

# Data driven Ximpel interactive video

---

J.Heymans

1275089

[june - july 2009]

Interactive video can provide viewers with more options than the usual play, stop, etc. Ximpel is one such interactive video platform which can provide users with questions and choices about which fragments within a video they would like to watch. In this thesis I will describe how adding database functionality to the Ximpel platform will make it more user-friendly and how it can provide possibilities to make the interactive videos created using Ximpel, more dynamic

## Table of Contents

Introduction .....	3
1. Overview of Ximpel interactive video platform .....	5
1.1. What is Ximpel. ....	5
1.2. Features & functions of Ximpel .....	5
1.3. The playlist.xml .....	6
1.4. The ximpelConfig.xml .....	8
1.5. Benefits of a data-driven Ximpel application .....	9
2. Requirements & goals of data-driven Ximpel application .....	10
2.1. Success scenario .....	10
2.2. Functional requirements .....	10
2.3. Non-functional requirements .....	12
2.4. Goal of the application .....	12
3. Design & implementation .....	13
3.1. Design decisions .....	13
3.2. Database structure .....	16
3.3. The code .....	19
3.4. External scripts used .....	21
4. Usage of data-driven Ximpel playlist builder .....	22
4.1. Tutorial .....	22
Conclusion .....	26
References .....	27
Appendix 1: example of a ximpel playlist.xml .....	28
Appendix 2: example of ximpelConfig.xml .....	29

# Introduction

Digital video has become a widely used media format in recent years. Partly because it can better convey the goals of a provider in means of image, motion, sound and text as opposed to static data types like plain text and graphics. Besides this, it's also thanks to content providers such as YouTube who have made it increasingly easy to share and distribute your own personal digital video content on the web.

Traditionally video data is provided in predefined linear format, providing the user the default control options as: play, stop, pause, forward and rewind.

The video is then watched in a mostly passive way from beginning to end during which the user's interaction is very limited. Interactive video could improve the user's ease and efficiency for instance by:

- providing a representative visual summary of the video prior to watching it
- presenting a list of visual entries that serve as access point to desired video content
- showing a list of navigational options that allow users to follow internal and external links between related items in the same video or perhaps other media documents.

For my bachelor project I have chosen to research and develop a tool that will add data-driven functionality to the Ximpel interactive video platform.

XIMPEL was developed for the Clima Futura climate game, in which interactive video is used as a basis for scenario-driven games, with additional mini-games for elaborating on specific topics or tasks that arise during game play. XIMPEL has also been deployed in education, for creating short viral videos which allow for further interactive explorations.

Ximpel itself currently uses manually created playlists in .xml format. The objective of my bachelor project is to develop an application that can automatically generate this xml playlist from data previously inserted into a database. I will develop this application using php & mysql, I have chosen for this web based environment because Ximpel itself is a flash application which can be embedded in websites. I will try to keep the code in php as simple as possible to make it easy to modify and understand the code even when I am no longer available to provide assistance. Because of time constraints I will use mysql as a database during the project, but afterwards I will port the database to sqlite to decrease the need for administrative permissions when using my application.

The goal of my application is to make it easier to create and edit a playlist for Ximpel interactive video applications. To determine if the tool actually makes Ximpel easier to use I will have the application tested by both people who are familiar with Ximpel and those that have never used it before. During the development of this application I will document the steps taken and summarize some of the added benefits of adding database functionality to Ximpel.

A brief explanation of the chapters in this document:

**Chapter 1: Overview of Ximpel interactive video platform**

In this chapter I provide a short explanation of what Ximpel is and I analyze the different features that Ximpel offers

**Chapter 2: Requirements & goals of data-driven Ximpel application**

In this chapter I describe the requirements that I think the application should comply to and I state the goal of the application.

**Chapter 3: Design & implementation**

In this chapter I document the various decisions taken during design and why they were taken.

**Chapter 4: Usage of data-driven Ximpel application**

This chapter is more of a tutorial describing how to use the application I have created.

# 1. Overview of Ximpel interactive video platform

In this chapter I will take a closer look at Ximpel and the possibilities it offers. This analysis is necessary for the development of my application. Because it should first and foremost give the user the means to create an interactive video with the default possibilities that Ximpel offers.

## 1.1. What is Ximpel.

Ximpel stands for the eXtensible Interactive Media Player for Entertainment and Learning. It is an interactive video platform which is being developed at the Multimedia Group within the Faculty of Sciences at the Vrije Universiteit. It was designed and developed by:

- Winoe Bhikharie, MSc (developer)
- Hugo Huurdeman, MSc (developer/designer)
- Marek van de Watering, MSc (designer)
- Anton Eliëns, prof.dr. (supervisor)

Ximpel was originally developed for the Clima Futura climate game but has also been used in education for creating short viral videos which allow for further interactive explorations. When watching a Ximpel interactive video the viewer can be presented with quiz question which have a true/false answer and branch questions. Branch questions require the user to make a choice by clicking on one of the overlays currently displayed on the screen. These overlays can be seen as possible answers to the branch questions. Clicking on one of these overlays leads to the playback of videos that are relevant for the chosen answer.

## 1.2. Features & functions of Ximpel

Ximpel's goal is to provide an open multimedia platform which can be used for both entertainment and education. The features that Ximpel offers are:

- Customizable, clickable overlays and visuals.  
These can be used to access different branches of the storyline and can link to both external and internal information sources.
- Customizable questions.
- A scoring mechanism to give a certain weight to the answers given.

The variables, such as the clips to be shown, the branches, overlays, questions and score points are all modifiable through a collection of 2 XML configuration files. The Ximpel player then reads the information stored in these files.

The 2 XML configuration files are:

- a playlist.xml
- ximpelConfig.xml

### 1.3. The playlist.xml

The playlist contains all the information about subjects, the video files, questions, overlays, scores and branches. Each of these items within the video.xml is in typical xml tag format and each has their own configurable attributes as listed below. An example of a playlist is also included in Appendix 1.

