Macrothink Institute™

Journal of Educational Issues
ISSN 2377-2263
2015, Vol. 1, No. 2

Automatic System for Producing and Distributing Lecture Recordings and Livestreams Using Opencast Matterhorn

Rafael Jonach

Educational Technology, Graz University of Technology

Münzgrabenstraße 35a, 8010 Graz, Styria, Austria

E-mail: rafael.jonach@student.tugraz.at

Martin Ebner (Corresponding author)

Educational Technology, Graz University of Technology

Münzgrabenstraße 35a, 8010 Graz, Styria, Austria

E-mail: martin.ebner@tugraz.at

Ypatios Grigoriadis

Educational Technology, Graz University of Technology

Münzgrabenstraße 35a, 8010 Graz, Styria, Austria

E-mail: ypatios.grigoriadis@tugraz.at

Received: November 30, 2015 Accepted: December 15, 2015

Published: December 15, 2015

doi:10.5296/jei.v1i2.8653 URL: http://dx.doi.org/10.5296/jei.v1i2.8653

Abstract

Lectures of courses at universities are increasingly being recorded and offered through various distribution channels to support students' learning activities. This research work aims to create an automatic system for producing and distributing high quality lecture recordings. Opencast Matterhorn is an open source platform for automated video capturing and distribution and will be integrated fully in the introduced system. Beside lecture recordings

livestreams of events in lecture halls are also being distributed, so that students have access to lecture content anywhere. In this publication we describe the technical implementation as well as discuss the strength and the weakness of such solution.

Keywords: Lecture, Recording, Livestream, Distribution, Opencast, Matterhorn

1. Introduction to Lecture Recordings

According to literature, the workflow to create a lecture recording is divided into four phases (Huerst, Mueller, & Ottmann, 2006; Mertens, Ketterl, & Vornberger, 2007). The phases are separated chronologically and must be run through step by step. They are called preparation phase, live event & recording phase, post-processing phase and post-usage phase. The first phase, the preparation phase, takes place prior to the recording. The lecturer prepares the material being presented in the lecture and the recording equipment is being set up. Since almost every presentation is based on electronic material, the best way to capture is grabbing the screen of the presentation device such as a video projector. Other presentation techniques such as blackboard, flip chart and overhead projector are recorded by a mostly additional video camera. In the second phase, the live event & recording phase, the lecture takes place and is recorded. Depending on the recording equipment, additional staff such as a cameraman is maybe needed. Ideally neither the lecturer nor the audiences notice the recording (Birnbaum, 2012). Often special recording software is used to produce a high quality raw material for the following phases (Wulff & Fecke, 2012). A high quality lecture recording should have captured every teaching aspect performed by the lecturer so students don't miss important parts of the lecture. In the post-processing phase the raw material is being converted to a unified target format. Furthermore some post production steps like reducing the ambient noise, trimming and cutting the audio and video material or insert branding and credits are made in this phase. The goal is to produce a unified lecture record with a defined structure also including all metadata needed for the next phase. In the final post-usage phase, the produced lecture record is being distributed over various channels, so that students can watch or replay the lecture (Grigoriadis et al., 2013).

These lecture recordings can be used in various scenarios and therefore be useful to reduce the amount lecturers or increase the amount of students attending one course (Mertens, Knaden, Krüger, & Vornberger, 2004) (Nagler et al., 2010). High quality lecture recordings can replace a classical lecture, so students do not need to visit all lectures of one course or one lecture will be streamed to another lecture hall to raise the amount of available seats. Other scenarios extend the classical lecture with additional presentation techniques and interactions (Ketterl, Mertens, & Morisse, 2006; Wachtler & Ebner, 2014). This paper focuses on lecture recordings offered to students, so they can watch or study the lectures after they took place (Nagler et al., 2008). Those high quality lecture recordings must be very intuitive to use and offer good navigation features (Mertens, Brusilovsky, Ishchenko, & Vornberger, 2006; Mertens, Ketterl, & Vornberger, 2007). A big number of lecturers use video projectors to present information from a computer during the lecture (Sewasew, Mengestie, & Abate, 2015). Therefore it is common to record two separate video streams, first the lecturer and second the video signal to the projector.

