D5.1: Case study scenario definition

Mitja Jermol (JSI), Matjaž Rihtar (JSI),
Carlos Turró Ribalta (UPVLC), Davor Orlič (K4A)

Distribution: Public

*trans***Lectures**

Transcription and Translation of Video Lectures

ICT Project 287755 Deliverable D5.1

Project funded by the European Community
under the Seventh Framework Programme for
Research and Technological Development.

Project ref no.	ICT-287755
Project acronym trans Lectures	*trans***Lectures**
Project full title	Transcription and Translation of Video Lectures
Instrument	STREP
Thematic Priority	ICT-2011.4.2 Language Technologies
Start date / duration	1. November 2011 / 36 Months

Distribution	Public
Contractual date of delivery	April 30, 2012
Actual date of delivery	May 31, 2012
Date of last update	May 31, 2012
Deliverable number	D5.1
Deliverable title	Case study scenario definition
Type	Report
Status & version	Final v1.0
Number of pages	31
Contributing WP(s)	WP5
WP / Task responsible	UPVLC, XRCE, JSI, K4A, RWTH, EML, DDS
Other contributors	
Internal reviewer	Alfons Juan
EC project officer	Susan Fraser
Keywords	

The partners in *trans***Lectures** are:

Universitat Politècnica de València (UPVLC)
XEROX Research Center Europe (XRCE)
Josef Stefan Institute (JSI)
Knowledge for All Foundation (K4A)
RWTH Aachen University (RWTH)
European Media Laboratory GmbH (EML)
Deluxe Digital Studios Limited (DDS)

For copies of reports, updates on project activities and other *trans***Lectures** related information, contact:

The *trans***Lectures** Project Coordinator
Alfons Juan, Universitat Politècnica de València
Camí de Vera s/n, 46018 València, Spain
ajuan@dsic.upv.es
Phone +34 699-307-095, Fax +34 963-877-359

Copies of reports and other material can also be accessed via the project's homepage:
http://www.translectures.eu

Executive Summary

This document contains details about WP5 task **5.1 Case study analysis and requirements definition** together with the implementation plans.

Table of Contents

1. Introduction

transLectures will provide technologies and solutions for the two case studies, VideoLectures.NET repository of video lectures developed and run by JSI and Polimedia repository of videotaped lectures that is being developed and operated by UPV. In addition, solutions developed in transLectures would need to be prepared in a way that will be used by the Matterhorn video capturing, processing and editing pipeline. Matterhorn is an open source integrated framework for academic video manipulation that is being developed by the consortium of 13 academic and research institutions in the frame of the Opencast community.

This deliverable provides the detailed description of the two cases studies and the Matterhorn framework with the use and feel and technical specifications. It is aimed at defining requirements by the site operators (VideoLectures, Polimedia and Matterhorn) that will serve as an entry point to the development activities in other research and technical work packages.

Requirements predefined in this document span over several levels. The architectural level defines two distinct architecture scenarios of implementing transLectures results in the case studies and in the Matterhorn framework. The first scenario foresees transLectures modules integrated directly in the case studies and providing SaaS functionality for Matterhorn. The second scenario implements the complete software service architecture in SaaS style.

The next level requirements are generic software requirements that are presenting the needs from the two case studies in the case transLectures services will be implemented in live applications. Here the focus is in scalability, services response times and accuracy of the in transcripted texts and translated texts in defined languages.

Based on the two architecture scenarios and general requirements for usability and scalability, the technical requirements have been defined and described in detail.

The final set of requirements deals with the look and feel level. Here the two case studies proposed the look and functionalities of transLectures services that will be included in user interfaces. This is why there a series of mock-ups are presented.

transLectures is a research project which means that some of the requirements that are answering real-life installation in VideoLectures or Polimedia could not be fully met in the first run. Only after extensive prototype testing in the real-life conditions the services will then be officially published at both sides. Certainly the requirements will be met for the prototype installations.

This document is structured in several chapters. It starts with the introduction (this one). The second chapter provides detailed description of VideoLectures service from the technical, workflow and usability perspectives. In the same way the third chapter provides detailed description of Polimedia case study. Chapter 4 deals with the user interfaces that include the user correction. Here several mockups are presented and discussed. Chapter 5 provides detailed information about Matterhorn with details on the workflow, current status and the

technical architecture. Document then concludes with Chapter 6 presenting general requirements that are spanning over the two case studies and Matterhorn.

2. VideoLectures

2.1 VideoLectures description

With the educational developments and rapid technological advances in recording, hosting and production of digital video content, a European distributed long-term living and extremely important video digital library has emerged - VideoLectures.NET (http://videolectures.net). It is a free and open access educational video lectures digital library currently offering more than 15000 video lectures. The lectures are given by distinguished scholars and scientists at some of the worlds' most famous universities and at important and prominent conferences, summer schools, workshops and science promotional events from many fields of Science. The portal is aimed at promoting science, exchanging ideas and fostering knowledge sharing by providing high quality didactic contents not only to the scientific community but also to the general public. In order to fulfill this aim the following challenges needs to be addressed:

- The quality of service. By offering innovative browsing and viewing applications, but in particular providing responsive service that is directly related to the problem of what and how to store materials.
- The quality of materials. The problem ranges from the quality of the lecture itself, technical quality and distribution channel/method quality.
- Long term content/context preservation. VideoLectures is committed to maintain the fixed URL preferably forever but at least as long as the owner (author, organization, event organizer) of the lecture is removing the lecture.

VideoLectures gathers different types of digital objects such as PowerPoint presentation, emails, word and pdf documents, rss feeds, social networking data over Twitter and Facebook and more importantly multimedia objects containing videos, audio, pictures. These digital objects are reusable and do not get discarded. Most of the training materials are being developed within the FP5, FP6, and FP7 European Framework Programs, where VideoLectures is being used as an educational platform for several past and ongoing EU funded research projects such as PASCAL NoE, Ecolead IP, SEKT IP, ACTIVE IP, COIN IP, E4 STREP, Euridice IP, NeOn IP, PetaMedia NoE, SMART STREP, Tool East STREP and many others. This can be a hint to the diversity of the video content presented and especially to the quantity of it.

At the moment VideoLectures have

- 13000+ videos in the following languages:
  - 11263 in English
  - 1366 in Slovenian
  - 54 in French
  - 46 in Croatian
  - 20 in Dutch

    - 18 in Spanish
    - 14 in German
- 1058 videos with manually synced slides
- 172 videos with subtitles

As a result of transLectures project we would like to have:

- the ability to transcript videos in the following original languages:
  English, Slovenian, French, German and Spanish
  for showing the subtitles and enhanced search capabilities
- the ability to translate these transcriptions to the following languages:
  minimum: English, Slovenian
  optimal: all five of the above listed languages

2.2 Current workflow in VideoLectures

VideoLectures makes its video production entirely with its own equipment, cameramen and video editors. The following is a detailed description of current workflow in VideoLectures:

Pre-filming and filming

The cameraman or stationary camera needs to have a filming position in the event venue that enables it to be positioned for direct capture of the speaker and the slide presentation. Ideally, the cameraman or stationary camera occasionally makes a close-up of the speaker and the slides separately. The reason for such a filming technique is the VideoLectures' post production and a wish to achieve an optimal viewing experience for our viewers. It is also required to have the appropriate lighting conditions in the venue as filming in a completely dark room produces unusable footage.

