Chapter 20
A Resource Oriented Multimedia Description Framework

Hildeberto Mendonça, Vincent Nicolas, Olga Vybornova, and Benoit Macq

Abstract This chapter presents a multimedia archiving framework to describe the content of multimedia resources. This kind of content is very rich in terms of meanings and archiving systems have to be improved to consider such richness. This framework simplifies the multimedia management in existing applications, making it accessible for non-specialized developers. This framework is fully implemented on the REST architectural style, precisely mapping the notion of resource with media artifacts, and scaling to address the growing demand for media. It offers an extensive support for segmentation and annotation to attach semantics to content, helping search mechanisms to precisely index those content. A detailed example of the framework adoption by a medical imaging application for breast cancer diagnosis is presented.

Introduction

Multimedia content have never been so widely disseminated as they are nowadays and will exponentially be from now on. This is mainly due to the simplicity of tools for the creation and dissemination of content and also to the accessibility of creative people to those tools. The production of good content is not an exclusivity of large production companies with expensive budgets anymore. Videos, music, photos, and other media are achieving multi-million audience, reveling talent artists and spreading messages that matter. However, the more content is represented as media the more it becomes difficult to be indexed by search engines and organized by applications. The drawback of this kind of content is its binary representation, whose intrinsic semantics is unclear for computers. Search engines rely on surrounding

H. Mendonça (✉)
Laboratoire de Télécommunications et Télédétection – TELE, Université catholique de Louvain, Louvain-la-Neuve, Belgium
e-mail: me@hildeberto.com

E. Wilde and C. Pautasso (eds.), *REST: From Research to Practice*,
DOI 10.1007/978-1-4419-8303-9_20,

texts and few metadata to index media content, which works in most cases for search purposes but it also leads to ambiguities, misunderstandings, obscurity, besides limiting possible applications on the exploration of these data.

The identification of semantics in media content can still be done automatically by computers. There is a considerable effort from the computer science and electrical engineering communities on the recognition of patterns in images, videos, audios and so on. They are achieving impressive results in terms of precision and performance, but they are not scalable enough to describe large multimedia repositories in a reasonable time or to be integrated to a webcrawler to index media published on the web. Being more pragmatic, the description of media is more precise and detailed when done manually by users because the content might have different interpretations and might contain meanings that are not easily given artificially. Besides that, pattern recognition still fulfills an important role in the description of media due to its productivity in comparison to manual practice when making short but effective descriptions.

We can see the need for media description not only on the Web, where there is no content limitation, but also in specialized applications, where meanings are carefully inherited from one or more domains of knowledge. These kinds of applications are focused on purpose, such as video surveillance applications detecting unforeseen situations, medical imaging applications detecting intra-corporeal elements using radio and resonance, sport applications tracking players in the game to analyze moves and improve performance, and many others that generate media with meaningful content.

Once recognized, manually by human beings or automatically by computers, media content and their meanings must be directly related and stored for posterior use. In order to visualize a solution for the lack of media content's representation, as presented previously, we realized that: if media content and meanings are directly related, then they can be provided together; if provided together, applications aware of it would be able to stop relying on surrounding content and start making use of semantic descriptions. Therefore, we have elaborated a few requirements to guide the design of a solution for this purpose:

- *Define a data model to represent meanings in media content*: when meanings are ready to be associated to the content, it is important to have a data model that suits well their format and syntax, and also delimits the location where they are present.
- *Reuse what was recognized before*: keep record of recognized meanings would avoid repeating automated recognitions, saving computer resources and also helping the analysis of massive data collected throughout time. Because media content are immutable (they are not changed and if a change is needed a new version is created), none of the records becomes outdated and can be permanently reused for several purposes.
- *Assist search engines on the indexation of media content*: when storing media a minimal content description is needed. Associating meanings to content helps in the indexation process, since media can be found by textual search on all

collected meanings. At the same time, existing search engines may be used somehow as alternative ways to find content in addition to basic database querying.
- *Use open standards for representation, communication and publishing*: to make media and their meanings available for several computers, open standards should be used in order to implement full interoperability. Therefore, information stored may be represented in JSON[1] or XML[2] format, the Internet used as a communication medium, and all stored content available on demand.

With these requirements in mind, we present through this chapter a proposal to make multimedia description openly available, simplify the support for media description for developers and apply the solution to the medical imaging domain, which is relatively complex and relevant for the society. In "Multimedia Description" we describe segmentation and annotation, which are the data structure used to describe media content. We illustrate the adoption of segments and annotations in "Description of Medical Images" using a domain-specific application as an example. Then we present in "The Yasmim Framework" a framework to help developers adding support for multimedia description in existing and new applications. In the sequence, we show in "Adapting an Application to Use the Framework" how an existing application is adapted to support the framework. At last, we conclude discussing benefits and issues of this approach in "Conclusion".