- **subject tags**

Required attributes: id

Optional attributes: leadsto

The main items in the video.xml are subjects, each identified by a unique id attribute. Branching occurs from one subject to two or more and is based on each subject's id. The leadsto attribute of a subject can be used to jump to the specified subject once all media clips of the current subject have been played. A subject also has a <description> tag which can hold text (e.g. full name of subject, descriptive text, etc) that is displayed in the Ximpel media player during playback of all videos belonging to that subject.

All questions, overlays, media, scores and branches are contained within the subject's tags.

- **score tags**

Required attributes: value

Optional attributes: name

The score tag allows the user to add a score value to subjects. The required value attribute contains the score and the optional name attribute can be used to keep track of multiple score parameters.

- **media tags**

Required attributes: none

Optional attributes: order

All media items are placed within media tags. Ximpel offers the possibility for users to define their own media types. Currently the only built-in media item type is video and for this thesis I will only take this type into account. Videos placed between media tags are normally played in the order in which they are defined. It is possible to randomize this order by defining the "order" attribute for the media tag. This attribute can have the following values: *default* (plays items in defined order), *random* (plays items in random order), *randomN* (picks N items and plays them in a random order).

- **video tags**

Required attributes: file

Optional attributes: leadsto, repeat

All information relating to videos is wrapped in <video> tags, this includes individual video files, questions, branches and overlays. Individual video files are listed using the <video> tag, within this tag the file attribute contains the name of the video file. For default flash video files the .flv can be omitted but for MP4 files the extension (.mp4) must be included within the file name. The leadsto attribute contains the id of a subject to which the playback will jump when the current video has ended. The repeat attribute contains a Boolean value (true/false) and can be used to force the user to make a choice by clicking on one of the provided overlays.

- **quiz question tags**

Required attributes: none

Optional attributes: starttime, duration

Quiz questions are shown at the bottom right corner of the video and are either true or false. Quiz questions are defined in pairs of <question> and <rightanswer> tags. Between the question tags you specify the question and between the rightanswer tags you provide a boolean value (true or false) representing the correct answer. The starttime attribute can be used to specify after how many seconds the question should appear and the duration attribute can be used to specify how many seconds the question should appear on screen.

- **branch question tags**

Required attributes: none

Optional attributes: none

Branch questions provide the user with a choice that must be made at a specific moment. Between the branch question tags you place a question that should be displayed. The overlays then represent the possible answers to this question and link to the relevant subject.

- **overlay tags**

Required attributes: none

Optional attributes: starttime, duration

Overlay tags are placed in between <overlays> tags and contain one or more overlaycel tags. The starttime attribute specifies the start time in seconds when the overlaycels should appear on screen and the duration attribute specifies how many seconds the overlaycels should be shown.

- **overlaycel tags**

Required attributes: x, y, width, height, leadsto

Optional attributes: description, color, hover\_color, alpha, hover\_alpha, image, hover\_image, text, textsize, textfont, textcolor, hover\_text, hover\_textsize, hover\_textfont, hover\_textcolor

The x and y attributes specify where the overlay should be placed on the video screen. The width and height attributes specify the dimensions of the overlaycel, and the leadsto contains the id of the subject to which the playback will jump if the overlay is clicked. Because the optional attributes are numerous I will list and summarize them.

- description: the text shown as answer to the branch question
- color & hover\_color: a rgb (format: 0xrrggbb) value representing the color of the overlaycel during normal state and mouseover.
- alpha & hover\_alpha: the transparency value (between 0 and 1) of the overlaycel during normal state and mouseover.
- image & hover\_image: contains a path to an image (images/test.jpg) which is placed in the overlaycel which changes to the image defined in hover\_image on mouseover. Allowed image types are jpeg, png and gif. By only specifying a hover\_image attribute an effect is created that only show an image on mouseover.
- text attributes: The text attributes are self explanatory and control the way text can be display in an overlaycel during normal state and mouseover.

#### **1.4. The ximpelConfig.xml**

The ximpelConfig contains standard configurable options such as text of labels, which file should be used as play list, etc. An example of a ximpelConfig.xml is included in Appendix 2. The options in de ximpelConfig.xml will not be editable in the playlist builder application. Because they are pretty straightforward and all are optional I will list them but will not go into in-depth explanations.



## **1.5. Benefits of a data-driven Ximpel application**

The xml configuration files described in the previous paragraph provide the Ximpel media player with the data it needs to display the interactive video. These xml files currently have to be manually constructed or edited. Although this not that complicated for people that have reasonable knowledge of xml it can be an obstacle for “normal” people. These people might be interested in creating interactive videos with Ximpel but lack the knowledge to create the required playlist xml.

### **Easy to update and more user friendly**

By adding a database to store the variables needed for the playlist xml file the process does not immediately become simpler. By providing users with a backend where they can enter these variables into the database does make the entire process simpler to understand because the actual editing of the xml file is not needed anymore. When all data has been inserted into the database the required xml can be generated by a php script and offered to the user for download. If Ximpel is being used on a website it is also possible to use the database application as a backend. The administrator of the website can change data (such as questions) and in this way provide visitors of the website with a different interactive video every day/week/month or different questions relating to videos.

### **Dynamic content**

Because all data is now contained inside a database it also becomes possible to provide the Ximpel media player with (semi) dynamic content. Having certain data changed in the database at regular intervals can lead to perhaps having different questions displayed on a certain video each time it's watched. For instance, by adding multiple questions with the same starting time to 1 video file in the database, it is possible to have the php script that generates the xml file select a random question each time the xml is generated. In this way a different question is shown during the video, all that would needed to be done is to generate the xml each time the video is watched. This dynamic usage could be applied to many parts of the Ximpel playlist, e.g. videos, overlays, overlaycells, etc.

## 2. Requirements & goals of data-driven Ximpel application

In this chapter I will determine the requirements which my application must fulfill. I will make a distinction between the functional and non-functional requirements and will indicate the importance of each.

To determine which requirements are necessary for the application, I will first describe a typical success scenario. In this scenario a user goes through the different steps in creating a simple interactive video using Ximpel and encounters no missteps.