Inserting video cassettes' information into the admin

Cameramen are special registered users of VideoLectures and have access to non-public data. After logging in to the VideoLectures' site the name of the user displays at the right hand top side of the main page. User can then make a switch from the Preview mode to the Edit mode. For each event the cameraman enters the data about new tapes in the Tapes section. Here he adds new tapes or video files associated with the previously or newly filmed event. Details of inserted information are the following:

- Recording/tape number
  Number of consecutive cassettes or video files.
- Recording date/time
  Date and time of the recording
- Approximate duration
  Clip duration, the real duration or how long one lecture was held. The entry is accurate to the minute, which means that the seconds are rounded up or down.
- Tape offset
  Starting time - when the lecture begins on the video cassette (there can be more than one lecture stored on one tape).

- Owner
  The cameraman or the username of the one who is making the entry.
- Camera type
  Here the camera type (or "Other" for video files) and the recording format is selected.
- Comments
  A detailed description is of great help for the video production and the publication teams and for the process of processing video and synchronization with slides.

Transition of raw files to video machines and edit

After returning home from an event each cameraman downloads the files from the cassette or card he used in the process of filming.

Postproduction

This encompasses the color and sound correction, titling and correction of graphics. The files are then edited at the beginning and end, with no major editorial cuttings as not to affect the content of the lecture.

Conversion of raw input files into other streaming formats

VideoLectures currently stores videos in 3 streaming formats:

- FLV Flv Format
  (Compatible with Flash Player 7 and higher), Video codec: VP6, Video Bit Rate: 300 - 500 Kbps, Frame Size: 512x288, Frame Rate: 25; Audio Bit Rate: MP3, 128 Kbps, 44.1 KHz, stereo, CBR
- WMV Windows Media Format
  (PAL Widescreen Source for High Quality Download), Video Codec: Windows Media Video 9, Video Bit Rate: 300 - 500 Kbps, Frame Size: 512x288, Frame Rate: 25; Audio Format: WMA v2, 128 Kbps, 48 KHz, stereo, VBR
- MP4 H.264 Format
  (PAL DV Widescreen High Quality), Video Codec: H.264 (AVC), Profile: Main@3.1, Video Bit Rate: 0.84-1.45 max., Frame size: 1024x576, Frame rate: 25; Audio Format: AAC LC, 48 KHz, stereo, VBR

Raw input files are manually converted to the above formats via professional video editing software.

Creation of lectures in the admin

From the Tapes section we proceed to create lectures which functions as a shell for a video. Events are pre-created and names of authors for each video and their relevant institutions are entered. Lectures are manually added into categories. After this is set up, video editors upload the converted video files to the server which are then installed by the content editors via the website. After the event is completed, a DVD is created and raw files are stored in backup.

Synchronization of slides

If slides of the lecture are also available, the synchronization of the slides with the video is done for an easier and more effective learning experience. Our editors have access to the online synchronization interface (we have a built-in web application for synchronizing video

and slides). This is done manually. The slides are cropped (for better resolution), zipped and loaded to our server where the application automatically unpacks them into JPGs. The editors then import them to the lecture. The next step is matching the time and slides on the video.

Transcription and Translation

transLectures functionality (transcription and/or translation) will be chosen as a last step in the current VideoLectures workflow. After the author checks the video, it can happen that the video is cut on some places or shortened, so new synchronization with the slides, if they are present, is needed. To avoid duplicate work, transcription and/or translation will happen only after all previous steps are finalized.
Editor could choose the process to be:

- Manual with selection
  Editor will choose the functionality (either transcription or translation or both and languages to translate into) and send the video for processing. He will then wait for the (remote) process to finish and return results.
- Semi-automatic
  Video will be sent for transcription and/or translation with the click on the button, after which default action and languages will be chosen and processed. The results will be stored automatically within lecture, as the process is finished.

Results returned from transcription and/or translation process will be in TTML/DFXP format, which will have to be converted to a format suitable for VideoLectures player (currently SubRip Text i.e. .srt format). This will be done automatically on results return.

Interactive correction of Transcription and Translation

This action will be enabled only for some of the users, typically the author of the video and video editors. Person doing the interactive correction will click on the link, which will take him to another page, where the GUI for interactive correction will be presented. For a detailed discussion about types of the users and the process of interactive correction see chapter “Interactive Correction (mockup)”.

2.3 Use scenario (mockup)

The design of current VideoLectures Web pages won’t be changed much after inclusion of the results from transcription/translation. The following changes will be visible:

- Player will get the [CC] button in the lower right corner. When selecting the [CC] button all available subtitles in original and translated languages will be shown. The user will select the language and subtitles will be shown in the lower part of the video in the player (overlaid on top of the video).
- For special users (author, video editors …) an additional link will be shown, which will take the user to the GUI for interactive correction of transcription and/or translation. When the user finishes with the correction, he will be returned to the base page of the lecture.

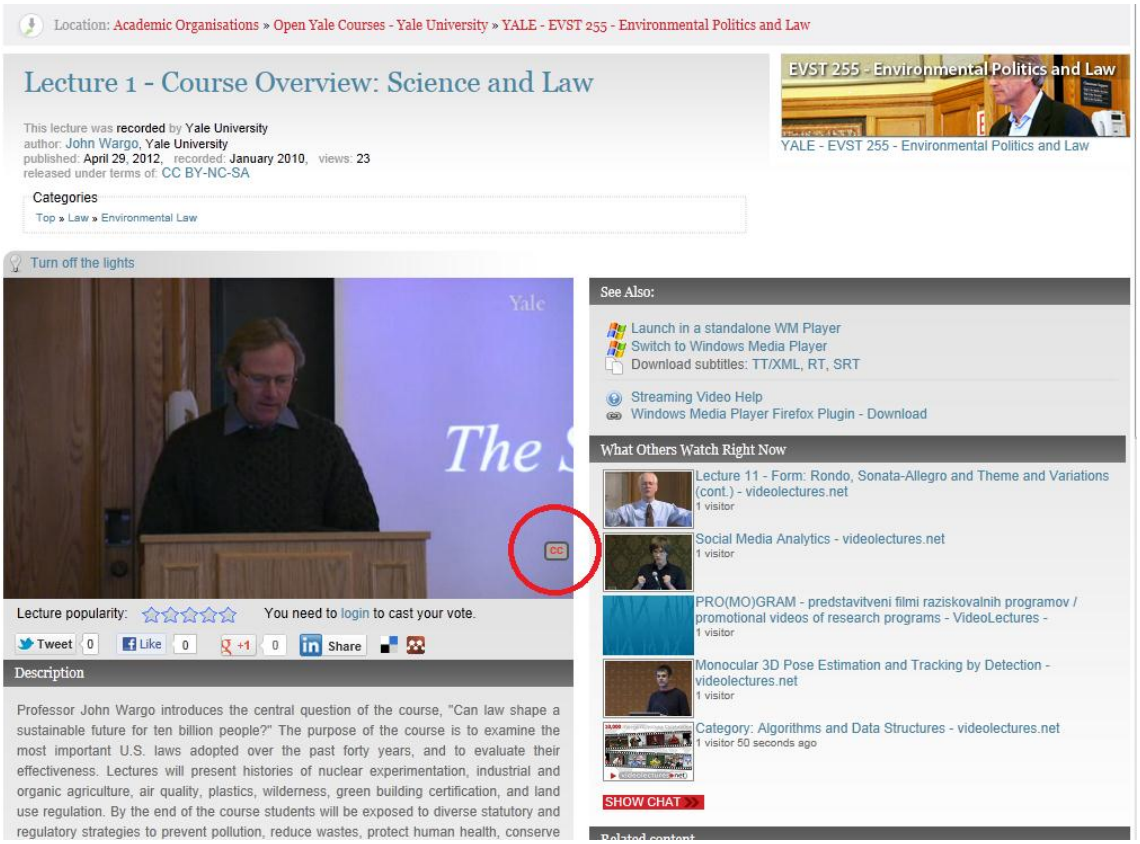


Figure 1: CC button for enabling subtitles/selecting languages

Future work based on available transcriptions will include enhanced Search capabilities like:

- Search for keywords inside videos and showing the resulting list of videos (with links to the moments in videos where searched keywords were spoken)
- Showing related videos
- Better ranking of videos
- Better classification of videos
- Assessment of video's difficulty

2.4 Technical requirements

VideoLectures will use a dedicated computer on a local network to optimize access to transcription and translation engine and to the results of this process, which will be updated from time to time. Typically, to get the transcription for a new video, whole video or at least audio tracks of the video will be sent to the dedicated computer via a Web service. This would impose a lot of network traffic as videos in better quality can be quite large (up to several GBs). Also, results from the transcription and/or translation will be updated from time to time as authors or editors fix errors and send the results back into processing.