Multimedia Description

A formal data model was created to support the description of multimedia files. This data structure is based on segmentation and annotation techniques. *Segmentation*, or fragmentation, consists of delimiting meaningful regions of media content (Shapiro and Stockman 2001). This process determines whether the region is useful or not for the purpose of the system application. This purpose guides the direct intervention of the end-user when manually creating segments and the implementation of recognition techniques to automatically delimit meaningful elements. *Annotation* is the assignment of meanings to segments in order to describe their content. When done manually by the user, a segment is selected and annotations are written or dragged to it. When done automatically, segments are created by recognition algorithms, indicating where the elements are located in the media, and annotated according to what the algorithm was trained to recognize.

[1] JavaScript Object Notation: http://www.json.org.

[2] eXtensible Markup Language: http://www.w3.org/standards/xml/.

Types of Segments

Spatial

The spatial segment delimits a static region in a visual media content. It can be bidimensional or tridimensional. Bidimensional segments can be used for images and video frames. Tridimensional segments can be used for 3D models. Bidimensional segments contain a set of points involving a region of interest:

$$S_s = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\} \tag{20.1}$$

where x and y are coordinates of the points in the Cartesian plane of the content. The value of x and y are correspondent to the pixels' position of a bitmap media. In a vectorial media, x and y are correspondent to the current canvas size, where the number of pixels is dynamically defined by the graphical controller. The limits for x and y are based on the image resolution and the pixels' gradient. Tridimensional segments contain a set of vertex:

$$S_s = \{(x_1, y_1, z_1), (x_2, y_2, z_2), ..., (x_n, y_n, z_n)\} \tag{20.2}$$

where x, y and z are coordinates of the vertex. The value of x, y and z are correspondent to the current canvas size, where the number of pixels is dynamically defined by the graphical controller.

The use of spatial segments is more frequent in images, whether bidimensional or tridimensional graphics, and less frequent in videos. The full content of an image can be seen at once, while a video is usually composed of thousands or even millions of frames, and its content is more meaningful when these frames are presented as a sequence. Therefore, the segmentation of only one frame might be insignificant.

Temporal

The temporal segment delimits a sequential fragment of audio or video. It defines when relevant information starts happening and when it finishes, but without any spatial delimitation. Taking an audio media as an example: a podcast is a series of digital media files that are released episodically and often downloaded through the web. These files contain rich discussions about relevant subjects, thus they are candidates to be segmented. Each interesting part of the content starts and ends in specific timestamps. The difference between the end time and start time delimits the temporal segment. It is also applicable to videos. A TV news, for instance, presents several subjects during a daily edition. These subjects can be distinguished from each other by their correspondent segments. Therefore, the definition of the temporal segment is:

$$S_t = [T_s, T_e] \tag{20.3}$$

where T_s is the starting timestamp and T_e is the ending timestamp, both included. The difference between T_e and T_s is equal to the duration of the segment.

Spatio-Temporal

The spatio-temporal segment is a merge of the spatial and temporal segments' concepts. It associates a time tag for each one of an uninterrupted sequence of frames. In an implementation level, it preserves all the characteristics of the spatial and temporal segments, but associates additional properties to the spatial segment, which are a sequential number and a correspondent time instant. Because there is a possibility of having two or more frames in the same timestamp, the sequential number was introduced to keep the sequence of those frames, independent of their timestamp. We can conclude that the formal definition of a spatio-temporal segment is:

$$S_{st} = [T_s(\{S_{s1}, S_{s2}, ..., S_{sn}\}), T_e(\{S_{s1}, S_{s2}, ..., S_{sn}\})] \tag{20.4}$$

where T is a timestamp of the temporal segment and S_s is a spatial segment in a certain instant of time. Each timestamp can be correspondent to one or more spatial segments. S_s can also represent a bidimensional (i.e. for videos and animations) or a tridimensional (i.e. for tridimensional graphics) spatial segment. This type of segment is applicable on videos, animations and 3D content and it is appropriate for tracking objects, people and other elements in the scene.

Links Between Segments

Segments by themselves are capable of meeting the needs to delimit several content samples. It allows a precise attribution of meanings to the right location, using annotations. However, there still exists gaps to fulfill in terms of representativeness. These gaps are in the limbo between segments; and our approach to fulfill them is creating *links between segments*. The arrangement of links has several configurations, such as: *sequence*, *hierarchy*, *composition*, *cause and effect*, and others. Seen as a sequence, links indicate that there is a logical order between the segments, as hierarchy they indicate the refinement of a large segment in several smaller ones, as composition they connect the parts of a bigger element, and as cause and effect they indicate the impact that a segment may cause on other segments.