### 2.1. Success scenario

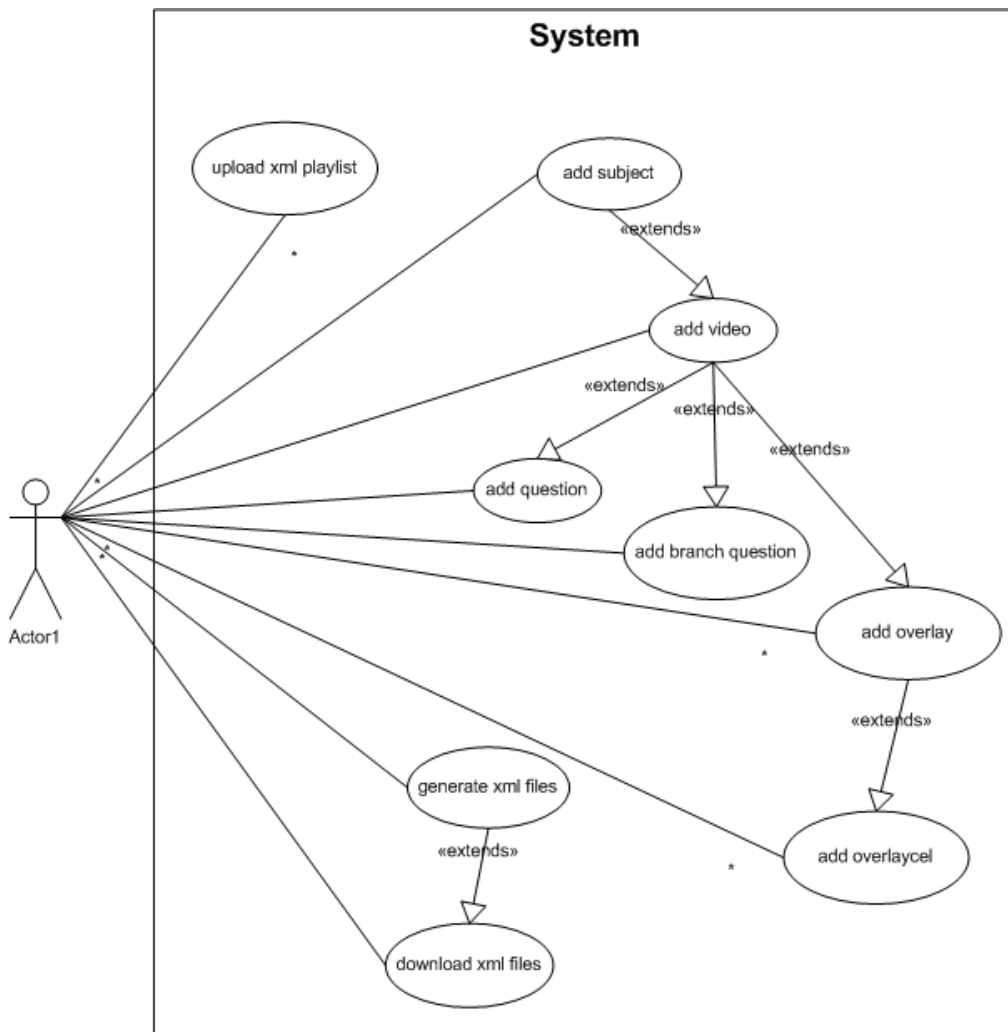
- User adds a subject to the project
- User adds an id, description & score to the subject.
- User adds video files to the subject
- User adds a question to one of the video files
- User adds a branch question to one of the other video files
- User adds 2 overlays (overlay 1 & overlay2) and some of their attributes to the video file containing the branch question
- User adds a second subject to which overlay1 points
- User adds a id, description & score to this subject
- User adds video files to this subject
- User adds a third subject to which the overlay2 points
- User adds a id, description & score to this subject
- User adds video files to this subject
- User generates the required xml files
- User can choose to download the xml file.

### 2.2. Functional requirements

Requirement	Description	Importance
add_subject	User can add a subject to the project and the relevant attributes and option of a subject	Must have
add_videos	User can add videos to a subject with their attributes	Must have
add_question	User can add questions to corresponding to video files	Must have
add_branch_question	User can add branch questions to video files, which provide the viewer with a choice of how to continue.	Must have
add_overlays	User can add overlays to the video file containing the branch question providing the user with	Must have

	the graphical indication of where to click to answer the branch question.	
add_scores	User can add a score to the subjects	Must have
configure_global_options	User can modify the global options of the interactive video	Won't have
generate_xml_files	User can generate and the xml files	Must have
download_xml_file	User can download the generated xml files if they will not be used on the current webserver/location.	Must have
create_mutiple_projects	User can have multiple ximpel projects	Could have

The requirements mentioned above represent the most important requirements that the application will need to meet. To visualize these requirements I also present them in the following use case diagram.



### **2.3. Non-functional requirements**

In this section I will describe the two non-functional requirements which I think are important and I would like my application to comply to. Non-functional requirements are requirements that specify criteria that can be used to judge the operation of a system, rather than specific behaviors.

#### **Usability**

Usability is a term that denotes the ease with which people can use a particular tool in order to achieve the intended goal. Within my application I think usability is the most important non-functional attribute, because my application should make the process of using Ximpel easier. It will try to do this by making the creation of the playlist xml automated and providing the user with a gui of sorts to insert all the data.

#### **Maintainability**

The second most important non-functional requirement of my application is maintainability. Maintainability denotes how easy it is to modify a certain software product. This is important because I will not always be around to maintain the program or explain it each time a modification must be made.

### **2.4. Goal of the application**

Beside the possibilities and features mentioned in chapter 2, Ximpel also offers some advanced possibilities. Some of these are the programming of custom media types and the use of callback events. These advanced features that Ximpel offers will not be included in this document and will also not be implemented in the application at this time. The goal of the application is to provide the means to create “standard” interactive Ximpel videos and through this document offer ideas of how database functionality can expand the possibilities of Ximpel. The main functionality of the application at this time is to create the xml file that Ximpel uses as a playlist.

### 3. Design & implementation

In this chapter I will discuss some of the decisions taken during the design and implementation of the application. For each decision taken I provide the rationale why a certain option was chosen and another one was not.