Our computer for transLectures engine and results storage will be a Linux server running CentOS 6.x (or possibly Ubuntu 12.04 LTS) operating system with the following hardware specifications:

- 2x Xeon E5-26xx processors
- 32 GB DDR3 RAM
- 3 TB RAID disk array
- 2x 1 Gbps LAN

3. Polimedia

3.1 Polimedia description

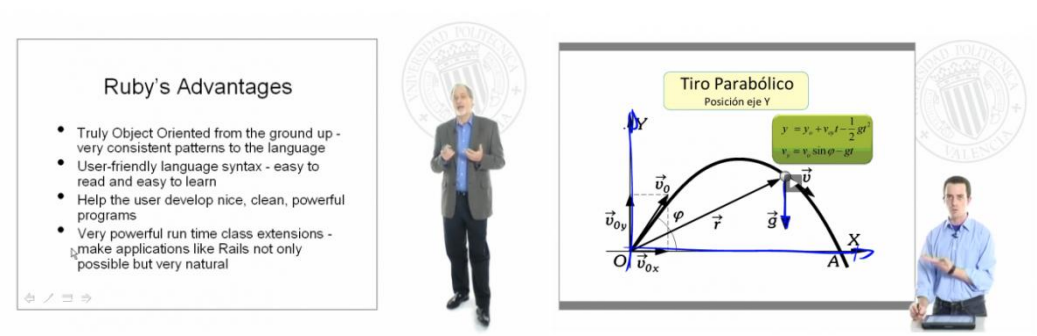
The Polimedia system belongs to asynchronous training e-learning category using the Learning Objects (LO) paradigm. So, the Polimedia system is designed to create small pieces of video content designed specifically for e-learning.

When creating a Polimedia, any teacher has to deal with some requirements in order to achieve a successful work. The main ones are:

- Structure the course in learning objects
- Use a clear, simple language, close to the listener.
- Take care as much the verbal communication as the non-verbal communication.
- Have a maximum length of around 10 minutes.

Any lecture recording system aims to produce a high quality and visually appealing view of the teacher, the computer screen or the blackboard, and the audio recording. That view has to be clear, smooth and must not distract the student from the learning process. With these issues in mind we have stacked horizontally the slide and the presenter, composing a 720p frame.

If we show only the bust of the presenter we allow a little overlap between the presenter and the slide, maintaining the frame size. It is only noted when the presenter extends his right arm.



Polimedia full shot Polimedia half shot

Figure 2: Polimedia video composition

We have designed the process carefully to achieve both a high rate of production and to get a final quality is comparable to a TV production, but with much lower cost. A key piece in this setup is the production studio.

A Polimedia production studio is a 4x4 meters room in which we deploy a camera, two PCs, a pocket microphone, lights and some A/V equipment, including a video mixer, and an audio noise gate. It is worth noting that the use of lights in such a reduced set allows getting a sharper image easier than in a lecture recording hall. The cost of this studio is around 20.000 €.

When a teacher wants to record a Polimedia, he/she arrives to the studio with the slides and shows a presentation directly on the studio. There we record both the screen and the video from the teacher in two different streams.

After that we have a raw preview of the Polimedia content by putting the screen and the teacher video side-by-side, and on that screen the teacher can review the material he has just created. If everything is OK we start a process for cropping, joining with a little overlap and encoding, in order to have an .mp4 file with h.264 encoding, suitable for distributing and archiving.

Then, the cropping, joining and encoding process is fully automated using avisynth and ffmpeg, so the teacher can see a preview of the recording immediately and the resulting file is ready in a few minutes. As stated, the final file is a .mp4 file at 500kbps, that we distribute by streaming through a flash media server. Also we encode an mp4 designed for mobile devices, like IPhones.

Actually there are 8 Polimedia production studios and about 5000 recordings

3.2 Current workflow in Polimedia

From teacher's point of view, recording a Polimedia object is as easy as arrive with a PowerPoint file, a laptop, or even a URL to our recording studio. There are two screens, on at the front of the teacher, and one at the right. In such position he or she will record the lesson and both streams will be recorded. Here you can see the set up for the studio and an image of a live recording.

Figure 3: Polimedia studio recording

After that the content is uploaded to the Polimedia server, where there is a set of processes that compose the final video, overlay the logos and finally deploy the final .mp4 file to the content server.

The teacher is then informed through e-mail that the video is ready to distribute.

3.3 Use scenario (Mockup)

The design of current Polimedia Web pages won't be changed much after inclusion of the results from transcription/translation. The following changes will be visible:

- Player will get the [CC] button in the lower right corner. When selecting the [CC] button all available subtitles in original and translated languages will be shown. The user will select the language and subtitles will be shown in the lower part of the video in the player (overlaid on top of the video).

- For special users (author, video editors …) an additional link will be shown, which will take the user to the GUI for interactive correction of transcription and/or translation. When the user finishes with the correction, he will be returned to the base page of the lecture.

3.4 Technical requirements

transLectures functionality (transcription and/or translation) will be chosen as a last step in the current Polimedia workflow. After the video is composed, the video will be sent for transcription and/or translation with the click on the button, after which default action and languages will be chosen and processed. The results will be stored automatically within lecture, as the process is finished.

Results returned from transcription and/or translation process will be in TTML/DFXP format (see chapter 6), which will be a format suitable for Polimedia player.

Interactive correction of Transcription and Translation

This action will be enabled only for some of the users, typically the author of the video and video editors. Person doing the interactive correction will click on the link, which will take him to another page, where the GUI for interactive correction will be presented. For a detailed discussion about types of the users and the process of interactive correction see chapter "Interactive Correction (mockup)".

4. Interactive Correction (Mockup)

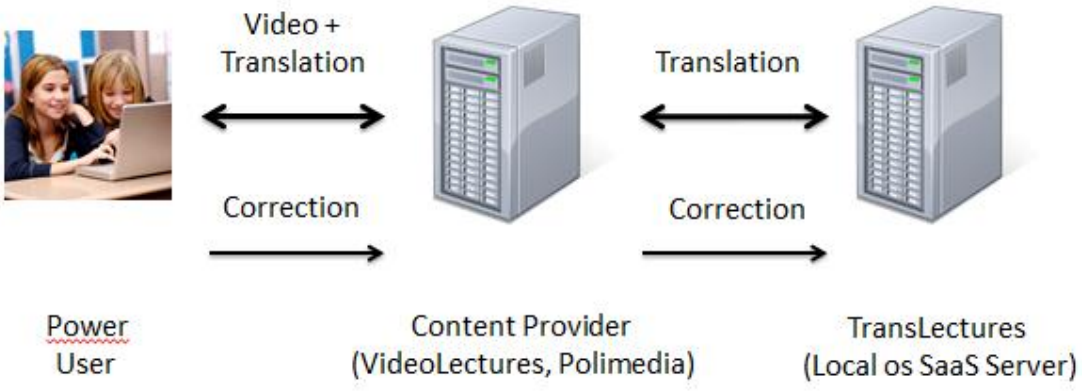


Figure 4: Overview of transLectures system

In order to implement the technical requirements described previously we will have to define a set of user's roles. Then we will develop an interface allowing interaction for them. While we will define now a complete list of roles it is likely that some of them will have the same interface, differing only in their access rights