Types of Annotation

The types of annotation go from a simplistic to a robust form of knowledge representation, giving more flexibility to different user profiles. They can be

assigned to segments and links, covering from simple to complex media content. The supported annotations are:

Property

Property is an annotation associated to a label or a key. This key indicates the meaning of the value. Formally speaking, a property is composed of a *key* and a *value*, where the key qualifies its respective value. Properties are appropriate to describe file characteristics, for example: dimensions, resolution, size, format, volume, duration, etc. As an example of a property's syntax, we have `<size> = 25GB`, where *size* is the key and *25GB* is the value identified by the key.

Tagging

Tagging is the assignment of keywords to the media content. Each keyword represents a simple word that identifies the content in the segment or in the links between segments. Keywords are simple, efficient and widely used nowadays to create indexes of information on the web. However, it has limited representativeness when compared to other forms of annotation, although it is more practical for most people and more efficient in terms of searching because most database systems nowadays have good support for text searching.

Transcription

Transcription is a textual and complete description of a speech, dialog or music lyric. Practical applications are the automatic recognition of speech in audio sequences, sub-titles, optical character recognition (OCR) in images containing text, etc.

Description

Description is a detailed text explaining the essence of the media content. It has the same advantages and disadvantages of transcription, but with a different purpose. Practical applications are story telling material, textual summarization, situation description, scenario-based prototypes, etc.

AdHoc

AdHoc does not have commitment to be accurate in terms of content meaning. They could represent opinions, comments, external links, references, etc. AdHoc also does not have any priority in the searching mechanism and it is retrieved when

the related media is already available for the user, appearing as an additional or complementary information. This is due to the fact that AdHoc annotations are informal, free-text, and can lead to erroneous decisions (Kompatsiaris and Hobson 2008).

Domain Concepts

Domain Concepts are a domain specific annotation technique. It uses ontology, which is an explicit specification of a conceptualization, providing a shared vocabulary that can be used to model concepts and their properties and relations (Gruber 1993). Concepts are more representative than tagging because they are well positioned in the domain, but they are also less efficient than tagging because there is an additional cost of exploring the graph of meanings related to them. Comparing with transcription and description, ontologies are less representative, but more explicit and computationally friendly.

The Yasmim Framework

The Yasmim Framework is an implementation of the data model used to describe multimedia content. It was designed to offer a rich description of media, attaching semantics to content of images, videos, audios, and 3D models. Using Yasmim, developers and researchers do not deal with the usual complexity of managing media content. An API to perform operations over those media is made available through web services. Therefore, instead of developing one more multimedia archiving system, we are contributing by simplifying the way multimedia is added to existing applications and making distributed multimedia management accessible for non-specialized developers.

The designed architecture aims to provide scalability, extensibility, and robustness. It is scalable because it is stateless (i.e. it does not save any state or temporary data, such as user's sessions, navigation, etc.) and uses several databases to support different kinds of data. It is extensible because several existing solutions can be used with a minimal integration effort. It is robust because the chosen technologies have been extensively applied on many other solutions, with years of experience and large communities around them.

Yasmim runs entirely on the server. Its user interface is essentially administrative. In order to access and maintain media, application clients should be developed to access the server. The communication is made through web services, using Internet protocols. They are also stateless, thus any temporal data should be managed by the client and sent to the server when necessary. Being stateless allows the system to dedicate all its computational resources to process media.

Figure 20.1 depicts the general architecture of a system using Yasmim for multimedia archiving. Yasmim is right in the middle, intermediating data between

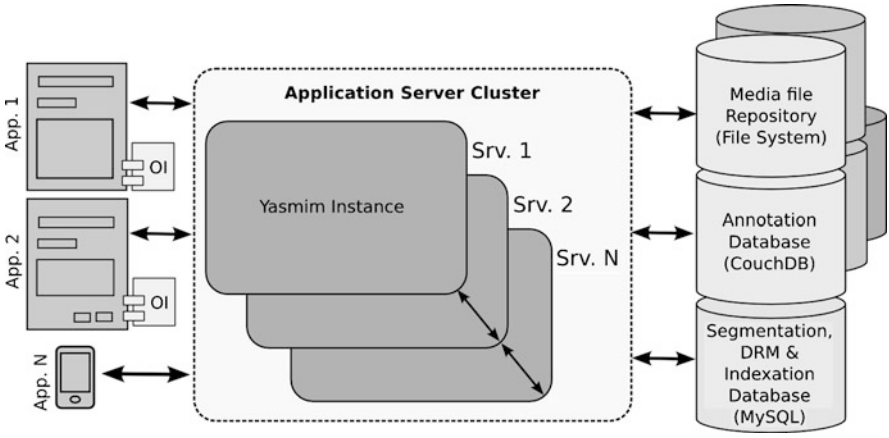

Fig. 20.1 General architecture

several data sources and several clients. In theory, the middleware behind it can be any application server available on the market that implements recent Java Enterprise Edition (Java EE) Specification Requests (Java EE JSR).[3] However, we have tested it only with the Glassfish.[4] It can manage instances of the same application on spread machines, expanding the processing capability according to users' demands.