#### 3.1. Design decisions

Decision #1		<i>Which scripting/programming language should be used?</i>
Options	<i>Option# 1</i> ✓	<b>PHP</b>
	<b>Description</b>	<i>PHP is a server side embedded programming language which is widely used and supported. It's free to use and most webhosting providers provide support for it.</i>
	<b>Status</b>	<i>Accepted.</i>
	<b>Evaluation</b>	<i><b>Advantages:</b> The PHP scripting language is free, it's easier to learn than ASP or JSP and it's widely supported. <b>Disadvantages:</b> Considered not as powerful as JSP.</i>
	<b>Rationale of decision</b>	<i>This option is accepted because it will make the application easier to edit by people other than the developer. Also because it is widely supported it will be very easy to find hosting for it.</i>
	<i>Option# 2</i>	<b>ASP/JSP</b>
	<b>Description</b>	<i>ASP is a programming language created by Microsoft for creating dynamic websites similar to those created with PHP. JSP is a Java technology that allows software developers to create dynamically generated web pages, with HTML, XML, or other document types, in response to a Web client request.</i>
	<b>Status</b>	<i>Rejected.</i>
	<b>Evaluation</b>	<i><b>Advantages:</b> ASP &amp; JSP might be considered more powerful languages than PHP. <b>Disadvantages:</b> ASP is not free, it requires a Microsoft server to run, and can only connect to a Microsoft database. JSP is harder to learn and not as widely supported as PHP.</i>
	<b>Rationale of decision</b>	<i>This option is rejected because the learning curve &amp; cost of using either ASP or JSP would be considerable higher than when using PHP.</i>

Decision #2		Which database system should be used?
Options	<b>Option# 1</b> ✓	<b>MySQL</b>
	<b>Description</b>	<i>MySQL is a relational database management system. MySQL stands for "My Structured Query Language". The program runs as a server providing multi-user access to a number of databases..</i>
	<b>Status</b>	<i>Accepted.</i>
	<b>Evaluation</b>	<i><b>Advantages:</b> Very widely used and supported. <b>Disadvantages:</b> Requires admin access to the database server to setup required tables.</i>
	<b>Rationale of decision</b>	<i>This option is accepted because it more popular than Sqlite, although it does require admin access to the database to be used.</i>
	<b>Option# 2</b>	<b>SQLite</b>
	<b>Description</b>	<i>SQLite is an embedded relational database management system contained in a relatively small C programming library.</i>
	<b>Status</b>	<i>Rejected.</i>
	<b>Evaluation</b>	<i><b>Advantages:</b> No admin access required to setup the database tables. <b>Disadvantages:</b> Not as popular as MySQL (yet) and therefore not as easy to modify.</i>
	<b>Rationale of decision</b>	<i>This option is rejected because although MySQL requires admin access it will be easier to find information on how to modify the code then if SQLite was used.</i>

Decision #3		<i>How should application be implemented?</i>
Options	<b>Option# 1</b> ✓	<b>Web based (the application is displayed and used within a web browser)</b>
	<b>Description</b>	<i>The application is displayed as a website to the user and can be used in most modern web browsers.</i>
	<b>Status</b>	<i>Accepted.</i>
	<b>Evaluation</b>	<i><b>Advantages:</b> No need to install any plugins or software to be able to use the application. <b>Disadvantages:</b> An internet connection can be necessary (unless the server is run locally).</i>
	<b>Rationale of decision</b>	<i>This option is accepted because it provides the easiest access to the application and because it is accessed by using a web browser. It is not necessary to take the different operating systems of users into account. Also Ximpel is effectively a web based video application designed to be used on the internet.</i>
	<b>Option# 2</b>	<b>Stand alone application</b>
	<b>Description</b>	<i>Create a custom application which the user can run/install on his computer.</i>
	<b>Status</b>	<i>Rejected.</i>
	<b>Evaluation</b>	<i><b>Advantages:</b> Can be used without a internet connection. <b>Disadvantages:</b> Defeats the purpose of the making the database driven design available to Ximpel projects which will be most likely distributed on the internet. Providing support for different operating systems will be more difficult</i>
	<b>Rationale of decision</b>	<i>This option is rejected because as Ximpel is web based itself; it would be the most logical choice to create a web based application as well for creating the playlists.</i>

### 3.2. Database structure

After analyzing the structure of the Ximpel playlist xml file (appendix 1) I made a distinction between five main components. These are:

- Subjects
- Videos
- Questions
- Overlays
- Overlaycells

These five components represent the main building blocks of the playlist xml file and I decided to model the database according to them. The database contains five tables each named after one of the components mentioned above. The tables are structured as follows:

Table name: subjects	
Column name	Data type
subjectTable_id	int
subject_id	text
leadsto	text
score	int
description	text
playing_order	text
n	Int
ordering	int

The subjects table is the “main” table in the sense that all the other tables are connected to it in some way. This reflects the structure of the xml playlist in which all other tags are contained within the subject tags. It has a field named “ordering” which is used to determine the order of the subjects, this can be changed in the application to make it easier to switch the playing order of subjects.

Table name: videos	
Column name	Data type
file	int
leadsto	text
repeat	text
subjectTable_id	int
ordering	int

The videos table contains information on the videos that are used, it is connected to a single subject identified by the field named “subjectTable\_id”.



Table name: questions	
Column name	Data type
questionTable_id	int
type	text
question_text	text
starttime	int
duration	int
rightanswer	varchar
videoTable_id	int

Table name: overlays	
Column name	Data type
overlayTable_id	int
overlay_name	text
starttime	int
duration	int
videoTable_id	int

The questions & overlays table contain all information on the questions and overlays respectively. They are both connected to a single video file by the field named "videoTable\_id".