The roles that we will use in transLectures are:

- **Viewer:** a viewer is just a user looking at a particular translation. So he will have a simple set of features, allowing him to view a synchronized subtitles track if the confidence level of that video is above the level set by the editor or the author. The player should allow a user to become collaborative user.
- **Collaborative viewer:** users with that role will have access to an advanced set of features from the transLectures engine; for them the interface will provide a confidence level for the overall transcription/translation, and also will provide alternate translations for difficult parts. The confidence level required to display a translation for a collaborative user will be much lower than the required for a standard viewer.
  It is expected that collaborative users can be asked through the interface for simple tasks, like transcribing some parts of the video.
  Input from collaborative users will add information to the translation, but will not replace the content, and will not trigger an update on the acoustic model.
- **Expert (e.g. professional translators):** an expert is a collaborative viewer with expertise in that field. So, experts can replace transcriptions and translations in parts of the video. Also an input from an expert will trigger an adjustment on the acoustic model.
- **Author:** An author is the owner of a video, and will be always taken as an expert for his uploads. Also he can decide if subtitles can be displayed by viewers, based on the average confidence level.

- **Editor:** An editor can set a confidence level for a whole site.

As stated previously, for the point we are considering now (interface design) we will have two kinds of roles: standard users and power users. For standard users we will use the usual interface of the content providers with some minor modifications (we will see later), while for power users we will have a unified advanced interface.

These mockups are tentative: they are showing our current designs, but they could be changed for the design reasons during the development of the project.

4.1 Basic Mockup for VideoLectures

Here is a basic mockup about how VideoLectures will look like for a standard user (see subtitles on the right and the next screen with detailed subtitles):

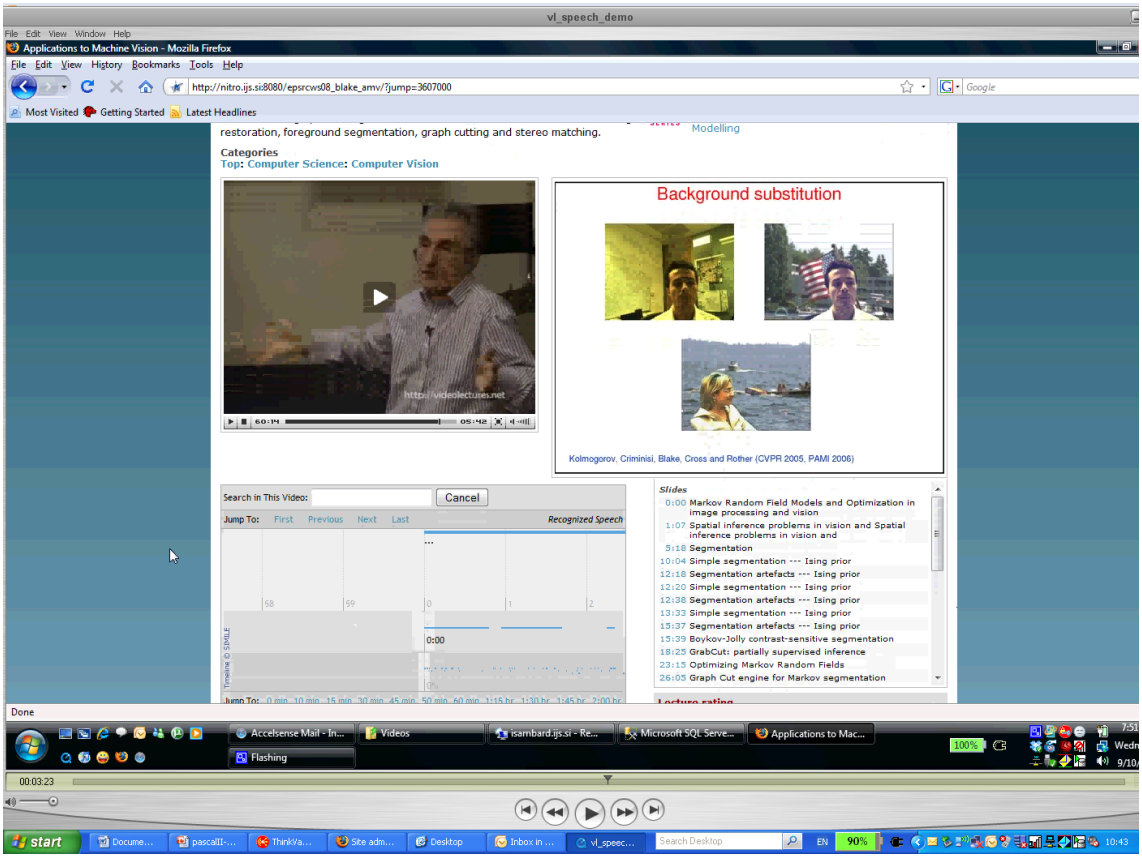


Figure 5: Basic VideoLectures mockup with subtitles and timeline

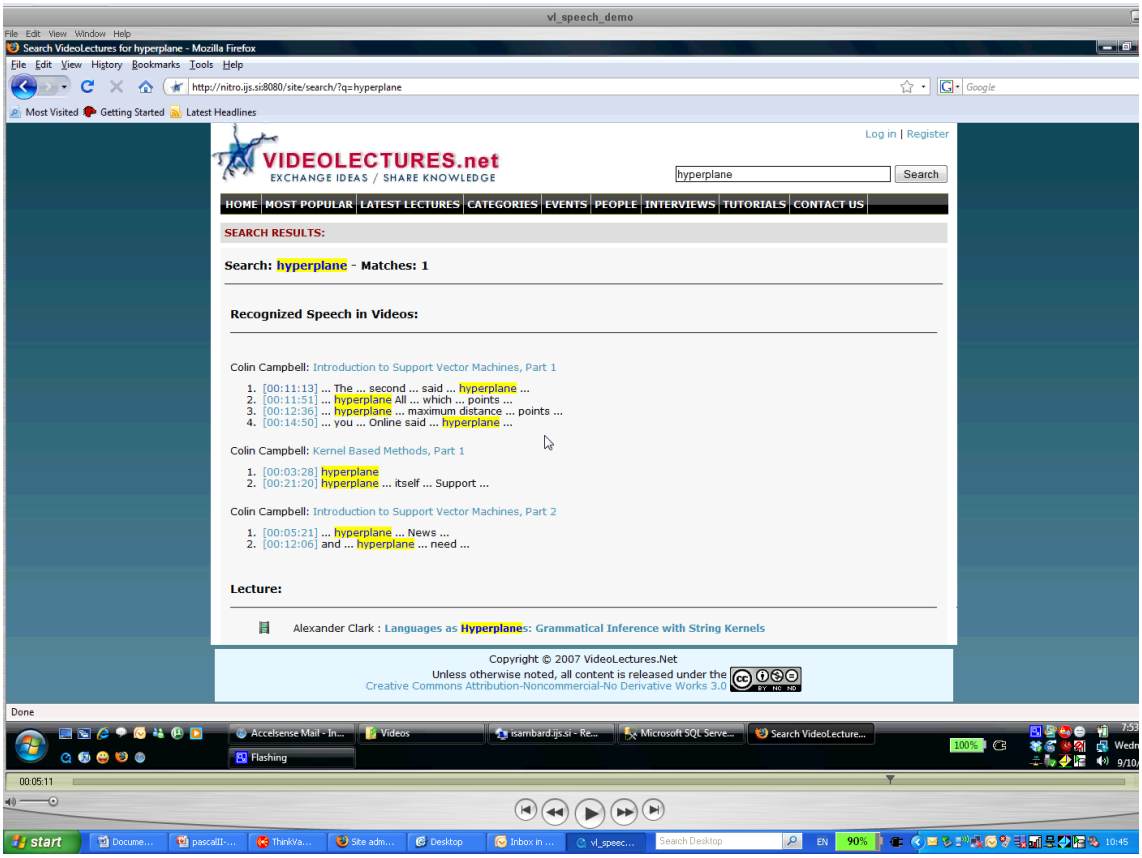


Figure 6: Output of enhanced search for keywords in transcription

4.2 Basic Mockup for Polimedia

Here is a basic mockup about how Polimedia will look like for a standard user (see subtitles for transcription and translation on the right):