The application manages the information that come from clients and organizes them in different databases. Each database was chosen according to the data that they were designed to store. These databases are:

- *Media File Repository*: Media files are stored directly in the file/storage system. The optimal efficiency on file access depends on the operating system and the storage system in use. Files are located by name, which is not exactly the original name, but the resource id registered in the database. To verify consistency, a batch process checks periodically whether there is a database record for each stored file. Orphan files are deleted in this process.
- *Segmentation and Indexation Database*: A relational database is used to store references to files in the repository because of its robust indexation mechanisms. It is also used to save segmentation data because tables have better support to store and retrieve numbers, since segments are basically coordinates and/or timestamps.
- *Annotation Database*: Annotations are stored in a document database system, which processes text more efficiently than relational databases.

[3] Java Community Process Java EE: http://jcp.org/en/jsr/platform?listBy=3&listByType=platform.
[4] Glassfish Application Server: https://glassfish.dev.java.net/.

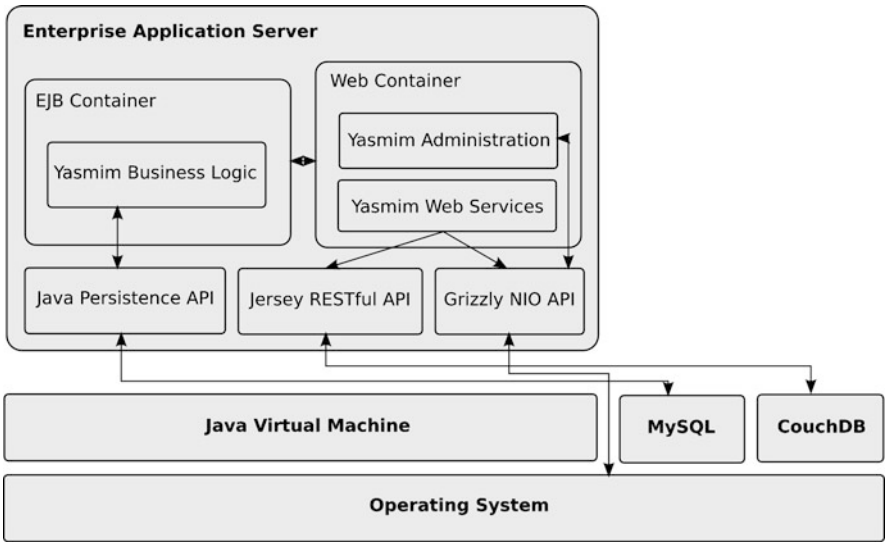

Fig. 20.2 Yasmim software architecture

The role of the client side is to process heavy operations, such as the support for several modalities, automatic segmentation, automatic extraction of meanings, and also to provide rich user interaction for intuitive manual segmentation and annotation. The data is synchronized with the server, making the media and all related data available for searching and sharing.

The server side provides REST web services (Fielding 2000), which is compatible with several kinds of clients developed in different languages, platforms and devices. According to REST architectural principles, the main data abstraction is a resource, which is represented by a media resource in our architecture. Every media and related information are reachable through unique identifiers, following the principle of addressability. Identifiers are known as URI (Uniform Resource Identifier),[5] which is used by the HTTP protocol to locate resources on the web (Richardson and Ruby 2007). With a REST-based framework, we could attach segments and annotations to media, making slight modifications on the URI. This way, not only search mechanisms can benefit from the media description, but many other practical applications as well, since REST web services are easily accessible by any socket library.

Figure 20.2 shows the server side where Yasmim runs. The application server has two execution environments: The EJB Container and the Web Container. The first one is appropriate to handle transactional data, which is suitable for operations with the relational databases. The second one is appropriate to handle and generate

[5] http://www.w3.org/TR/uri-clarification/.

content based on the target users. This content is stored and retrieved by the EJB Container and other sources and appropriately transformed for different purposes. The EJB Container runs Yasmim's business logic that takes care of the database data. The Web Container runs Yasmim's administration, which is the user interface to perform back office (maintenance) operations, and the web services, which are the interface with clients.

These containers have some facilities to access data. The *Java™ Persistence API*[6] is a Java™ specification for relational data access. It is capable of mapping table with Java™ classes in order to transform table tuples in Java™ objects, consequently simplifying data access. The *Jersey RESTful API*[7] is an implementation of the Java™ specification for providing and consuming REST web services. It is used to implement and to consume REST web services. The *Grizzly NIO API*[8] is an implementation of the New Java™ IO specification. It is used to store, retrieve and modify media files asynchronously, optimizing parallelism.