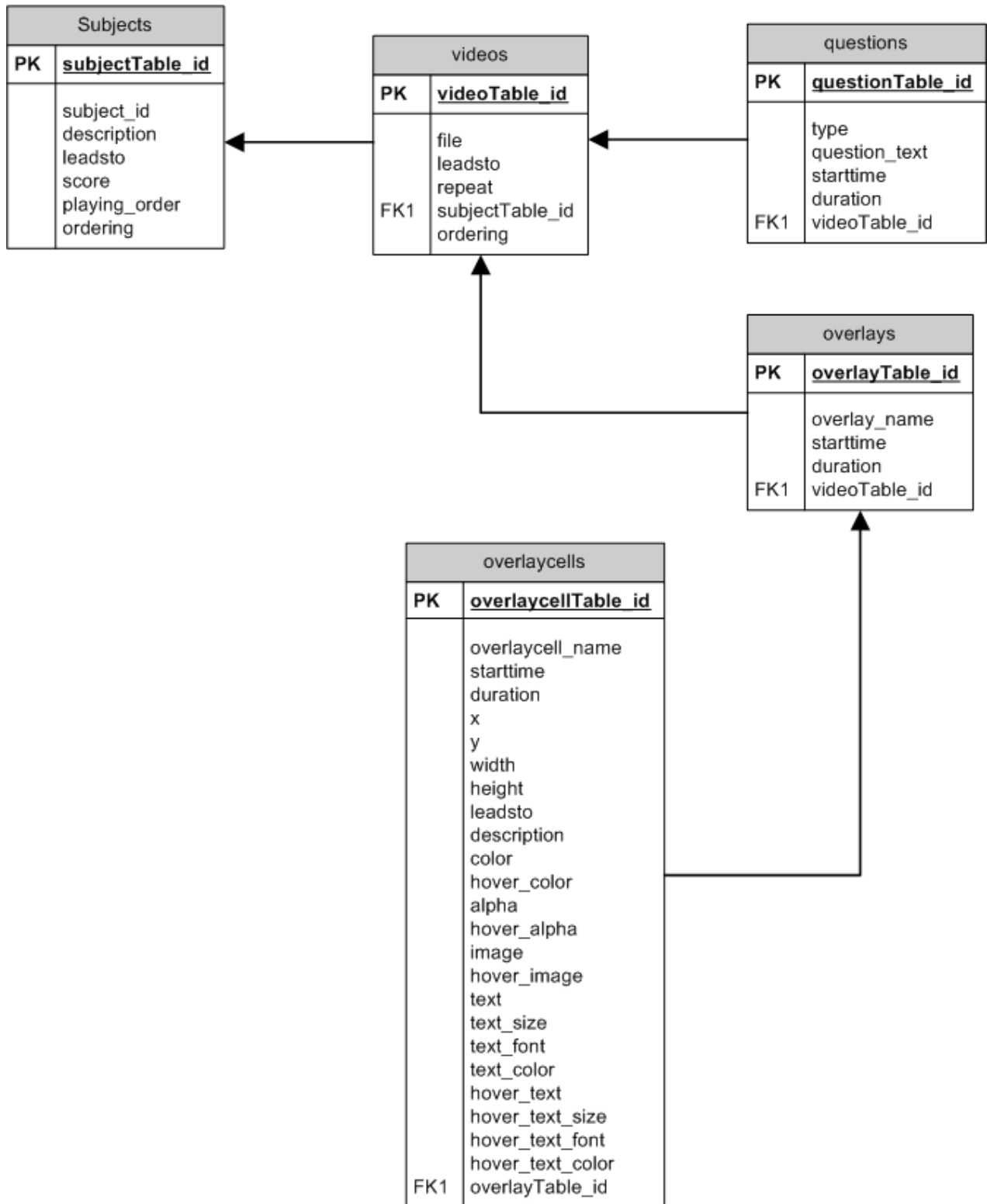
Table name: overlaycells	
Column name	Data type
overlaycellTable_id	int
overlaycell_name	text
x	int
y	int
width	int
height	int
leadsto	text
description	text
color	text
hover_color	text
alpha	double
hover_alpha	double
image	text
hover_image	text
text	text
text_size	int
text_font	text
text_color	text
hover_text	text
hover_text_size	int
hover_text_font	text
hover_text_color	text
overlayTable_id	int

The overlaycells table contains all information on the overlaycells. Each overlaycell is connected to a single overlay by the field called "overlayTable\_id".

The field's named subjectTable\_id, videoTable\_id, questionTable\_id, overlayTable\_id and overlaycellTable\_id are set to auto increment in their respective tables (e.g. subjectTable\_id in subjects, videoTable\_id in videos, etc) the database because this ensures that their value

is always unique. Because it is unique it can be used to link components together and to identify single table entries.

The diagram below illustrates how the different tables are “linked” together.



### 3.3. The code

The functions used in the application are bundled in the following files:

- functions.php
- remover.php
- updateDB.php
- validator.php
- generator.php
- output.php
- upload.php

#### ***functions.php***

The functions in this file are for adding, editing and listing the different components. The application shows lists of subjects, videos, questions, overlays and overlaycells. Each of the five main components have their own set of functions for adding, editing and listing. The functions are as follows (substitute **xx** by the name of one of the five components):

*xxList()*

This function retrieves all entries of a certain type (subject, video, etc) and present them in list format to the user.

*addxx(\$edit)*

This function is used to add an entry (subject, video, etc) to the relevant table. The variable \$edit is used to determine whether it is a new entry or one that has been edited. \$edit is the id of the entry that has to be modified.

*get\_xx(\$xxTable\_id)*

This function gets a single entry from the relevant table so that it can be edited by the user. The variable \$xxtable\_id represents the unique id of the entry that the user wants to edit.

#### ***remover.php***

The functions in this file are used to remove entries from the different tables. By deleting a certain component entry, all entries related to it will also be removed. This happens in the following way:

**Deleting a Subject** will delete all video, question, overlays & overlaycells belonging to it.

**Deleting a video** will delete all questions, overlay & overlaycells belonging to it.

**Deleting a question** will only delete that question

**Deleting a overlay** will delete all overlaycells belonging to it.

**Deleting a overlaycell** will only delete that overlaycell.

The functions are named: *delete\_xx(\$xxTable\_id)*

### ***updateDB.php***

This file is used to update the ordering of the subjects & videos. These are currently the only components which can have their order changed by the user dragging and dropping the different rows into the desired order. The file is called by Ajax to update the ordering in the relevant table without the user having to refresh or navigate away from the current page.

### ***validator.php***

The file is currently only used to validate the subject\_id before it is entered into the database. It is called by Ajax while the user is typing in the subject id, and gives real time feedback if the chosen subject id is unique. This is so far the only value that has to be absolutely unique. The file could be expanded to validate other values.

