An investigation of new distribution forms for traditional media online

Markus Näsman
Abstract

An investigation of new distribution forms for traditional media online

Markus Näsman

For decades watching the news, movies or television series has been something you do in front of your TV. But things are changing, more and more TV-channels make parts of their playlists available online. But exactly how much broadcast material is available online? And is it possible to collect all available broadcast material on a single web portal and enable users not only to watch but also to record this material? The objectives of this thesis were to investigate how much and in which form different TV-channels make their broadcast material available online. The investigation results was used in constructing an online web portal prototype. This portal contains TV-channel playlists with available broadcast material as well as the ability to record this material, if possible.

The investigation gave that most Swedish TV-channels has very little broadcast material available online. Swedish TV-channels that had enough material available to be considered for the prototype were SVT1, SVT2, SVT24, ZTV and The Voice. A few foreign TV-channels where also investigated and included in the prototype. These were RUV TV, BBC 24, Direct8, Sky News, and Bloomberg UK. Investigation on the recordability of the available broadcast material gave that VideoLAN Client was to be used. Since it worked well with Windows Media Video streams, which was the format most commonly used.

The prototype was developed using the open source tools Python, Django, MySQL, Apache, at, xmltv tv-grab and VideoLAN Client. The prototype implements the ideas of the thesis for the following TV-channels: SVT1 SVT2 SVT24, Direct8, Bloomberg UK, RUV TV, Sky News, BBC24, The Voice and ZTV. A recording feature is in place in all cases where the stream is playable in VideoLAN Client.

Handledare: Daniel Byström
Ämnesgranskare: Mikael Pettersson
Examinator: Anders Jansson
IT 08 046
Tryckt av: Reprocentralen ITC
Part I
Preface

This paper is written as a 15hp bachelor thesis in computer science at the Institute of Information Technology, Uppsala University. The thesis is written to address and implement an idea by Red Bridge AB.

Part II
Table of contents

Contents

I Preface 1

II Table of contents 1

III Introduction 3
1 Background 3
2 Objectives 3
3 Limitations 3
4 Technical requirements 3

IV Analysis 3
5 Pre-study 4
5.1 Tools 4
5.1.1 Recording 4
5.1.2 TV-Channel playlists 5
5.2 Available online TV broadcast material 5
5.2.1 Requirements 5
5.2.2 Investigated channels 5
5.2.3 Conclusion 7
5.3 Conclusion 7

6 Prototype 7
6.1 Web framework 7
6.2 Software design overview 8
6.3 System components overview 9
6.3.1 URL Dispatcher 9
6.3.2 View ................................................. 9
6.3.3 Template ............................................. 9
6.3.4 Model .................................................. 9
6.3.5 Channel Application ................................. 10
6.3.6 Recording Application .............................. 11
6.3.7 Recorder .............................................. 13

6.4 System component indepth look ....................... 14
  6.4.1 RUV TV interface ................................. 15
  6.4.2 SVT1 interface ..................................... 16

6.5 Result .................................................. 18

6.6 Known prototype problems ............................. 20
  6.6.1 Time storing ....................................... 20
  6.6.2 Daylight saving .................................... 20
  6.6.3 TV-show updates for Non-CS channels .......... 21

7 Conclusions ............................................. 21
Part III

Introduction

1 Background

Red Bridge AB wants to develop a online service for distribution of traditional TV-broadcasts. They want to use already publicly available online sources for TV-broadcasts. They want to do this using open-source software. The available online services that already exists on the subject are:

a) Individual TV-channels providing either part of their playlist or the full playlist as streaming media.
b) Websites listing TV-channels who provide the streams mentioned in a).
c) Websites providing just the TV-channel playlists without any broadcast media.

The belief is that there are still unexplored ways of exploiting the current situation by making a website combining a), b) and c).

The idea is making a web portal where TV-channel playlists can be viewed. In each playlist all TV-shows having broadcast material available online will be watchable as well as recordable, if possible.

2 Objectives

There are two main objectives:

1) Investigating how well the the idea can be carried out considering the current situation.
2) Implement a prototype of the idea with the information gathered in 1)

3 Limitations

The graphical design of the web-page and such is not to be considered. Neither are legal issues to be considered. Very little time will be spent on evaluating different programming languages, frameworks and such since time is limited and there is already a stack of software preffered by Red Bridge AB.