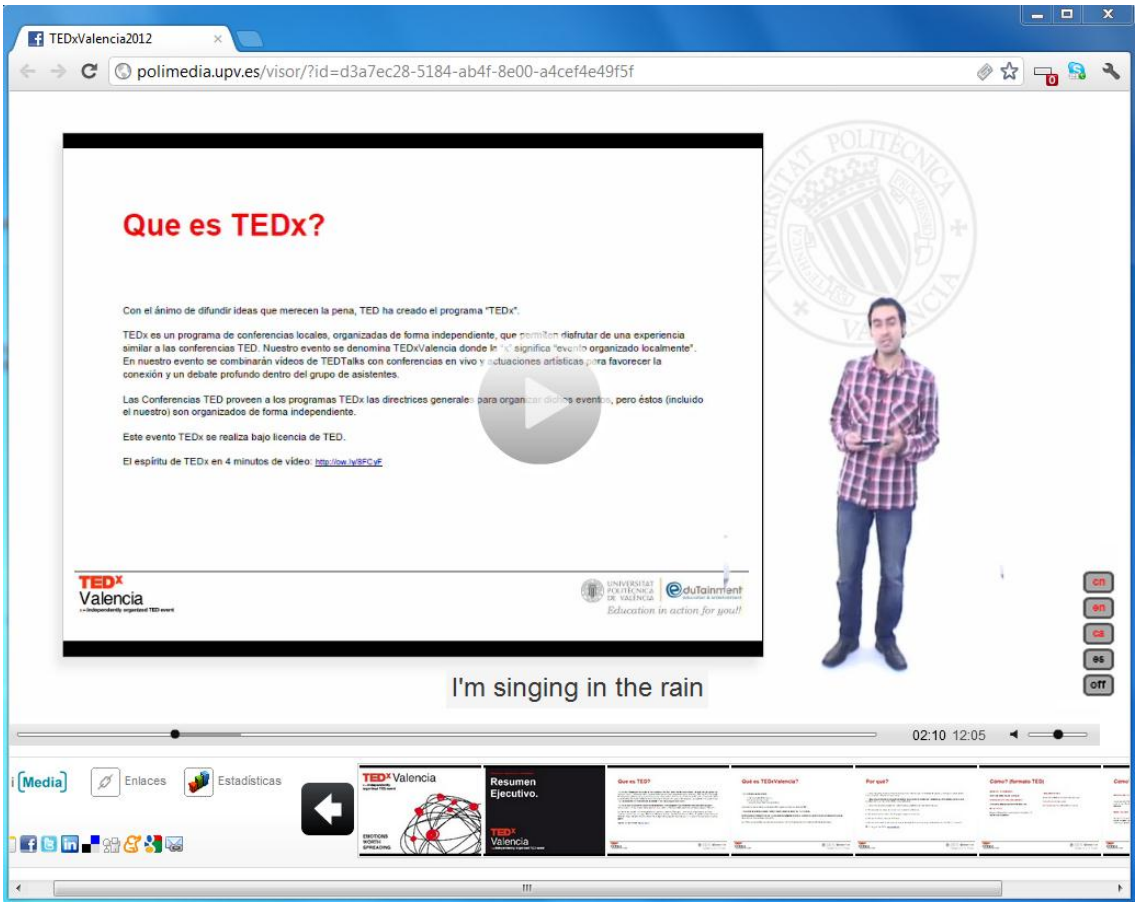


Figure 7: Basic mockup of Polimedia player with subtitles

4.3 Mockup for power users

For power users we will develop an HTML5 advanced editor converting the needs of interaction between those users and the system with the following features:

- Play/pause/jump video and subtitles
- Synchronized transcription/translation playing
- Confidence level for the video
- Confidence level for the sentence (if available)
- Alternate text (if available)
- Ability to modify a sentence
- Ability to replace a word in the whole transcription/translation

These features will be also available through the API described in Section 6.

Figure 8: Mockup of GUI for editing the transcription/translation for power users

5. Matterhorn integration requirements

5.1 Matterhorn description

Opencast Matterhorn project provides a framework of media services for the management of educational audio and video content. Institutions will use Matterhorn to produce lecture recordings, manage existing video, serve designated distribution channels, and provide user interfaces to engage students with educational videos. As a framework it is highly configurable to meet individual institutional needs. Matterhorn's architectural, design and software principles allow it to integrate different technologies.