### ***generator.php***

This file creates the actual xml playlist file. The code in this file queries every table in the database and write the contents to an xml file in the correct structure that the Ximpel application can read.

### ***output.php***

This file gives some “debug” information when the xml file is created. It currently checks 3 conditions:

- That every subject contains videos
- That every video with a branch question also contains at least 1 overlay
- That every overlay contains at least 1 overlaycell

These messages are only warnings or notices and will not stop the creation of the xml file. This because the conditions do not have to be met for the user to create an xml file, it is just to give notice to the user in case he might have missed something by mistake.

### ***upload.php***

This file parses the existing xml which the user uploads, it checks for all tags and their attributes and inserts them into the database. It is only compatible with the xml file for or after the June 2009 version of Ximpel. The file is not saved on the server, so there is no issue with permissions.

### **3.4. External scripts used**

To make the playlist builder more user-friendly I implemented some java scripts I found on the internet. I will list them with a short explanation, the links to where I found them and more information about the individual scripts are also in my references.

#### ***jquery* [4][5]**

This collection of libraries is used in the index.php & new\_subject.php. In the index.php it provides the Ajax functionality to drag & drop the row into a different order. Behind the scenes the script then does a call to the updateDB.php which updates the database with the new order of the fields. In the new\_subject.php the script provides the Ajax functionality which checks if the subject id is unique. It does a call to validator.php on every “key up” and tells the user if the subject is still unique.

#### ***wz\_tooltip* [6]**

This script as it's name implies provides the tooltip functionality used in all the windows that the user can use to add a component.

#### ***colorPicker* [7]**

This provides the color selection tool when adding overlay cells. It provides a visual way for the user to select the color they want and does not require them to know the exact format in which to enter the color.

## 4. Usage of data-driven Ximpel playlist builder

In this chapter I will provide a tutorial on how to use the playlist builder. I will start with an example of how a user might create the playlist for a simple interactive video. I will then give step by step explanation of how to use the playlist builder accompanied by screenshots were necessary.

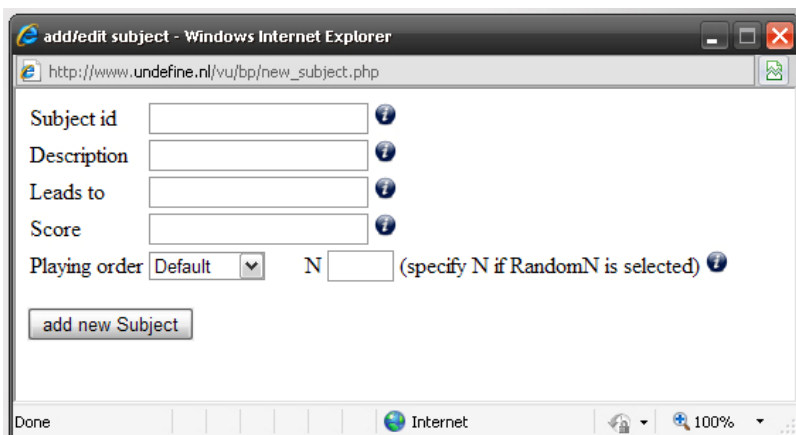
### 4.1. Tutorial

I have chosen to present the information to the user as a collection of lists. To keep things as structured and straightforward as possible new options become available after each step in the following order:

Action	Options that become available
Initially	Upload existing xml playlist, Create subject
Creating subject	Adding videos
Add video	Adding question, Adding overlays
Add Question	-
Add Overlay	Adding overlaycells
Add Overlaycell	-

The only option available to the user in the beginning is to add a subject. This can be done by clicking the large green button titled “add Subject”. This green button with a different text appears on every page in the playlist builder and is used for adding each of the different objects in the project.

When the user clicks the add Subject button the following screen appears.



The screenshot shows a web browser window titled "add/edit subject - Windows Internet Explorer". The address bar shows the URL "http://www.undefine.nl/vu/bp/new\_subject.php". The form contains the following elements:

- Subject id:
- Description:
- Leads to:
- Score:
- Playing order:  (dropdown menu)
- N:  (specify N if RandomN is selected)
- add new Subject:

screenshot 1

Once the user has filled in all the different fields the “add new Subject” button is clicked and the subject is added to the project and appears in the subject list which can be seen in the

following picture (screenshot2).



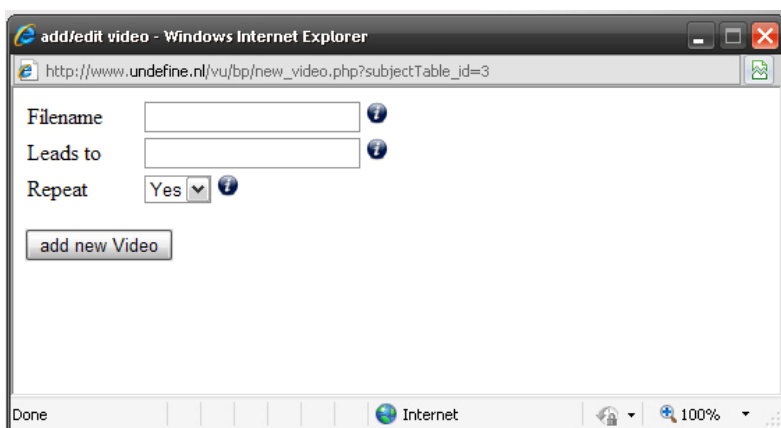
screenshot 2

Once there is a subject available the user can choose to edit or delete the subject and also to continue and add videos to it. When clicking the Videos link the following page appears.



screenshot 3

Here the user can add a video by clicking on the green button now titled "add Video". This will make the following window appear.



screenshot 4

The user fills in the desired fields and clicks the "add new video" button.