The platform is composed of the Java™ Virtual Machine (JVM), MySQL database, and CouchDB document database. They run on top of the operating system, which is also responsible for the media storage. The JVM is responsible for the execution of Java™ programs, which includes the application server and the applications running on it. MySQL[9] is one of the fastest relational databases available and its indexation, relational and transactional features are essential to deal with a large amount of numbers and unique references, which should be consistent. Finally, because annotations perform an important role in this research, they should be stored in a high scalable way, such as the one provided by CouchDB,[10] a document based database (Anderson et al. 2010). All data in CouchDB is accessible by REST web services, allowing clients to access it directly, without Yasmim mediation, although only Yasmim can write data there.

Catalog of Services

The relevant services for general understanding are listed on Table 20.1. The first column indicates the name of the service, helping the developer to identify which service is more appropriate for his/her needs. Second column shows the HTTP methods, that could be GET, POST, PUT, and DELETE. The third column shows the relative URI, starting with "http://[server-name/domain]/resources". The brackets indicate that there is a value to fulfill. This value could be pre-defined, which is the case of [type], or generated, which is the case of [id]. The last column lists the

[6] JSR 317:http://jcp.org/en/jsr/summary?id=317.

[7] JSR311:http://jcp.org/en/jsr/summary?id=311.

[8] JSR51:http://jcp.org/en/jsr/detail?id=51.

[9] http://www.mysql.com.

[10] http://couchdb.apache.org.

Table 20.1 Catalog of RESTful web services

Service	Method	URI	Parameters
Save media	POST	.../[type]s	
Get media	GET	.../[type]s/[id]	version=#
			width=#
			height=#
			rotate=#
			filter=*filter-name*
			sample=*true/false*
		.../[type]s	search=*keywords*
Remove media	DELETE	.../[id]	version=#
Save segment	POST	.../[media-id]/segments	
Get segments	GET	.../[media-id]/segments	shape=*shp-name*
			type=*type-name*
			duration=#
			search=*keywords*
Get segment	GET	.../[media-id]/segments/[id]	binary=*true/false*
Update segment	PUT	.../[media-id]/segments/[id]	
Remove segment	DELETE	.../[media-id]/segments/[id]	
Save annotation	POST	.../segments/[seg-id]/annotations	
Get annotations	GET	.../segments/[seg-id]/annotations	search=*keywords*
			type=*type-name*
		.../[media-id]/annotations	search=*keywords*
			type=*type-name*
Get annotation	GET	../segments/[seg-id]/annotations/[id]	
Update annotation	PUT	../segments/[seg-id]/annotations/[id]	
Remove annotation	DELETE	../segments/[seg-id]/annotations/[id]	

parameters to be appended to the URI. None of the parameters is mandatory, except for the parameter "search" in the "Get Media" service to avoid a high amount of records retrieved.

The value [type] can assume the following values: "image", "video", "audio", and "3d", which are the types of media supported by Yasmim. These values are mainly useful to summarize the available services. [id] is a UUID,[11] an alphanumeric string of 32 characteres with so many combinations that it theoretically never repeat for two different records. Because each id is unique, data synchronization, replication and merges are very simplified. UUID is used to define all ids, thus [id], [media-id], and [seg-id] follow the same rules.

Each parameter starts after a semicolon and can be written in any order. Some parameters are not appropriate for all types of media. "rotate", for example, cannot be applied to an audio file (Get Media) and "duration" cannot be applied to a spatial segment (Get Segments).

[11] Universally Unique Identifier: http://www.ossp.org/pkg/lib/uuid/.

Only media cannot be updated because media files are immutable. Segments and annotations can be inserted, updated, queried and deleted normally. In case a media file needs to be updated, a new version is created with the new file and the previous version is kept historically. The implementation of filters on the server would impact the overall performance. However, the decision to implement them was made because they are atomic operations, which means that there is only one algorithm for each filter and its output is exclusively used by the client. In order to improve performance, we save a version of the filtered image to retrieve in case the it is requested once again in the future, working as a buffer. The same rules are valid for format. The saved versions have a different file name pattern. Besides the id, the name also have a sequential number and the retrieval of the correct file is managed by the framework.

The hypermedia aspect of the services helps to retrieve media resources according to their respective mime types and also informs to application clients the available filters for each requested kind of media. Yasmim does not offer other references beyond these ones, thus hypermedia is not seen as a workflow but a set of options available for retrieving and processing resources.

Description of Medical Images

Annotation of medical images consists of segmenting and annotating relevant elements in images produced by hospital equipments, such as radiography, ultrasound, magnetic resonance and others. Taking breast radiography as an example, the image may depict anomalies in the breast region that might be a tumor. The analysis of a specialist (doctor) will determine whether the anomaly is a tumor, a calcification, or any other possible diagnosis.

