Bull. Nov. Comp. Center, Comp. Science, 45 (2021), 69-78

A cassette as a specialized repository for factographic systems

A.G. Marchuk, P.A. Marchuk

Abstract. When published on the internet, cassettes can be used by various information systems needing this information block. This mechanism gives us the opportunity of a flexible system configuration and a realization of distributed information systems.

This approach was implemented in the information systems created by the A.P. Ershov Institute of Informatics Systems SB RAS.

Keywords: cassette, factographic systems, distributed systems, non-specific ontology

Introduction

The factographic approach [1] has been implemented in the A.P. Ershov Institute of Informatics Systems in several information systems [2, 3] designed for digital archiving. Currently, these systems are being modernized and new features are being added. Also, new projects are being started based on this technology. A specific feature of these projects is a special treatment of media documents as an organic part of acquiring information.

The repository is a common concept in information technologies, which can be used 1) for document and data storage; 2) for version control systems; and 3) for software package distribution. Document storage repositories used in addition to databases help save files effectively and access them.

In this , we consider repositories keeping a wide variety of multimedia documents and files in general. Universal solutions, such as the OS files systems used in Windows or Unix, as well as more specialized solutions for highload, such as those used in Hadoop technologies (HDFS) [4] or cloud-based like Amazon Web Services S3 [5] or Yandex Cloud Object Storage [6], lack some characteristics essential to access documents.

Specialized systems for photo and video management, such as Google Photos [7] or Apple Photos [8], are usually limited to certain multimedia types or to some automatic software usages. Moreover, they can sometimes drastically change the entire framework. This is what happened with Google Picasa some time ago: it was closed and migrated to Google Photos.

Another file repository system is Git LFS (Large File Storage) [9] as an extension to Git. Though it is one of the most open repositories to implement, it also lacks the features required, such as relations between database entries, media-files and media-preview-files.

In addition to solving the universal task of keeping files effectively, we also need to solve the file management and access task, which is a must for multimedia files. Media files should be delivered to the end-user in near-realtime and with fewer expenses (traffic wise or computational). It is done with specially prepared copies of different sizes. For example, a TIFF photo can be 50Mb and its high quality copy for fast viewing can be 200Kb. Even smaller copies of such images can be only several Kb. This helps us create in near-realtime photo compositions that include large-sized photos and many small-sized photos.

We also apply the distributed approach [10] in a broad sense. We need to create some blocks of information that can be potentially accessed and used in various places. Distribution in a broad sense means that blocks can be located both on one server and on several different servers in the different parts of the internet. Moreover, different information systems can use these blocks in different sets, independently including or excluding them as they desire.

We are implementing these blocks as cassettes. A cassette is a special kind of repository designed for the structured storage of multimedia files providing access to them. Besides storing files, it keeps the chunks of factographic graph information related to this file.

1. What is a cassette

A cassette has a hierarchical structure of directories, subdirectories and files. The cassette meta information and database are RDF/XML [11] files. A cassette has an identifier, which coincides with the name of its root directory. The cassette's identifier is not to be modified. The cassette's directory structure is as follows:

```
[cassette_id]
  cassette.finfo
  [originals]
    [0001]
      0001.ext
      0002.ext
      ...
    [0002]
      0001.ext
      0002.ext
      ...
  [documents]
    [small]
      [0001]
        0001.pre
        ...
    [medium]
      [0001]
    [normal]
      [0001]
```

```
[meta]
  cassette_id_current.fog
```

Here, the names in square brackets define directories and their names, while the names without brackets define files. Therefore, a cassette has multimedia original documents, their preview copies in small, medium, and normal sizes and a file with meta information in the "meta" folder. The root directory has a "cassette.finfo" file, which defines this directory as a cassette. This file also includes default parameters used for preview-files generation, the cassette structure version and other data.

.ext is a specific file extension for .jpg, .png, .tiff, .mp4, .avi, .mov, .mp3, .wav, .svg, .pdf and other files; .pre is an extension for preview files: .jpg is used for images, .mp4 and .webm for video and .mp3 for audio. There can be several previews of a specific size. For example, a video played in different browsers in the web can require more than one file, such as .mp4 and .webp. The main requirement is that all the preview-files for the same file in originals/DDDD should be in the respective documents/SIZE/DDDD folders with the same DDDD. Though the extensions can be different, the DDDD stays the same for each document.

File names are 4 digits (padded from left with 0 if necessary). There are no specific recommendations on how to generate them, but usually it starts with 0001 in each section. In future releases, digit file names can be changed to letter file names.