The goal of this paper is to introduce an automatic system for producing and distributing high quality lecture recordings. Opencast Matterhorn is an open source platform for video capturing and distribution. The introduced system extends Opencast Matterhorn with some functionality to fulfil all requirements. This paper describes all modules and components besides Opencast Matterhorn and how they interact with each other.

2. Research Design

In our research study we are strongly following the approach of information systems prototyping. According to (Alavi, 1984) as well as (Larsen, 1986) prototyping is based on four steps: identifying basic requirements, development of a working prototype, implementation and usage, and revision. Therefore, we identified students' requirements according lecture recording (Nagler et al., 2008) and developed a working system (Nunamaker & Chen, 1990). The following chapters will describe these new components in more detail.

3. Architecture of the Prototype

Opencast Matterhorn can receive new recording material in three different ways: As a scheduled recording from a capture agent, uploading the recording via the administration interface or by calling a REST method (Opencast Matterhorn Documentation, n.d.). Since the goal is to automate the process, the only valid method is calling a REST service to ingest new lecture recordings.

Figure 1 presents the basic architecture around Opencast Matterhorn. The admin user interface (admin UI) and engage user interface (engage UI), as well as the REST service, are provided as part of Opencast Matterhorn (Opencast Matterhorn Documentation, n.d.) but the Matterhorn client API and file system scanner are custom applications which will be described later. Their goal is the fully automatic ingestion of new AV media (audio-visual media) into Opencast Matterhorn. Opencast Matterhorn uses OSGi to implement a modular framework (Opencast Matterhorn Technology Stack, 2013) and therefore can be extended easily by custom modules. OSGi is an open standard for a hardware independent modular system for the Java programming language.

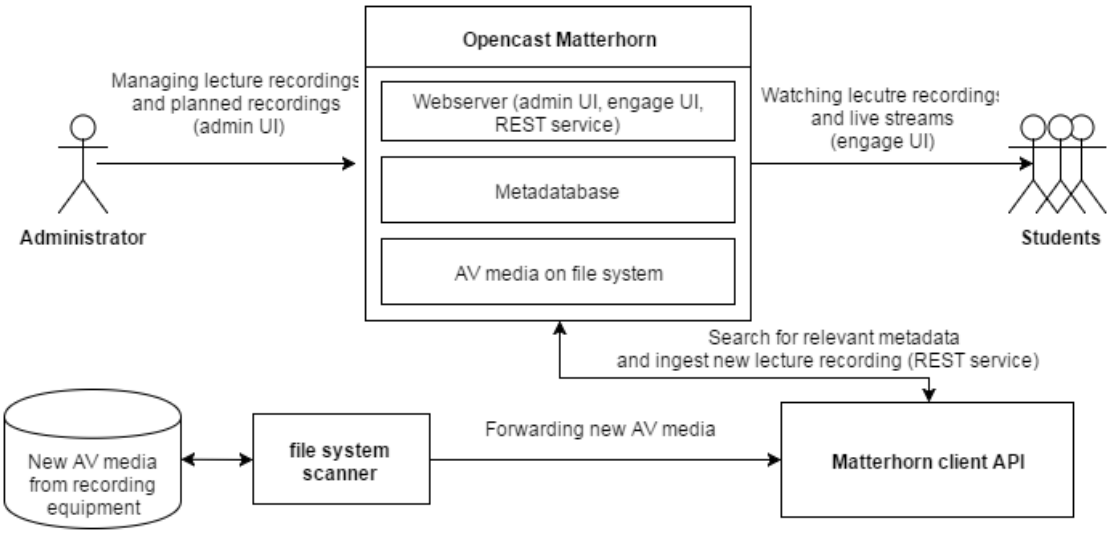

Figure 1. Overview of the workflows for ingesting, managing and watching lecture recordings

As a central component Opencast Matterhorn manages all lecture recordings. It provides a web server which offers essentially three modules: Admin UI for managing lecture recordings, engage UI for watching lecture recordings and REST service for automation of processing all data. The file system scanner and the Matterhorn client API are responsible for receiving and ingesting new AV media into Opencast Matterhorn. The file system scanner receives new raw material from the recording equipment (AV media) and forwards it to the Matterhorn Client API. The Matterhorn client API uses the REST service of Matterhorn to collect the appropriate metadata and ingest a new lecture recording to it. Both components are described in detail later.