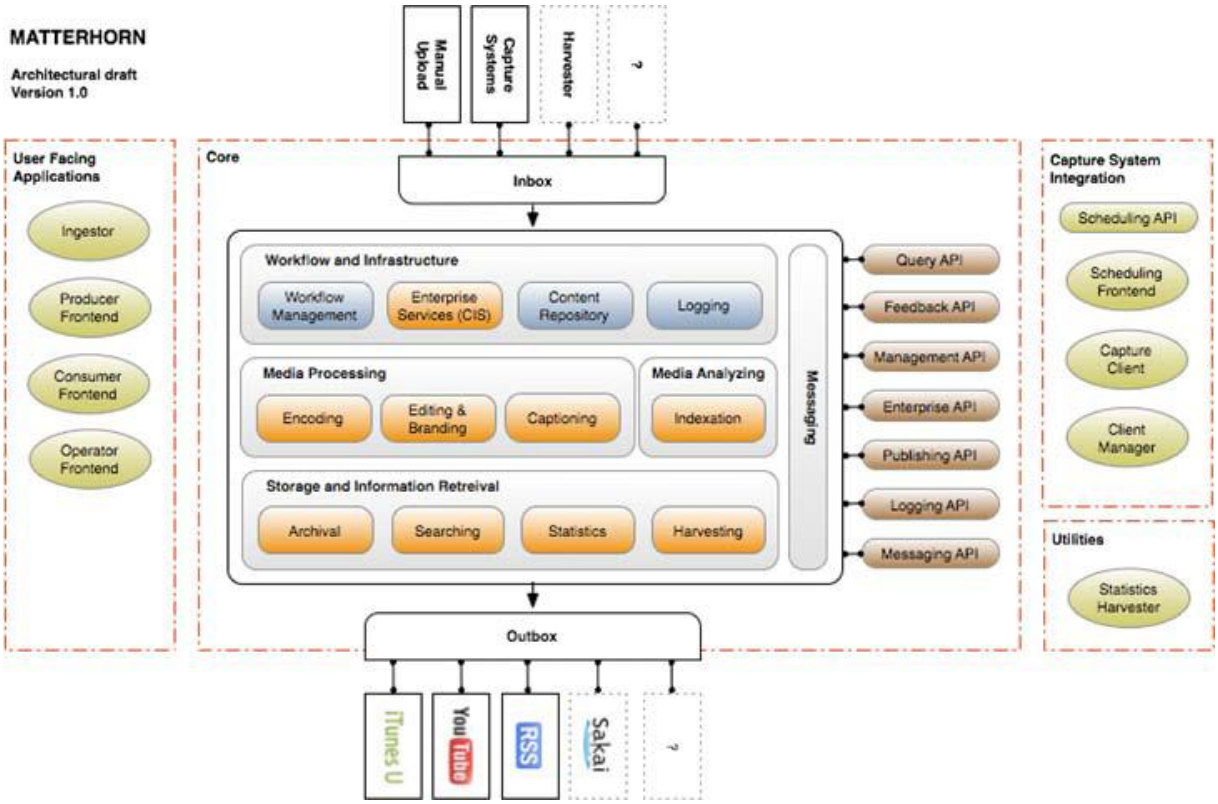


Figure 9: Matterhorn architecture

Underlying technologies

Matterhorn is an open source; this means that the product is fully based on open source products. The members of the Opencast Community have selected Java as programming language to create the necessary applications and a SOA infrastructure. The overall application design is highly modularized and relies on the OSGI (dynamic module system for Java) technology. The OSGI service platform provides a standardized, component-oriented computing environment for cooperating network services. Matterhorn is as flexible and open as possible and further extensions should not increase the overall complexity of building, maintaining and deploying the final product. To minimize the coupling of the components and 3rd party products in the Matterhorn system, the OSGI technology provides a service-oriented architecture that enables the system to dynamically discover services for collaboration. Matterhorn uses the Apache Felix[1] implementation of the OSGI R4 Service Platform[2] to create the modular and extensible application. Matterhorn provides getting started guides and additional information for developers on the public project wiki-page[3].

Schedule/Prepare & Capture

The recording process begins with determining what is to be recorded, where and in what form. Campus data will be integrated by the universities' IT departments. For this purpose, Matterhorn is open to both the learning management systems and administrative data bases. Syllabi, lecture and room timetables do not only provide the basic information to answer the

1 http://felix.apache.org/site/index.html
2 http://www.osgi.org/Main/HomePage
3 Opencast documentation: http://wiki.opencastproject.org

question raised above, but in an ideal case, most of the metadata related to the recording (lecturer, title, summary, language, etc.) as well. Recording devices are then scheduled to automatically record in lecture hall X.26, every Tuesday from 10:00 c.t. to 12:00, the lecture on "XYZ" by Prof. ABC.

Process

At the end of the recording the tracks are sent to an "inbox" to be processed. The inbox also serves as "ingest" for other video objects to be integrated in the subsequent workflows of Matterhorn. At most institutions, objects such as self-produced podcasts, image films or digitalized historic recordings would constitute only a small part of the whole audiovisual material (in contrast to the rapidly increasing number of lecture recordings), but Matterhorn should nonetheless offer a uniform solution for all audiovisual materials, that is a "video management system". In this module, the functions are mostly taken from the REPLAY system developed by ETH Zurich[4]. The different recording tracks (audio, content, video) are bundled to a media package, content-indexed (at first through optical character recognition of the slides, later through audio recognition) and if necessary archived in the most native formats. They are encoded according to the specified distribution parameters.

Distribution

The distribution demands of the universities are extremely heterogeneous: they go from simple integration of the videos in local WCMS or blogs, to posting in password-protected LMS, to distribution via iTunes U or YouTube. Here, the distribution module copes not only with the heterogeneous distribution formats (RSS, Atom, Web service interfaces), but also with the recording formats specified at the beginning (cf. “Schedule/Prepare & Capture”) which are transmitted in homogeneous form to external services and platforms. In addition, the distribution channel re-transmits the information necessary for statistical analysis and user data (e.g. most viewed video or annotations). This is where Matterhorn provides more than the classical distribution channels.

Engage

Although Distribution and Engage modules are closely linked together since both must manage presentation and use of the objects, applications in the Engage module make it possible to use comprehensive information (metadata, video and audio analysis, annotations, use analysis) for intelligent user interfaces. Likewise, support of learning management systems (LMS) or virtual learning environments (VLE) is an important issue. To make sure that the produced material will be used, Matterhorn video and audio player components are easily integrated in existing course websites, wikis, and blog systems. Just as in the distribution module, collection of user statistics is supported and the virtPresenter project is leveraged as the baseline for the Engage applications[5]. Social annotations[6], which can be used to improve search or navigation and feedback possibilities, are flown back to the system like the user statistics already mentioned.

4 http://www.replay.ethz.ch
5 McGreal, R. (2004). Learning Objects: A Practical definition. International Journal of Instructional Technology and Distance Learning 1(9)
6 Waitelonis, J., Sack, H. (2008). Zeitbezogene kollaborative Annotation zur Verbesserung der inhaltsbasierten Videosuche. in: Birgit Gaiser and Thorsten Hampel and Stefanie Panke (eds.): Good Tags and Bad Tags - Workshop "Social Tagging in der Wissensorganisation",Waxmann, 2008

In this module, barrier-free accessibility is more than a catch phrase; components are designed to support captions, screen readers and keyboard navigation. Not all channels, external systems and platforms are supported right from the start of the Matterhorn project. But the open architecture makes it possible to create interfaces to existing systems. Overall, the possibility of integrating existing applications in Matterhorn is one of the main properties of the architecture.

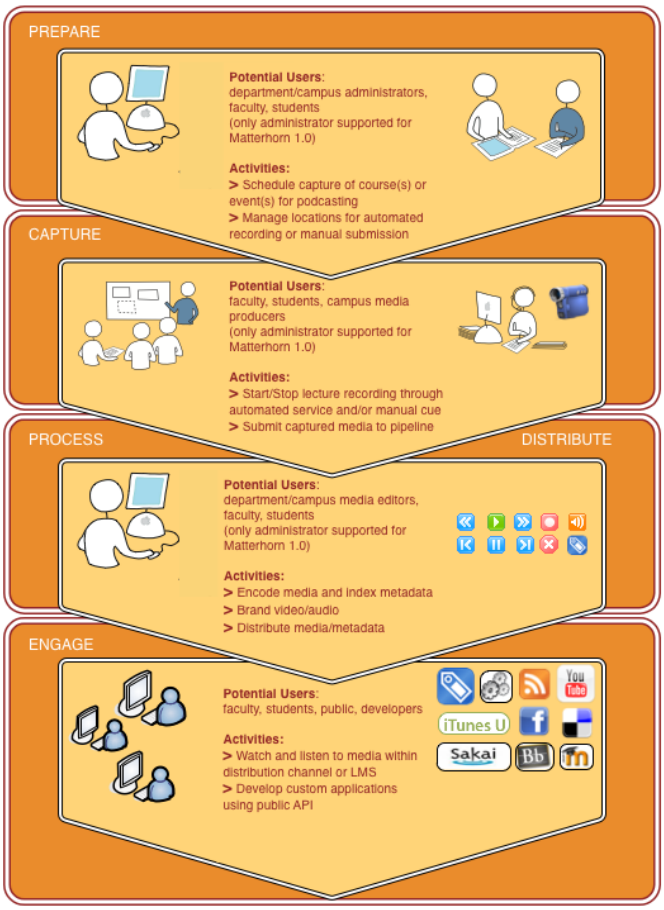


Figure 10: Phases of the Matterhorn Workflow

Media Analysis

After audio/video material has been sent to the inbox (see “Process”), the media is bundled into a media package. A media package is considered the business document within the Matterhorn system. Besides the media objects, it includes further information from media analysis as well as metadata. Every media package therefore consists of a manifest and a list of package elements that are referred to in the manifest. Package elements are media tracks (audiovisual material; movie container); metadata catalogues and further attachments (slides, pdf, text, annotations). Services are planned to modify media packages (update metadata, change attributes). A media package example as well as service description can be found online in the project wiki documentation. Media analysis helps to implement not only the basic navigation features for engage applications (e.g. slide changes, chapters). By indexing slides as well as (English) audio, media analysis also opens the content of the video over time by creating isochronous metadata. Stored in MPEG-7, they are forming the basis for the

searchability of the video and its subsequent accessibility. REPLAY from ETH Zurich as well as different research projects have shown successfully that this technology can be used. For its further development, Opencast Matterhorn is looking forward to benefit from work of the OCRopus[7] group for document analysis and OCR and speech recognition related research.