The division of all files into several subdirectories (sections), such as in Git [12]), is common practice used to facilitate access.

Information on every file is placed as a separate entry (element) in meta/cassette_id_current.fog. The relations of this document (file) with other elements are also placed there; the elements can be defined in this cassette or in other external cassettes.

The cassettes generated can then be published in various places on the internet. When needed, they can be joined together within a specific information system. This implements the distributed approach and provides high flexibility in working with the information field. For example, a cassette can have documents of an individual event, and these documents are associated with participants, organizations, and geographical places. Such a cassette can then be used in an information system dedicated to the events of a certain scope.

2. Data structure

In the factographic systems [13] created using the cassette approach, the factographic data are kept in the RDF/XML format files with .fog extension. The database consists of a set of entries about the entities of different types (persons, orgsystems, geosystems, documents) and relations between them.

Cassettes have dual identification of documents, which shows that a document is a logical entity with parameters (for example, the time it was created) and relations with the world (a document is part of a collection, has authors, etc). At the same time, a document is important because of the data it contains, or its content. An example is a book in a library. The book has a library card with parameters (name, year) and relations (author,

publisher, etc). Also, the book is an object itself, which needs to be held somewhere and access to it should be provided.

Logically, a document is identified by an id, which is a unique string of symbols, and all the links to this document are made using this id. "Physically," it is identified by a URI indicating in which cassette and in which place the document is held. If needed, a preview-file of this document can be obtained by slightly changing the URI.

2.1. Fog file structure

Fog file structure is described using an ontology [14]. Currently, the ontology_iis-v13 is used. In the future, the ontology can be extended or altered.

This is an example of a cassette fog file:

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF             xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://fogid.net/o/" owner="iis_1" prefix="cass01_" counter="1001">
  <cassette rdf:about="id1">
    <name>cass01</name>
    <cassetteUri>iiss://cass01@iis.nsk.su/meta</cassetteUri>
  </cassette>
  <collection-member rdf:about="id2">
    <in-collection rdf:resource="cassetterootcollection" />
    <collection-item rdf:resource="id1" />
  </collection-member>
  <photo-doc rdf:about="id3">
    <name>GoldenBalk</name>
    <from-date>2021-04-28T18:55:17</from-date>
    <uri>iiss://cass01@iis.nsk.su/0001/0001/0001</uri>
    <documenttype>image/jpeg</documenttype>
  </photo-doc>
  <collection-member rdf:about="id4">
    <in-collection rdf:resource="id1" />
    <collection-item rdf:resource="id3" />
  </collection-member>
  ...
</rdf:RDF>
```

The root element rdf:RDF has a namespace xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" required for RDF and a namespace xmlns=http://fogid.net/o/ used when there is no prefix. Then goes the attribute of the cassette's owner, which defines the user who is entitled to change the cassette's content and structure and modify the fog file. In addition, there are two auxiliary fields, a prefix and a counter, used to create unique identifiers. The next unique identifier is generated as a concatenation of the prefix and counter, after which the counter is incremented.

This file contains two recommended elements identifying the cassette itself and its relation (collection-member) to some root collection (cassetterootcollection). This is done to facilitate the inclusion of cassettes in the general database by referring to their root collection.

In this example of a fog-file, there are also two typical elements common for the cassettes keeping documents. We can see the definition of a photodocument and the

element stating that this photodocument is part of the cassette. The example has the sample ids: id1, … id4, though a global identifier can be formed using other approaches. In this case, however, an identifier defined by a prefix and a counter is preferable, so instead of id1 there could be cass01_1001, instead of id2 there could be cass01_1002, etc.

Today, we can use any globally unique identifiers including GUID. In the future, we plan to introduce a distributed cassette storage system and distributed database. Then we will need some syntax properties of an identifier to tell the system which cassette should be addressed at a given time.

The example does not show another common way of structuring information, namely, the creation of collections hierarchy. One can define a collection in the fog-file and connect it through collection-membership to the cassette or to another collection. Then the documented elements are placed not in the main cassette collection, but one of its hierarchical child collections. This gives us the opportunity to add directory structures with files to our database and retain in directories the information about its previous hierarchical structure.

2.2. Documents and their attributes

Usually, we assume documents to be multimedia, i.e. they can not only stored but also visualized or used in some other way. Thus every document in the database is accompanied with a file with the contents of this document. Its attributes are the name, creation date (from-date) and some others. Two of the system attributes, uri and documenttype, have a special function: uri indicates the place of the document storage, and documenttype shows its its mime-type.

The current ontology comprises the following four types of documents:

- document,
- photo-doc,
- video, and
- audio.

Document is the most general type; the others are more specialized mediatypes. They all have a common uri attribute with the following structure:

iiss://cassetteid@iis.nsk.su/0001/DDDD/DDDD.

We suppose that a change of the protocol (iiss) should entail a new structure.

documenttype should be a MIME-code for this document. The document type can be determined by the file extension or with the help of the basic analysis of the file contents. Currently, the following table is used:

```
static public DocType[] docTypes = new[] {
    new DocType(".png", "image/png", ONames.TagPhotodoc),
    new DocType(".tif", "image/tiff", ONames.TagPhotodoc),
    new DocType(".jpg", "image/jpeg", ONames.TagPhotodoc),
    new DocType(".gif", "image/gif", ONames.TagPhotodoc),
    new DocType(".mp3", "audio/mpeg", ONames.TagAudio),
    new DocType(".wav", "audio/x-wav", ONames.TagAudio),
    new DocType(".mpg", "video/mpeg", ONames.TagVideo),
    new DocType(".mpeg", "video/mpeg", ONames.TagVideo),
    new DocType(".avi", "video/x-msvideo", ONames.TagVideo),
    new DocType(".swf", "video/swf", ONames.TagVideo),
    new DocType(".flv", "video/flv", ONames.TagVideo),
    new DocType(".wmv", "video/wmv", ONames.TagVideo),
```

```
new DocType(".mp4", "video/mp4", ONames.TagVideo),
new DocType(".mov", "video/mov", ONames.TagVideo),
new DocType(".mts", "video/mts", ONames.TagVideo),
new DocType(".txt", "application/text", ONames.TagDocument),
new DocType(".doc", "application/msword", ONames.TagDocument),
new DocType(".rtf", "application/rtf", ONames.TagDocument),
new DocType(".pdf", "application/pdf", ONames.TagDocument),
new DocType(".xls", "application/excel", ONames.TagDocument),
new DocType(".xml", "application/xml", ONames.TagDocument),
new DocType(".fog", "application/fog", ONames.TagDocument),
new DocType(".htm", "application/html", ONames.TagDocument),
new DocType(".html", "application/html", ONames.TagDocument),
```

This table uses some of the general types found at https://en.wikipedia.org/wiki/Media_type. Also, there can be new specialized types, such as .fog.

The most common way of using documenttype is as follows. When a file is added to the database, we determine its document type and specify it in its document entry. To generate preview-files, the specific types of documents are used for their creation and subsequent viewing/playing. For photo documents, the preview format is .jpg; for video, .mp4 and .webm; and for audio, .mp3. Also, documenttype is used for an HTTP document output in the Content-type field.

There can also be another special element for the documents in the document element in the database. This element stores document meta information. In the current ontology, it is docmetainfo, which is key-value pairs storing media details such as width, height, fps, EXIF and others.

3. Features of the realization

3.1. Acquiring information when storing a new document

A document is information. Information about a document is information about information, or meta information. As mentioned earlier, the fields important for a database are the name, creation date, last modification date and various multimedia parameters, such as width, height, aspect ratio, fps, etc. This information needs to be acquired somewhere.

Some of the information can be found in the file of the document itself. Sometimes, the acquisition of meta information is encouraged by file standards such as EXIF. Information can also be retrieved from open source libraries, though this only concerns digital photos and videos. The documents digitized by scanning or through composition conversion do not display the correct creation and modification dates. Moreover, some file types do not have meta information at all.

The other part of information is meta information about the file, which is sent to be stored and processed. However, some of this information can be partially lost in the process as the filesystems or archiving software can alter the original file creation and modification dates,

There is even more uncertainty with video documents. Though there have been attempts to develop a unified standard for video factographical information (such as the

date of shooting or date of composition creation), this has not been achieved yet. In this situation, it makes sense to rely on some widely used open source video processing tools (like ffmpeg [15]).

3.2. Used tools for information retrieval and document processing

The following tools were chosen based on their applicability, user, and expert assessment, as well as on personal experience. All of the software is open source and working in Windows and Linux.

Meta information can be retrieved using MediaInfo, which helps to obtain such information from video and audio files. Another advantage of this application is that it can output not only plain text but XML or JSON formats. Here is an example of Mediainfo output:

```
<?xml version="1.0" encoding="UTF-8"?>
<Mediainfo version="0.7.35">
  <File>
    <track type="General">
      <Complete_name>F:\Altai\AltayDzhozator.wmv</Complete_name>
      <Format>Windows Media</Format>
      <File_size>653 MiB</File_size>
      <Duration>1h 1mn</Duration>
    </track>
    <track type="Video">
      <Duration>1h 1mn</Duration>
      <Bit_rate>1 356 Kbps</Bit_rate>
      <Width>720 pixels</Width>
      <Height>576 pixels</Height>
      <Display_aspect_ratio>5:4</Display_aspect_ratio>
      <Frame_rate>25.000 fps</Frame_rate>
    </track>
    <track type="Audio">
      <Sampling_rate>44.1 KHz</Sampling_rate>
      <Bit_depth>16 bits</Bit_depth>
      <Stream_size>56.1 MiB (9%)</Stream_size>
    </track>
  </File>
</Mediainfo>
```

The meta information fields we can see above are easily recognizable. For our purposes, we generally need width, height or aspect ratio; for demonstration purposes, width/height and duration are essential; for download, file size counts, and so on.

ffmpeg is another tool that helps process video and audio files. It operates on a wide variety of file formats and processing parameters. This software can use various codecs, vary the compression rates of video and audio, change dimensions, fps rate and perform many other tasks. It has a large documentation manual, but the basic usage is:

```
ffmpeg -i {0} -y -vcodec libx264 -b 4050K -ar 22050 -s 1440x1080 {1}.mp4
```

Here we have a file input {0}; libx264 is one of the video codecs used for output; -b argument states bitrate for the new video stream; -ar argument manages audio bitrate; -s (size) states the output size; and lastly we have the output file name.

Importantly, the output size aspect ratio needs to be controlled; otherwise, the picture can be distorted. As mentioned earlier, the video file content dimensions can be acquired using this or other tools (like MediaInfo).

Similar actions can be executed for images. Since image processing is a common software development task, as a rule, every programming language has appropriate tools. Images can be loaded into RAM, processed and saved in various formats. There are libraries for EXIF information management, though EXIF is not used in all of the formats.

For bitmap image conversion and preview-file creation, vips application can be used. Here is an example:

```
vips resize map.jpg out.jpg 0,078125
```

Another widely used cross-platform library for image processing is ImageMagick.

Conclusion

This paper introduces a cassette structure as a specialized repository for multimedia documents storage and as a database of entities associated with documents. We present its description, file and data structure, and the specifics of its realization.

When published on the internet, cassettes can be used by various information systems needing this information block. This mechanism gives us the opportunity of a flexible system configuration and a realization of distributed information systems.

This approach was implemented in the information systems created by the A.P. Ershov Institute of Informatics Systems SB RAS, namely:

- SB RAS Photo Archive http://soran1957.ru, and
- Open archive of SB RAS http://odasib.ru.

It was also realized in the following information systems:

- Archive of Summer school for youth developers,
- Roll-call of Mathematics and Mechanics of Novosibirsk State University, and
- GLOBUS alumni community platform https://globusnsu.ru.

References

[1] Marchuk, A.G., Kraineva, I.A., Marchuk, P.A. Technologies of history factography: Digital photo archive SB RAS // Proc. All-Russia science and practical conference “The museums of Siberia integration into regional social and cultural space and world museum community”, Ulan-Ude, September 6-9 2009. - Ulan-Ude: BSC SB RAS Publisher, 2009.

[2] Marchuk, A.G., Marchuk, P.A. Archive factographic system // Digital Libraries: Advanced Methods and Technologies, Proc. XI All-Russia science conference RCDL 2009. - Petrozavodsk, 2009.

[3] Marchuk, A.G., Marchuk, P.A. The features of digital libraries with linked content // Digital Libraries: Advanced Methods and Technologies, Proc. XII all-

Russian science conference RCDL-2010, Kazan, Russia, October 13–17, 2010. — Kazan, 2010. — P. 19–23.

[4] HDFS Architecture Guide https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html.

[5] AWS S3. - https://aws.amazon.com/ru/s3./

[6] Yandex Cloud Object Storage. - https://cloud.yandex.ru/services/storage.

[7] Google Photos Library API overview. - https://developers.google.com/photos/library/guides/overview.

[8] Apple PhotoKit. - https://developer.apple.com/documentation/photokit.

[9] Git Large File Storage. - https://git-lfs.github.com/.