MedicalStudio

There are applications to help doctors on the analysis of such media content. The one that we are taking into consideration is MedicalStudio because we have access to the source code and the application needs a rich support for segmentation and annotation of medical images. MedicalStudio is a component-oriented platform designed to ease the creation of medical imaging workstations (Trevisan et al. 2007). Besides simplifying the work of developers, MedicalStudio also streamlines clinical trials and end-user's experience. The platform provides a collection of reusable components that can be assembled to produce new applications considering image processing, data access, interaction design and others. Each assembly of components will produce a different application that may target different uses. In the case of an image registration application, for example, each algorithm will be seen as a component, and there will be several UI components to meet different user

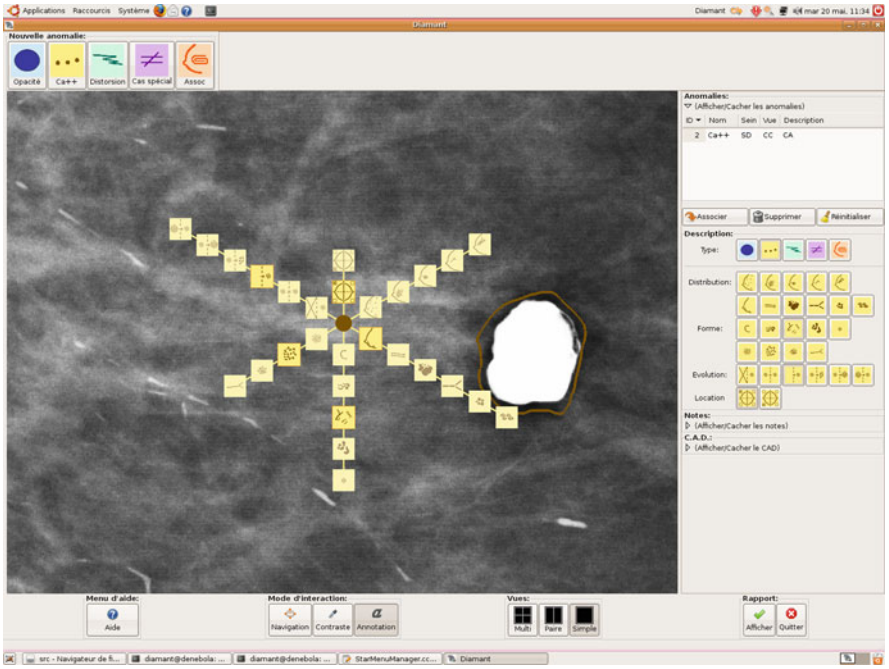

Fig. 20.3 MedicalStudio running components for mammography

profiles: (a) a configuration UI for tuning algorithms used by engineers; (b) another configuration UI for tuning options oriented for clinical researchers; and (c) a visualization UI for doctors to perform their specific clinical diagnosis.

Figure 20.3 depicts MedicalStudio's user interface, where an image of a mammography is shown. Looking at the image, the doctor can visually identify micro-calcifications, select them using spatial segments and annotate these segments using domain-specific annotations, which is well known by the doctor, who is a specialist in the field.

The platform is entirely written in C++ and relies on well accepted and powerful libraries, such as Visualization Toolkit (VTK[12]) for visualization, Insight Segmentation and Registration Toolkit (ITK[13]) for image segmentation and registration, DCMTK[14] for Digital Imaging and Communications in Medicine (DICOM[15]) interoperability and GTKmm[16] for graphical user interface. These libraries are not

[12] http://www.vtk.org.

[13] http://www.itk.org.

[14] http://dicom.offis.de/dcmtk.php.en.

[15] http://dicom.offis.de.

[16] http://www.gtkmm.org.

always enough in specific cases, so that list is not fixed and the architecture is flexible enough to allow interoperability with any other toolkit as far as it can be bound with C++.

Breast Diagnosis Domain Representation

The annotation of mammography for breast cancer diagnosis is a good case to explain how the integration of both frameworks will be valuable. At the same time, it is evident that the case can be easily transposed to other cases of medical image annotation tasks, changing the domain of application. For this particular case, spatial segments are used to delimit what was identified by specialists and domain concepts are used to annotate the segments, describing the medical diagnosis. For this purpose, an ontology was created to explicitly specify each concept of the breast clinical domain. When an ontology is designed, it describes only one knowledge domain in order to be consistent and to provide its coherence, reuse, compatibility with other ontologies and lessen the risk of duplicity and ambiguity (Tudorache et al. 2008).

The ontology is developed by means of ontologies modeling Protégé platform.[17] The considered ontology gives all information about the patient (name, state of health), the type, place and date of study performed for this particular patient, and all possible outcomes of the study. The specialized medical part of the ontology is based on the classification by The American College of Radiology (ACR) that established the Breast Imaging Reporting and Database System (BI-RADS) to guide the breast cancer diagnostic routine (D'Orsi et al. 2003). BI-RADS is a quality assurance tool designed to standardize mammo-graphic reporting, guide radiologists and refer physicians in the breast cancer decision making process, reduce confusion in breast imaging interpretations, facilitate outcome monitoring and patients management.