Metadata

While the indexation of slides and audio provides much of the isochronic metadata to search the video, static metadata is still needed to describe and classify the object – and to facilitate its exchange across institutions. While this domain calls for different areas to be covered (standards like LOM/ IEEE 1484.12.1-2002, protocols like OAI-PMH or technologies like SRU/SRW), the Opencast Community has taken the first step to work on a metadata scheme describing academic video and recorded lectures in particular[8].

License, Miscellaneous

Matterhorn is published under the Educational Community License (ECL) 2.0 developed by UC Berkeley, a license based on Apache 2.0 licensing which takes into account certain particular needs of academic institutions. The software is being developed using Agile software development methodologies to be able to cope with the relatively short duration, the communication between the Opencast Matterhorn consortium and the Opencast Community and a team dispersed over two continents. For the project management the Atlassian products, Confluence (project management) and Jira (issue tracking) are used.

Content distribution and Engage Applications

In order to bring the content to the users, Matterhorn includes web and streaming server solutions for media and content distribution. In addition to the open source streaming server applications Red5 or Mammoth, the corresponding web server applications such as Lighty or Apache with mod_H264 support are also being integrated. Naturally, apart from the SWF-FLV video format, other formats are also supported (e.g. MPEG-4, WMV, podcast variations, HTML5 etc.). In the Distribution and Engage modules, the exchange of information takes places over service interfaces. Data is requested over SOAP or REST and transmitted and processed in form of JSON, XML, ATOM or RSS messages to the relevant components. For the intelligent user interfaces, Flex programming has been used for the most part in conjunction with Ajax technologies. The virtPresenter system from the University of Osnabrück is the main source for the development in this area[9]. Fluid[10], an Open Source community project, assists in the development of new user interfaces, providing guidelines and components to achieve barrier-free accessibility and user friendly interfaces.

Integrating Existing Applications

As mentioned before, the Matterhorn consortium brings together a range of partners with different focuses and strengths in the process of recording and distributing lectures. The SOA

7 http://code.google.com/p/ocropus
8 http://www.opencastproject.org/project/metadata
9 http://www.virtpresenter.org/
10 http://fluidproject.org

concept and the fundamental understanding that a monolithic system cannot satisfy the heterogeneous needs of international universities should also play a key role in attracting other universities to participate in the project, especially those who already have their own system with respective strengths. The objective of the Matterhorn partners is to keep the system, its design and its development as flexible as possible.

5.2 Current state/workflow

Matterhorn workflow on a high level is described in detail in previous chapter (Description). For the transLectures project it is important to know that processing of the videos inputted to the system is almost automatic (i.e. batch processing). At the moment the only possibility to stop the automatic processing of the video is to select the "Review/Trim before encoding" Hold option before inputting the video in the system.

For the transLectures project several new workflows and/or options for the processing of a video would need to be created:

- Workflow that will include transcription
- Workflow that will include transcription and translation
- New Hold option: Review/Edit transcription and/or translation

Matterhorn already has a rudimentary speech recognition API (in development), where an outside Web service is contacted. Service accepts video (or audio only), processes it and returns the result of transcription in service-specific XML format, which include speaker's ID (if there are more speakers), confidence level and alternative words. This format is converted (internally in Matterhorn) to MPEG-7 format and WebVTT and/or TTML/DFXP subtitle format for the use in Engage or other players.

For transLectures this speech recognition API would have to be extended to cover communication with our Web services. Converters between formats (WebVTT <-> MPEG-7 <-> TTML/DFXP) would also have to be written, if they are not already present in Matterhorn.

5.3 Technical requirements

Matterhorn system can run on Linux servers (supported OSs are Red Hat, CentOS and Ubuntu) and on Mac OS X. For processing the input videos it requires quite a lot of CPU power, memory and disk space because:

- It's written in Java
- There's a lot of internal conversions of same input video to different video formats (including H.264)
- Input videos can be quite large (up to several GBs)
- Internal processes include compression and OCR

Because of these requirements it would be unreasonable to run Matterhorn system and transLectures processes on the same machine.

6. Generic system requirements

The transLectures software could be used in the following use cases:

1. Installed at a local server at VideoLectures, Polimedia or any other participant
2. Used in a "Software as a Service" (SaaS) model from a service provider from transLectures

For every case we will define a set of standard procedures that will allow the platform of Content Providers (VideoLectures, Polimedia) to interact with the transLectures server and to get the transcription and translation for any media.

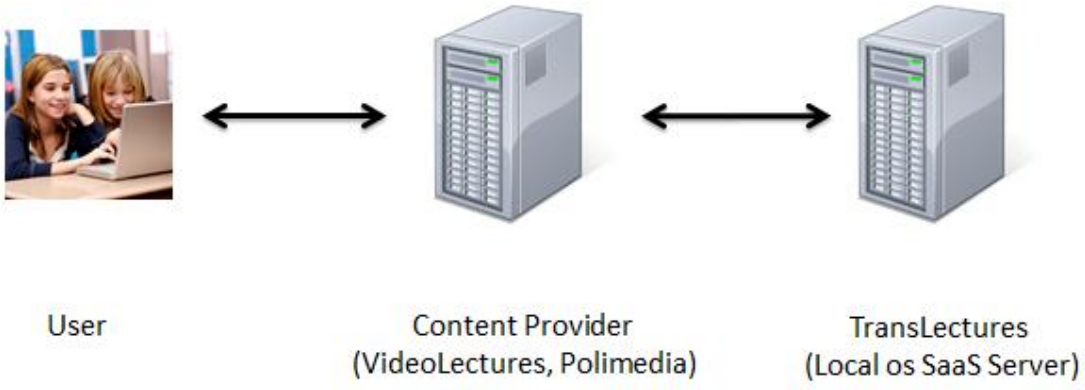


Figure 11: Overview of transLectures system

Most importantly, we will define a standard format for the data exchange between the user, Content Provider and the transLectures server. Also, we will define the endpoints that will enable the communication between the Content Provider and the transLectures server.

6.1 Encoding Formats

An important issue on the design of our solution is to choose the encoding of the data between the users, Content Provider (VideoLectures, Polimedia) and the transLectures server.

We have chosen the standard **MPEG-4 Timed Text** (also known as **TTML** or **DFXP**) as our exchange file format. We are aware of other possible alternatives like MPEG-7 of WebVTT. As we are committed to allow integration with Matterhorn and possible other platforms we may provide interfaces for those standards if needed.

The complete definition of the current TTML/DFXP standard is available at http://www.w3.org/TR/ttaf1-dfxp/.

TTML supports extensions (see Appendix E in the above document), so we will define an extension namespace for the transLectures project and inside that namespace we will define the specific XML tags required for the project needs.

A TTML document instance consists of a `tt` document element that contains a header and a body, where the header specifies document level metadata, styling definitions and layout definitions, and the body specifies text content intermixed with references to style and layout information and inline timing information.