A new lecture recording consists of the raw material captured by the recording equipment and all metadata like name of the lecturer, location, date, name of the course and name of the lecture. Both, the AV media and metadata, needs to be passed to Opencast Matterhorn to ingest a new lecture recording. The metadata must be entered before the lecture takes place and is stored in the metadatabase of Opencast Matterhorn by extending it with the s*cheduled recordings and livestream module* (SRL module). This module stores all metadata for scheduled recordings and livestreams. Ideally, the metadata come from an information management system of the university. Due to the fact that scheduled recordings and livestreams have very similar metadata, they are stored in the same module and extend the system with livestream prospects. The SRL module extends the REST service of Opencast Matterhorn for creating, reading, updating and deleting scheduled recordings and livestreams.

4. Implementation

Three components are necessary to implement the fully automatic workflow for ingesting lecture recordings and providing live streams: the *file system scanner*, the *Matterhorn client API* and the *scheduled recordings and livestream module*. Each component is described in one subchapter. The last chapter describes the new engage User Interface (UI) called Tube (http://tube.tugraz.at), which is a redesign of the standard Opencast Matterhorn engage UI

with additional functionality. Figure 2 describes the workflow in an UML activity diagram.

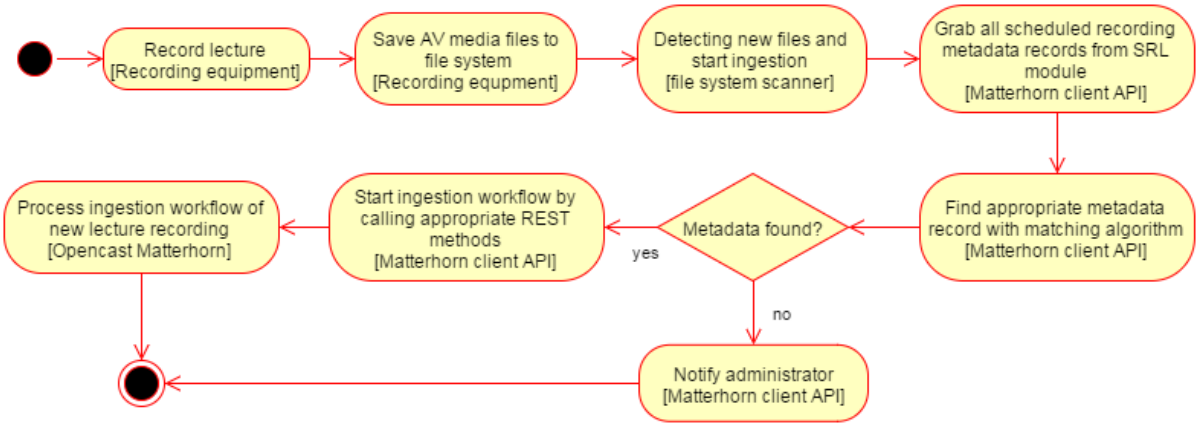

Figure 2. UML activity diagram showing all activities inside the ingestion workflow

Each node states the actor, which executes the described action in the squared brackets. The workflow touches all actors described in this chapter and shows the activities in sequential order. Note that the administrator must enter the necessary metadata for the lecture before the workflow starts (see section 3.3).

4.1 File System Scanner

The recording equipment produces the raw material for the lecture recording and saves it in a defined place on the file system. The file system scanner is responsible for monitoring this place and, in case of new AV media, starting the ingestion workflow by calling the Matterhorn client API. Since the file system is managed by a unix-like operating system, the file system scanner is implemented as a shell script.

