepository design demonstrates a balance between fast searching on the one hand and metadata extendibility on the other hand.

6 Conclusion

In this paper we have presented an architecture that enables multimedia content management and the deployment of intelligent pro-active multimedia content.

ACEs, as a realization of pro-active multimedia content will run under the control of the aceFramework, which will guarantee system integrity and, at the same time, provide the functional base needed for applications to manage collections of ACEs.

The first implementation of this architecture is showing promising results and further research is being conducted. In the networking area, studies are underway in order to better integrate the aceMedia architecture with web technologies and to assess the system performance when transmitting ACEs across heterogeneous networks. In the content storage area, the integration of the content repository to enable better knowledge exploitation through semantic queries is being studied.

The software architecture described includes some features the authors think are of paramount importance in future multimedia content management systems, such as, remote multimedia processing, distribution, extensibility and portability.

References

[1] ISO/IEC 21000-7, Information Technology – Multimedia framework (MPEG-21) -- Part 7: Digital Item Adaptation

[2] OSGi Alliance. http://www.osgi.org

[3] JXTA. http://www.jxta.org

[4] JDOM. http://www.jdom.org

[5] XPath. http://www.w3.org/TR/xpath

[6] RDF/XML.http://www.w3.org/TR/rdf-syntax-grammar/

[7] SOAP. http://www.w3.org/TR/soap/

[8] Java Message Service. http://java.sun.com/products/jms/

[9] XML-RPC. http://www.xmlrpc.com/

[10] Java Remote Invocation Method. http://java.sun.com/products/jdk/rmi/

[11] J. Stauder, J. Sirot, H. Le Borgne, E. Cooke, E. O'Connor, *Relating Visual And Semantic Image Descriptors*, EWIMT'2004, London, U.K., 25-26 November 2004

[12] V. Mezaris, Y. Kompatsiaris and M. Strintzis, *A Knowledge-Based approach to domain-specific compressed video analysis*, IEEE International Conference on Image Processing (ICIP) 2004, Singapore, 24–27 October 2004

[13] M. Ehrig, S. Staab, and C. Tempich, *Platform Selection*, SWAP EU IST-2001-34103 Project Deliverable D4.1

[14] F. Bellifemine, G. Caire, A. Poggi, G. Rimassa, *JADE A White Paper*, exp - Volume 3 - n. 3 - September 2003

[15] JADE. http://jade.tilab.com/

[16] Microsoft Corporation, *Understanding Universal Plug and Play White Paper*, June 2000

[17] UPnP. http://www.upnp.org

[18] S. Bloehdorn, N. Simou, V. Tzouvaras, K. Petridis, S. Handschuh, Y. Avrithis, I. Kompatsiaris, S. Staab and M. G. Strintzis *Knowledge representation for Semantic Multimedia Content Analysis and Reasoning.* EWIMT 2004

[19] N. Simou, V. Tzouvaras, Y. Avrithis, G. Stamou and S. Kollias, *A visual descriptor ontology for multimedia reasoning.* WIAMIS'05

[20] N. Voisine, S. Dasiopoulou, V. Mezaris, V. Spyrou, T. Athanasiadis, Y. Kompatsiaris, Y. Avrithis, M. Strinztis, *Knowledge-assisted video analysis using a genetic algorithm.* WIAMIS'05