A new video will then appear in the video list and provide the user with a choice to add a

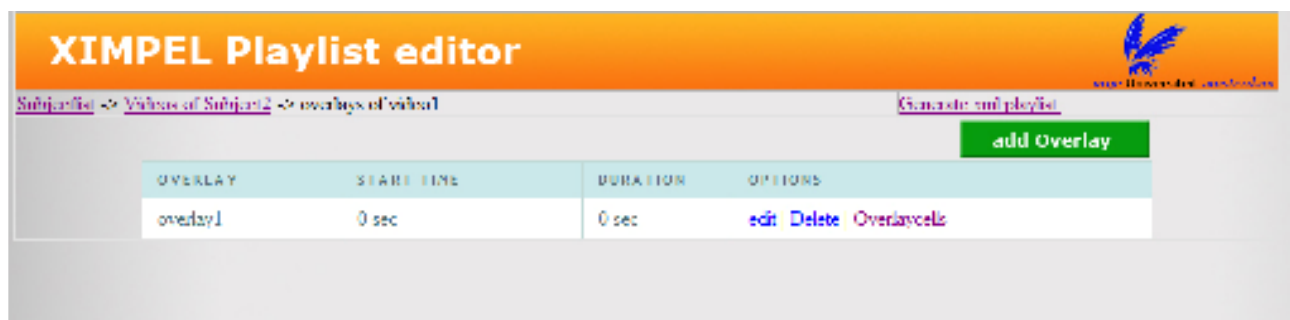
question or an overlay (screenshot3).

Clicking the questions link leads to the question list and the option to add a question as shown below. Clicking add question will bring up a window similar to the window of the add subject and add video buttons.



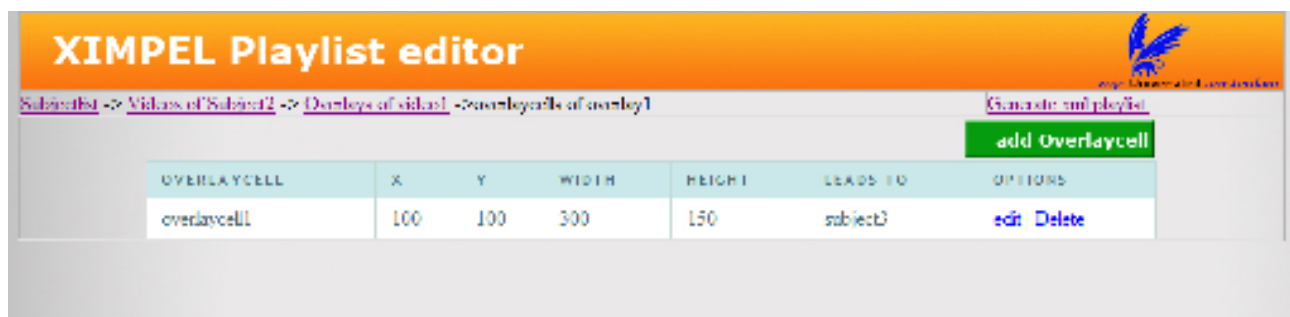
screenshot 5

Clicking the overlay link leads to the overlay list and the options to add an overlay as shown in screenshot6 below.



screenshot 6

When an overlay is added, the final option becomes available and that is the option to add an overlay cell. Not adding an overlay cell to an overlay is actually pretty useless, as the overlay has no purpose then. Clicking on the overlay cell link will lead to the final page with the option to add one or more overlay cells (screenshot7).



screenshot 7

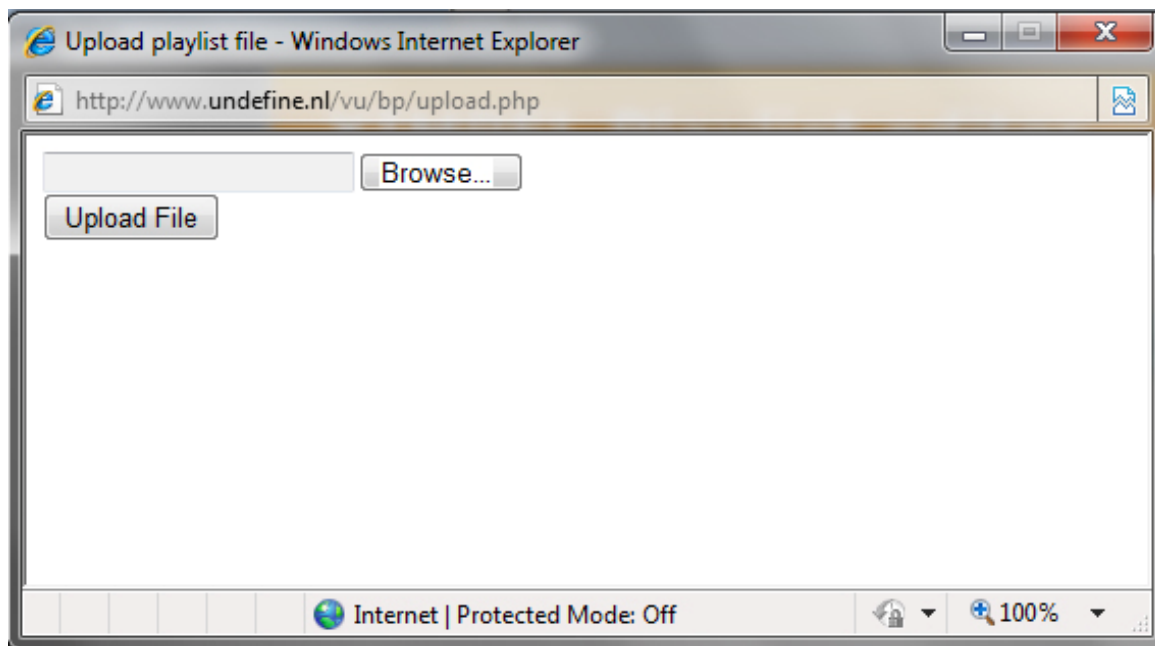


At any point during the process the user can click on the “generate xml playlist link which is visible on all pages. When this link is clicked a new window will open with a debug message and a link to download the created xml file. No matter what kind of debug information is displayed, the xml playlist will always be created with the information available in the database. It is up to the user to determine if the information that was added to the database is correct. For instance it is allowed to have overlays without any overlay cells within them. The debugger will give a warning about this but it will not stop the creation of the xml file.