4.2 Matterhorn Client API

The Matterhorn client API is a library written in Java, which abstracts all the tasks of an administrator to insert a new lecture recording into Opencast Matterhorn. Thus, the machine-to-machine communication is facilitated and other programs can easily insert new lecture recordings via this API. If the API receives new AV media from the file system scanner it needs to look up the relevant metadata record stored in the SRL module by calling appropriate REST service methods provided by the module.

The Matterhorn client API uses Apache HttpComponents (2015) to communicate with the REST service of Opencast Matterhorn. It takes about nine various calls to the REST service to ingest a single lecture record. The Matterhorn client API simplifies this task to one single call.

There are three ways to use the Matterhorn client API: Including it as an external library, executing the API with specific arguments for a single ingestion or let it start a server socket

and send it a message. The first way is convenient, if the API gets called by another Java application. The other two ways are more independent and are done by executing the Matterhorn client API with specific arguments.

4.3 Scheduled Recordings and Livestream Module

The scheduled recordings and livestream module (SRL module) is an OSGi module and extends Opencast Matterhorn. It adds the functionality to store and manage scheduled lecture recordings and livestreams by extending the database with one table. For each scheduled recording and scheduled livestream a single record is inserted into the table. This record contains all metadata but no AV media of the lecture recording. Since the metadata of a scheduled recording and a scheduled livestream have basically the same structure, they are stored in the same table. Therefore the system can be extended by the functionality of livestreaming without immoderate effort.

The SRL module also extends the REST service of Opencast Matterhorn to enable the communication with other applications and services. It is possible to create, read, update and delete scheduled recording and livestream records. Opencast Matterhorn provides a test area for all its REST service methods. Within this test area an administrator can manage the records by calling the methods to manipulate the database. As described later, the future goal is to implement an additional user interface for managing scheduled recordings and livestreams, but for now the administration within the test area is sufficient.

The Matterhorn client API requires the AV media and the corresponding metadata to start the ingestion workflow. But it only receives the AV media and must look up the metadata in the SRL module. This is done by a matching algorithm: The AV media is named in a specific pattern including the recording start date and the lecture hall. The matching algorithm fetches all scheduled recordings from the SRL module and filters them by the given recording date and lecture hall, to find an appropriate metadata record. Whenever no appropriate metadata record is found, the system automatically alerts the administrator.

4.4 TUbe Portal

The TUbe portal is a redesign of the Opencast Matterhorn engage user interface. It consists of static html sites, which retrieve their data from the REST service of Opencast Matterhorn. Only a few functional features are implemented in the TUbe portal. It uses the Paella Player (2015) instead of the Matterhorn Engage Player for watching the recordings.

Since the automatic ingested lecture recordings have the same internal structure as traditional lecture recordings, the TUbe portal, which serves as an output channel for those recordings, does not need to differentiate between the two types of lecture recordings. The only new feature built into Opencast Matterhorn is the ability to play livestreams back. To fulfil this feature, the TUbe portal contains of an additional site to watch livestreams.

5. Evaluation

The implementation of the system has been described in the previous chapter and was done without significant complications. The TUbe portal is online since 01.10.2014 and replaced

the old commercial system Desire2Learn named Curry. Currently there are 63 series in the system, therefrom 43 represent a complete course and can be visited without attendance. Within these series, there are a total of 588 separate recordings, since each course has about 13 lectures during one semester. The remaining 20 series contain 221 lectures about other education related material or other events at the Graz University of Technology.

During a single semester currently about 10 lectures a week are recorded and distributed as an online lecture to students in TUbe. The ingestion of the new lectures is currently done manually because the automated process presented in this document is just in the testing phase.

The fully automated process for ingesting new web lectures to Opencast Matterhorn was successfully tested on the current test system. A new version of Opencast, Opencast 2.0 (Matterhorn is now Opencast, n.d.), was released during the implementation and the project management team decided to upgrade the software before using the automated process for ingestion. As described in the next chapter, most likely there will be some problems with the upgrade of Opencast since the presented components and modules more or less depending on the implementation and interfaces of Opencast.

The TUbe portal is not streaming the lectures by itself to the client browser, but a streaming engine, Wowza Streaming Engine (2015), takes over this job. Wowza is officially supported by Opencast Matterhorn, so that by following a step-by-step instruction (Installing Wowza as Opencast Streaming Engine, n.d.) the installation of Wowza in Opencast Matterhorn is done within minutes.