Example of an empty TTML document structure:

```
<tt xml:lang="" xmlns="http://www.w3.org/ns/ttml">
  <head>
    <metadata/>
    <styling/>
    <layout/>
  </head>
  <body/>
</tt>
```

For the body part, a TTML document looks like follows:

```
<body region="subtitleArea">
  <div>
    <p xml:id="subtitle1" begin="0.76s" end="3.45s">
      It seems a paradox, does it not,
    </p>
    <p xml:id="subtitle2" begin="5.0s" end="10.0s">
      that the image formed on<br/>
      the Retina should be inverted?
    </p>
    <p xml:id="subtitle3" begin="10.0s" end="16.0s" style="s2">
      It is puzzling, why is it<br/>
      we do not see things upside-down?
    </p>
    <p xml:id="subtitle4" begin="17.2s" end="23.0s">
      You have never heard the Theory,<br/>
      then, that the Brain also is inverted?
    </p>
  </div>
</body>
```

For the use in transLectures project it will be necessary to extend the TTML file format with some new tags, reflecting the added value of the project. It is expected that if any standard TTML player encounter any of these tags, it will skip them, but they will be recognized by the transLectures player.

We will add the following tags:

- `<#globalconfidence value="XX"/>` tag
  where XX is an integer value between 0 and 100 meaning the confidence index for the text inside of that tag. It can appear only in the `<head>` section.

- `<#conf value="XX"> </#conf>` tag
  We will use this tag to mark a single word or a group of words with a different confidence level than the one of the header.

Example:

```
<p xml:id="subtitle4" begin="17.2s" end="23.0s" >
  You have <#conf value=”12”> never </#conf> heard the
Theory,<br/>
  then, that the Brain also is inverted?
</p>
```

Sets a different confidence level for the word “never” than the specified in the header section.

`<#conf>` tags can be nested, for instance:

```
<p xml:id="subtitle4" begin="17.2s" end="23.0s" >
  <#conf value=”44”>
    You have <#conf value=”12”> never </#conf> heard the
Theory,<br/>
    then, that the Brain also is inverted?
  </#conf>
</p>
```

Sets a confidence level of 44 for the whole sentence except the word never, that has a confidence level of 12.

- `<#alternate> </#alternate>` and
  `<#altoption value=”XX”> </#altoption>` tags

  Example:

  ```
  <#alternate> Most Likely Text
    <#altoption value=”XX”> Less likely text </#altoption>
    <#altoption value=”YY”> Another less likely text </#altoption>
  </#alternate>
  ```

  This is a tag specifying alternatives for a given text. Enclosed will appear the most likely option. The other options will appear enclosed in `<#altoption>` tags with their own confidence level.

- `<#manual author=”Name”> </#manual>` tag
  This is a special tag specifying that author named `“Name”` has made a manual update for a block of text. These tags will enclose only the text that has been manually updated.

6.2 transLectures engine interface/endpoints

To allow the interaction between the Content Provider Server and the transLectures server we need to define a set of standard endpoints that will allow uploading a media file, getting the specific media id’s transcription/translations and finally uploading modifications for a specific media id’s transcription/translations.

The interfaces will be the following:

- `/ingest`
  This will be a POST service in which an authorized user can submit a recording to be transcripted/translated. The upload will be composed of a recording in a suitable video/audio format and a manifest.xml file. The manifest.xml will include information about the recording, the author, the language of the recording, the place of recording, and maybe a proposed id.

  `/ingest` will return a `<translectures_id>` for the media uploaded.

- `/status?id=<translectures_id>`
  This will be a GET service that will be used to see the status of an upload (waiting for processing, processing, processed, transcripted, translated). It will also return the list of translations available for that id.

- `/status?id=<translectures_id>&lang=<lang_id>`
  This will be a GET service that will be used to see if there is a translation for the language `<lang_id>` for a specific id. `<lang_id>` is a standard language IETF tag (RFC 5646).

- `/dfxp?id=<translectures_id>`
  This will be a GET service that will return transcription and all translations of the video in the TTML format.

- `/dfxp?id=<translectures_id>&lang=<lang_id>`
  This will be a GET service that will return a translation of the video to language `<lang_id>` in the TTML format.

- `/mod`?...
  will be a POST service to change part or parts of a transcription/translation, with the following parameters:

  - `id=<translectures_id>` the id of the media to be changed
  - `p_id=<paragraph_id>` the paragraph to be changed (the id of the `<p>` tag)
  - `lang=<lang_id>` the standard language IETF tag (RFC 5646)
  - `translation="text"` the corrected text

6.3 Requirements summary

	VideoLectures	**Polimedia**	**Matterhorn**
transLectures SW (engine)	Running on own server (Linux) or accessible via remote interface	Running on own server (Linux) or accessible via remote interface	Accessible via remote interface
Data exchange format	TTML/DFXP	TTML/DFXP	TTML/DFXP
Data exchange interface	Web service/endpoints	Web service/endpoints	Web service/endpoints

Transcription performance	85%	85%	85%
Translation performance	60-70%	60-70%	60-70%
Scalability	Min. 10 full lecture videos/day	Min. 20 shorter videos/day	(Depends on organization)

If transLectures SW (engine) will be running locally, it should observe the HW limitations as described in VideoLectures/Polimedia Technical requirements' chapters.

7. Conclusion

This document presents the set of requirements for the RTD and integration work packages. They are taking into consideration the two case studies namely VideoLectures and Polimedia as well as Matterhorn open source academic video processing pipeline.

Defined requirements cover the two defined architecture scenarios. It has been decided however that because of multiple developing partners, different licenses and different levels of maturity, the selected architecture will reside on the SaaS model.

Based on this decision, the technical requirements, use and feel requirements and requirements for the accuracy and efficiency of transcription and translation services have been set. It has to be noted however that these requirements have been set for the final case of real-life installation of services in the two case studies and Matterhorn framework.

8. References

1. “Annex I - Description of Work” of Grant agreement for project transLectures, version 2011-07-11
2. transLectures project Web page: http://translectures.eu
3. VideoLectures Web page: http://videolectures.net
4. Internal Polimedia Web page at UPV
5. Matterhorn Web page: http://opencast.org/matterhorn/
6. Latest version of Timed Text Markup Language (TTML/DFXP) standard: http://www.w3.org/TR/ttaf1-dfxp/
7. Latest version of Web Video Text Tracks (WebVTT) standard: http://dev.w3.org/html5/webvtt/
8. Latest version of Multimedia Content Description Interface (MPEG-7) standard: http://mpeg.chiariglione.org/standards/mpeg-7/mpeg-7.htm
9. UPVLC, XEROX, JSI-K4A, RWTH, EML and DDS. Transcription and Translation of Video Lectures. In Proc. of EAMT, 2012.

[All webpages accessed on 31. May 2012]

A) Acronyms

UPVLC	Universitat Politècnica de València
XRCE	XEROX Research Center Europe
JSI	Josef Stefan Institute
K4A	Knowledge for All Foundation
RWTH	RWTH Aachen University
EML	European Media Laboratory GmbH
TTML	Timed Text Markup Language format/standard
DFXP	Distribution Format EXchange Profile
WebVTT	Web Video Text Tracks format/standard
FLV, MP4, WMV	Different video formats
MP3, AAC, WMA	Different audio formats
CBR	Constant BitRate
VBR	Variable BitRate
H.264	MPEG-4 part 10 or AVC (Advanced Video Coding) standard for video compression
CC	Closed Captioning
API	Application Programming Interface
SOA	Service-Oriented Architecture
OSGI	Open Services Gateway Initiative framework for Java
WCMS	Web Content Management System
LMS	Learning Management System
OCR	Optical Character Recognition
SOAP	Simple Object Access Protocol for implementation of Web Services
REST	REpresentational State Transfer, a style of software architecture for distributed system
JSON	JavaScript Object Notation, lightweight text-based open standard
ATOM	XML language used for Web feeds
RSS	RDF Site Summary Web feed format
RTD	Research & Technical Development