Calcifications in the system followed by the considered ontology are described according to size, morphology and distribution. The findings are then interpreted and an assessment rendered that includes the degree of suspicion for malignancy, and any pertinent recommendations. This ontology provides the BI-RADS categories that are used to standardize interpretation of mammograms among radiologists. The implementation of BI-RADS categories in the ontology is shown in Fig. 20.4.

The ontology also defines the overall general composition of the breast defining possible type, shape, location of lesions, types of tissue density that can be present in patients. When instantiated with particular data for a particular patient, domain concepts allow efficient interpretation of the data obtained during the study and facilitate decision making concerning the perspectives of treatment and follow-up treatment plan.

[17]Protégé ontologies modeling platform: http://protege.stanford.edu.

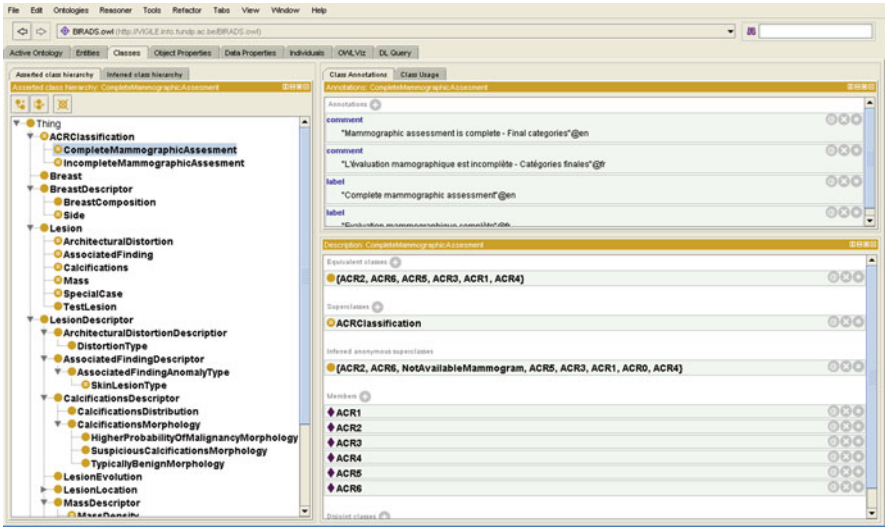

Fig. 20.4 Implementation of BI-RADS standard categories in the ontology

Adapting an Application to Use the Framework

MedicalStudio supports segmentation and annotation, but they were stored in DICOM format, which was not supported by Yasmim because it was developed for very specific needs on the medical domain. In order to support DICOM, Yasmim's data model was carefully compared with DICOM's data model. Modifications have been incorporated in Yasmim's model, since DICOM is a standard and it cannot be easily modified. Therefore, the integration of MedicalStudio and Yasmim depended on how compliant Yasmim's model is to DICOM.

After the compatibility check, MedicalStudio stopped storing in a DICOM format and started storing in Yasmim. The images have been stored in the media repository, segments in the relational database, and annotations in the document database. An additional web service was developed to load all this data and generate a DICOM file on demand. The resulting file can then be distributed to other medical systems. The URI to generate a DICOM file is the following: http://[server-name/domain]/resources/image/[id];format=dicom.

The algorithm used by MedicalStudio to manage this format was migrated to the new web service and the platform started using this service. There were two advantages on this approach:

1. MedicalStudio would become less complex by migrating part of its source code to Yasmim, consequently reducing the maintenance cost.
2. Other medical applications would profit from the new web service, by simply reusing it to generate their DICOM files.

In order to access this and other services, the library LibcURL,[18] a client-side URL transfer library, was added to MedicalStudio allowing HTTP connections to the server.

The current MedicalStudio version implements only one kind of segmentation, the spatial one, and three kinds of annotations, which are:

1. *Property*: used to annotate low level features of the image.
2. *Description*: if necessary, some description of the segment can be added.
3. *Domain concept*: the most common annotation, since MedicalStudio implements an ontology, as described in "Breast Diagnosis Domain Representation".

The process of mammography screening is as follow:

1. *Visualization*: mammographies are analyzed, they are retrieved with the patient records from the PACS[19] and HIS[20];
2. *Lesion detection*: each visible lesion in the image is detected, spatially localized (manually or with automatic algorithm) and categorized in one of the 5 possible type of lesions;
3. *Lesion annotation*: for each lesion a set of standardized characteristics is entered, all theses characteristics are dependants on the type of the lesion, the whole set is organized in an ontology based on a medical standard called BIRADS;
4. *Automatic annotation*: alongside the manual user annotation, a set of automatic algorithms characterises the same lesion with low level descriptions;
5. *Reporting*: finally, a diagnosis report is generated from the whole set of annotations made on all images and sent to the hospital information system.