Until the end of 2014 it could be observed in the online statistics of Wowza that approximately 30 to 40 streams were played at the same time. Since 2015, a mechanism in Wowza was unlocked, which falsified this statistics. It is assumed that the number of streams has increased slightly. The maximum server load was measured during a livestream with 303 viewers, which proceeded without problems.

6. Discussion

The Matterhorn client API is written in pure Java and very modular. Due to the fact that it is easily extendable, there are much more thinkable use cases. For example, it can be extended by a graphical user interface for administration. Matterhorn Remote Inbox is a similar application developed by the University of Osnabrück (Matterhorn Remote Inbox, n.d.).

Since the Matterhorn client API and the SRL module are heavily depending on the REST interface and OSGi framework of Opencast Matterhorn, the upgrade to new releases of Opencast Matterhorn is likely to produce additional effort to adapt the modules to all changes made in the new release. Matterhorn client API has a code coverage of 100% by all unit tests and therefore all changes made in a new release are found reasonably fast. The SRL module does not depend on any interface rather than on the way the OSGi framework is used and configured in Opencast Matterhorn. So it is sufficient to make a single manual test to check whether the module works correctly or needs some adaptions.

The fully automated workflow for ingesting new lecture recordings starts by scheduling the recordings manually. This can be very time-consuming, since every lecture recording must be entered separately by hand. One future goal of the TUbe portal is to provide a calendar view to the administrator for easily managing and scheduling lecture recordings. It should be possible to import an iCal file generated by the information management system of the university. So the task for the administrator will be simplified and the administrator gains an overview over all scheduled lecture recordings.

The matching algorithm described in the SRL module is very inexact due to the fact that it can match more than one or no metadata record. In both cases, an administrator must manually interfere and choose the appropriate metadata record. This is the case when the records are badly scheduled and more than one or no record is scheduled for the lecture. In the ideal case, the recording equipment tags the AV media with a unique identifier, which is also part of the metadata record and the matching algorithm always finds the correct one.

7. Conclusion

The fully automated workflow for ingesting new lecture recordings has different advantages. The students are more independent because they can watch missed lectures in case of illness or overlapping or even prepare themselves before the exam. Another goal was reducing the administration effort, which will be the case whenever the discussed calendar view in the TUbe portal is implemented. The calendar view will have convenient ways to manage multiple scheduled recordings or import them from the universities information management system.

Opencast is community driven and grows fast. New features and bug fixes are released continuously. Therefore it will be a future task to adopt all changes, which depends in some way on Opencast. It is even possible that Opencast implements some functionality described in this paper, so these became obfuscated and needs to be dropped. Nevertheless, this research study pointed how a automatic recording system can be established in higher education using an open source system.

References

Alavi, M. (1984). An assessment of the prototyping approach to information systems development. *Commun. ACM, 27*(6), 556-563. http://dx.doi.org/10.1145/358080.358095

Apache HttpComponents. (2015). Retrieved November, 28, 2015, from https://hc.apache.org

Birnbaum, N. H. (2012). Distribution von AV-Medien an der Hochschullehre. *Wissenschaftlicher Verlag Berlin*.

Grigoriadis, Y., Stickel, C., Nagler, W., Ebner, M., & Schön, M. (2013). Automated Podcasting System for Universities. *International Journal of Emerging Technologies for Learning, 8*(1), 24-32. http://dx.doi.org/10.1109/ICL.2012.6402060

Huerst, W., Mueller, R., & Ottmann, T. (2006). The AOF Method for Automatic Production of Educational Content and RIA Generation. *International Journal of Continuing*

Engineering Education and Life-Long Learning, 16(3/4), 215-237. http://dx.doi.org/10.1504/IJCEELL.2006.009200

Installing Wowza as Opencast Streaming Engine. (n.d.). Retrieved November, 28, 2015, from https://opencast.jira.com/wiki/display/MHDOC/Wowza+Media+Server

Ketterl, M., Mertens, R., & Morisse, K. (2006). Alternative content distribution channels for mobile devices. *Mictrolearning 2006* (pp. 119-130). Innsbruck, Austria.