The reason why I decided not to impose hard restrictions or requirements for creation of the xml file is because I wanted to provide the user with as much freedom as possible. Users will probably come up with ways to use Ximpel that I have not even thought of yet and I wanted to try and keep all options open if they decide to use this playlist builder for new creative ideas about Ximpel.

### ***Uploading existing playlist xml***

When there are no subjects created yet, the user is offered the option of uploading an existing playlist xml file. The xml file that will be uploaded must be in the format that Ximpel uses since June 2009; this is because certain tags have different names and attributes prior to the June 2009 release.



## Conclusion

After completing the development of the playlist builder application and having it tested by some novice and experienced computer users a couple of conclusions can be reached. The functional requirements have been achieved in the sense that the playlist builder provides the user with all the possibilities that manually editing the xml file would. The application also serves as an example for other developers to expand the usage of data driven ximpel applications. By expanding my application developers could provide users with more dynamic content in interactive videos. By using the playlist builder as a “administrative backend” the interactive videos can be more easily distributed on websites because changing data used in the interactive video can be easily changed or expanded.

The non functional requirements were partly reached in the sense that maintainability should be adequate. Usability of the application itself is also satisfactory but the user will still need moderate knowledge of Ximpel itself before he can fully understand and use all the options in the playlist builder.

## References

**[1] *Playlists in XIMPEL***

Retrieved June 22, 2009, from Ximpel website  
[http://ximpel.few.vu.nl/tutorials/Playlists\\_in\\_XIMPEL.html](http://ximpel.few.vu.nl/tutorials/Playlists_in_XIMPEL.html)

**[2] *Configuration files in XIMPEL***

Retrieved June 22, 2009, from Ximpel website  
[http://ximpel.few.vu.nl/tutorials/Configuration\\_files\\_in\\_XIMPEL.html](http://ximpel.few.vu.nl/tutorials/Configuration_files_in_XIMPEL.html)

**[3] *Overlays in XIMPEL***

Retrieved June 22, 2009, from Ximpel website  
[http://ximpel.few.vu.nl/tutorials/Overlays\\_in\\_XIMPEL.html](http://ximpel.few.vu.nl/tutorials/Overlays_in_XIMPEL.html)

**[4] *Using Ajax to validate forms. (2007)***

Retrieved July 7, 2009, from jquery for designers website  
<http://jqueryfordesigners.com/using-ajax-to-validate-forms/>

**[5] *Dynamic Drag'n Drop With jQuery And PHP.***

Retrieved July 16, 2009, from webresources depot  
<http://www.webresourcesdepot.com/dynamic-dragn-drop-with-jquery-and-php/>

**[6] *JavaScript, DHTML Tooltips***

Retrieved July 22, 2009, from www.walterzorn.com  
[http://www.walterzorn.com/tooltip/tooltip\\_e.htm](http://www.walterzorn.com/tooltip/tooltip_e.htm)

**[7] *Color picker***

Retrieved July 24, 2009, from dhtmlgoodies.com  
<http://www.dhtmlgoodies.com/index.html?whichScript=submitted-color-picker>

## Appendix 1: example of a ximpel playlist.xml

Playlist example 6: Linking subjects

```
<?xml version="1.0" encoding="utf-8"?>
<ximpel>
  <subject id="overlayJump">
    <description>Jump by overlay</description>
    <media>
      <video file="jump1" repeat="true">
        <branchquestion>Choose your jump</branchquestion>
        <overlays>
          <overlay>
            <overlaycel x="0" y="0" width="300" height="600" leadsto="mediaItemJump"/>
            <overlaycel x="500" y="0" width="300" height="600" leadsto="subjectJump"/>
          </overlay>
        </overlays>
      </video>
    </media>
  </subject>
  <subject id="mediaItemJump">
    <description>Jump by media item leadsto attribute</description>
    <media>
      <video file="jump2" leadsto="overlayJump"/>
    </media>
  </subject>
  <subject id="subjectJump" leadsto="overlayJump">
    <description>Jump by subject leadsto attribute</description>
    <media>
      <video file="jump3-1.mp4"/>
      <video file="jump3-2.mp4"/>
    </media>
  </subject>
</ximpel>
```

## Appendix 2: example of ximpelConfig.xml

```
<ximpel>
  <settings>
    <playlist>xml/fast.xml</playlist>
    <assetsDir>images/</assetsDir>
    <videoDir>video/</videoDir>
    <extraQuestionDuration>8</extraQuestionDuration>
    <scoreMultiplier>1000</scoreMultiplier>
    <extraQuestionScore>0.5</extraQuestionScore>
  </settings>
  <scoreMessages>
    <scoreThreshold upperlimit="1" message="That's really bad..."/>
    <scoreThreshold underlimit="3" message="Well done!"/>
    <scoreThreshold underlimit="1" upperlimit="2"
      message="Thanks for playing! Try again."/>
  </scoreMessages>
  <language>
    <questionLabel>Question</questionLabel>
    <subjectLabel>Subject</subjectLabel>
    <scoreLabel>Score</scoreLabel>
    <extraQuestionRightLabel>Right!</extraQuestionRightLabel>
    <extraQuestionWrongLabel>Wrong!</extraQuestionWrongLabel>
    <extraSegmentLabel>Extra footage!</extraSegmentLabel>
    <nextButtonLabel>Next</nextButtonLabel>
    <playButtonLabel>Play</playButtonLabel>
    <playAgainButtonLabel>Play again</playAgainButtonLabel>
  </language>
  <titleScreenImage>title.jpg</titleScreenImage>
  <titleScreenHeader>XIMPEL Application</titleScreenHeader>
  <instructionScreenHeader>
    Instructions Interactive Video Application
  </instructionScreenHeader>
  <instructionScreenHTMLText>
    <UL>
      <LI>Watch the video</LI>
      <LI>Make choices by clicking in the video</LI>
      <LI>Find hidden segments to score bonus points</LI>
      <LI>Answer extra question correctly for more bonus points</LI>
    </UL>
  </instructionScreenHTMLText>
  <evaluationScreenHeader>Your storyline</evaluationScreenHeader>
  <evaluationScreenScoreLabel>Your score</evaluationScreenScoreLabel>
  <evaluationScreenSubjectLabel>Played subjects</evaluationScreenSubjectLabel>
</ximpel>
```