Analyzing that process, we can exposes the inputs and outputs and maps them to YASMIM services in the integrated architecture. Table 20.2 lists them all. For each input and output there is a related web service, in the second column, and the type of data that has been manipulated, in the third column.

Integrated with Yasmim, MedicalStudio has another immediate benefit, which is the possibility to support multiple domains. It would allow specialists from different medical specialties to analyze the same image, enabling multi-disciplinary diagnosis. It would also allow the annotation of other kinds of images, besides mammographies. And last but not least, all data and annotations will be accessible to other medical clients, that is a simple and standardized way in addition to the complex DICOM document format. That access will allow the development of very thin clients, such as web clients or even smart-phone and tablet clients.

[18] http://curl.haxx.se/libcurl/.

[19] PACS : Picture Archiving and Communication System, usually implemented with the DICOM standard. http://medical.nema.org.

[20] HIS : Hospital Information System, a "in-house" system, but HL7 standard starts spreading. http://www.hl7.org.

Table 20.2 Breast cancer diagnosis: inputs and outputs

Tasks		Service	Data type
1. Visualization			
Outputs:	Images	Get media	Media resource
	Patient records	Get annotations	Properties
2. Lesion detection			
Inputs:	Spatial segment	Save segment	Segment
	Type selection	Save annotation	Properties
		Save annotation	Domain concept
Outputs:	List of types	Get annotations	Domain concept
3. Lesion characterization			
Inputs:	Characterization	Save annotation	Domain concept
Outputs:	List of characteristics	Get annotations	Domain concept
4. Automatic annotation			
Inputs:	User confidence	Save annotation	Domain concept
	Low level characteristics	Save annotation	Domain concept
Outputs:	Image information	Get annotations	Properties
5. Reporting			
Input:	Structured report	Save media	Resource
		Save annotation	Properties
Outputs:	Lesion and characteristics	Get media	Resource
		Get annotations	Properties
		Get annotations	Domain concept

Conclusion

This chapter presented a case study of a multimedia archiving framework fully implemented on the REST architectural style and applied on a medical imaging application. The architectural style represented by REST plays an important role on this evolution, precisely mapping the notion of *resource* with *media artifacts*, and being scalable to address the growing demand for media. A high level and a low level description of the architecture, our decisions concerning the design of the URIs, and a complete catalog of available services were presented.

The main concern about the use of Yasmim, at the moment, is a slightly decrease of performance due to network latency. Because every data is centralized on the server, the network is always considered. This issue might be addressed by saving some temporary data on the clients and synchronize them when necessary. It may also allow the tool off-line work.

Anyway, the adoption of Yasmim by MedicalStudio was positive because it contributed to reduce the complexity of MedicalStudio by migrating part of its features to Yasmim and, consequently, transforming it in a distributed application. Several specialists may have access to the repository at any time and place, and contribute with new images, segments and annotations.

Yasmim is an open source project, under Apache License 2.0.[21] It is maintained in the context of the 3D Media Project.[22]

References

Anderson, C., Lehnardt, J., Slater, N.: CouchDB: The Definitive Guide. O'Reilly Media Inc., Sebastopol, CA, USA (2010)

D'Orsi C.J., Bassett L.W., Berg W.A.: Breast Imaging Reporting and Data System: ACR BI-RADS-Mammography (ed 4), Reston, VA, American College of Radiology (2003)

Fielding, R.: Architectural styles and the design of network-based software architectures. Ph.D. thesis, University of California, Irvine (2000)

Gruber, T.R.: A translation approach to portable ontology specifications. Knowledge Acquisition 5, 199–220 (1993)

Kompatsiaris, Y., Hobson, P.: Introduction to semantic multimedia. In: Semantic Web Services: Concepts, Technologies, and Applications, chap. 1, pp. 3–13 (2008)

Richardson, L., Ruby, S.: RESTful Web Services. O'Reilly Media Inc., Sebastopol, CA, USA (2007)

Shapiro, L.G., Stockman, G.C.: Computer vision. Prentice Hall. (2001)

Trevisan, D., Nicolas, V., Macq, B., Nedel, L.: Medicalstudio: A medical component-based framework. In: Workshop de Informatica Medica - WIM (2007)

Tudorache, T., Noy, N.F., Tu, S.W., Musen, M.A.: Supporting collaborative ontology development in Protégé. In: Seventh International Semantic Web Conference, Karlsruhe, Germany, Springer. (2008)

[21] http://www.apache.org/licenses/LICENSE-2.0.html.

[22] http://mediatic.multitel.be/platforms/3dmedia.html.