Larson, O. (1986). Information Systems prototyping. *Proceedings Interez HP 3000 Conference* (pp. 351-364), Madrid. Retrieved October, 2015, from http://www.openmpe.com/cslproceed/HPIX86/P351.pdf

Matterhorn Is Now Opencast. (n.d.). Retrieved November, 28, 2015, from http://www.opencast.org/matterhorn

Matterhorn Remote Inbox. (n.d.). Retrieved November, 28, 2015, from http://zentrum.virtuos.uos.de/mhri

Mertens, R., Brusilovsky, P., Ishchenko, S., & Vornberger, O. (2006). Time and Structure Based Navigation in Web Lectures: Bridging a Dual Media Gap. *World Conference on ELearning, in Corporate, Government, Healthcare & Higher Education* (pp. 2929-2936), E-Learn 2006, Honolulu, HI, USA, October 13-17, 2006.

Mertens, R., Ketterl, M., & Vornberger, O. (2007). The virtPresenter lecture recording system: Automated production of web lectures with interactive content overviews. *International Journal of Interactive Technology and Smart Education, 4*(1), 55-66. http://dx.doi.org/10.1108/17415650780000076

Mertens, R., Knaden, A., Krüger, A., & Vornberger, O. (2004). Einsatz von Vorlesungsaufzeichnungen im regulären Universitätsbetrieb. *GI Jahrestagung, 1*, 429-433.

Nagler, W., Saranti, A., & Ebner, M. (2008). Podcasting at TU Graz: How to Implement Podcasting as a Didactical Method for Teaching and Learning Purposes at a University of Technology. *Proceeding of the 20th World Conference on Educational Multimedia, Hypermedia and Telecommunications (ED-Media)* (pp. 3858-3863).

Nagler, W., Grigoriadis, Y., Stickel, C., & Ebner, M. (2010). Capture Your University. *IADIS International Conference e-Learning 2010* (pp. 139-144).

Nunamaker, J. F. Jr., & Chen, M. (1990). Systems development in information systems research, System Sciences. *Proceedings of the Twenty-Third Annual Hawaii International Conference* (Vol. 3, pp. 631-640). January 2-5, 1990. http://dx.doi.org/10.1109/HICSS.1990.205401

Opencast Matterhorn Documentation. (n.d.). Retrieved November, 28, 2015, from https://opencast.jira.com/wiki/display/mh16/Matterhorn+Release+Docs+for+1.6

Opencast Matterhorn Technology Stack. (2013). Retrieved November, 28, 2015, from https://opencast.jira.com/wiki/display/MHDOC/Technology+Stack

Paella Player. (n.d.). Retrieved November, 28, 2015, from http://paellaplayer.upv.es

Sewasew, D., Mengestie, M., & Abate, G. (2015). A comparative study on power point presentation and traditional lecture method in material understandability, effectiveness and attitude. *Educational Research and Reviews, 10*(2), 234-243. http://dx.doi.org/10.5897/ERR2014.2027

Wowza Streaming Engine. (n.d.). Retrieved November, 28, 2015, from https://www.wowza.com/products/streaming-engine

Wachtler, J., & Ebner, M. (2014). Unterstützung von videobasiertem Unterricht durch Interaktionen–Implementierung eines ersten Prototyps. *Zeitschrift für Hochschulentwicklung, 9*(3), 13-22.

Wulff, B., & Fecke, A. (2012). LectureSight–An Open Source System for Automatic Camera Control in Lecture Recordings. *Multimedia (ISM), 2012 IEEE International Symposium* (pp. 461-466). http://dx.doi.org/10.1108/ITSE-07-2014-0019

Glossary

AV media: Audio-visual media;

OSGi: Specification describing a modular system for the Java programming language;

REST: Representational State Transfer, a paradigm for web services;

SRL module: Scheduled recordings and livestream module;

TUbe: Video portal of Graz University of Technology;

UI: User interface;

UML: Unified modeling language.

Copyright Disclaimer