4 Technical requirements

As Red Bridge AB is working mainly with open source software one requirement is that the prototype is to be built using open source software and frameworks. The software stack to be used as specified by Red Bridge is Linux, Apache, MySQL and Python unless changing any of these components would realize significant gains.
Part IV
Analysis

5 Pre-study

The pre-study can be divided into two categories: evaluation and exploration. Where the first is evaluating the possibilities regarding the available TV-broadcasts, and the second is examining different third-party tools and frameworks that can be used. About 4 weeks was spent on the pre-study. This included choosing and experimenting with different tools and finding TV-channels that might be of interest. Included in these 4 weeks was also investigating the found TV-channels and learning the concepts of the Python language and Django Framework. Before the pre-study was finished another 1 week was spent documenting the pre-study.

5.1 Tools

5.1.1 Recording

One feature watching TV-broadcasts on a regular TV is that you can use a Videocassette recorder (VCR) to record a show that you for some reason want to see again or a show you want to watch at a different time than the broadcast time. This is a feature that the existing online TV-services somewhat lack, and we are going to examine the possibility to change that. The format for video streams on the internet are almost always either Windows Media Video (WMV) or RealMedia (RM) format. Considering streams in these two formats research gives two open source candidates: VideoLAN Client (http://www.videolan.org/vlc/) and Mplayer/Mencoder (http://www.mplayerhq.hu/) for the task of acting as a VCR.

VideoLAN Client (VLC)

Works well for most streams in WMV format. With the exceptions of Windows Media Video version 10 (WMV 10) streams with Digital Rights Management (DRM) or WMV streams where the audio is encoded with a fairly new Windows Audio format. Streams in RM format doesn’t work at all. Scheduling of stream recording is, according to the manual, possible through the telnet- or web interface. Experimenting with the built-in scheduler was unsuccessful and was determined not to be of any use. VLC is considered to be of interest for the project because of it’s fine WMV recording abilities.

Mplayer/Mencoder

Works with most streams of WMV format, with the exception of WMV 10 streams with DRM. Works with most RM streams except where DRM is in place. However the recorded video file tends to be out-of-sync with the audio and in some cases the video is choppy. This was the case for both recorded files of RM and WMV streams. No solution to this problem could be found within
a reasonable amount of time and therefore Mplayer/Mencoder was determined to be of no interest to the project at this moment.

5.1.2 TV-Channel playlists

It is essential to always carry the latest playlists for the TV-Channels that are being handled. Research gives a name of a open source tool which can provide this kind of functionality for us.

xmltv tv_grab

Generates a xml-file using the xmltv-format*. A number of different TV-channel playlists are available for most larger nations in the world as well as a few smaller ones, including Sweden. This tool was considered to be of great value since it can provide playlists for a large amount of TV-channels in a easy to use format without the need for developing a new tool for each TV-channel found.

* xmltv-DTD: http://xmltv.cvs.sourceforge.net/xmltv/xmltv/xmltv.dtd?view=markup

5.2 Available online TV broadcast material

The most important step in the pre-study is to investigate exactly how much TV-broadcasts that are available in their full and in which form. Another important part of the pre-study is to explore the possibility to write automated interfaces to find broadcast material for playlists of interest. The investigation mainly focused on the big Swedish TV-channels, as well as a number of non-Swedish TV-channels.

5.2.1 Requirements

For a channel to be of our interest a mayor part of it’s playlist’s broadcast material should be available online and the content should preferably be recordable using VLC. Exceptions to the requirement of recordability can be made if a really large part of a channel’s playlist is available online. Furthermore the available broadcast media should be full episodes, not short clips.

5.2.2 Investigated channels

SVT1/SVT2/SVT24

A large portion of their in-house material is made available online directly after broadcast or when the broadcast has started. The format of the video streams are either WMV or RM, and in many cases both formats are available. In the case where there is a WMV stream, this stream works well with VLC and can thus be recorded. The possibility to in a automated way access the video streams for today’s playlist are decent; there are two different way’s that SVT uses to mark that there is a video stream for a specific TV-show. There has to be work done to find patterns describing these two different ways.
TV4
Most video streams offered by TV4 are just short clips, with the exception of their news broadcast. They have a service called TV4Anywhere, which is built upon a flash-platform, where more TV-shows are available online. However TV4Anywhere is a paid service. The possibility for accessing the video streams for today’s playlist in an automated way are extremely small because of the flash-platform used.

TV5
A very small amount of TV5’s in-house material is made available online some days after the broadcast. The video stream is a WMV 10 stream with DRM, which means no recording is possible using VLC. The possibility to automate the process of acquiring the video streams for today’s playlist are good.

TV3/TV6
Quite a lot of broadcast material is available online. However most of the material is really old. The video streams made available are in WMV format and works well with VLC.

ZTV
The whole playlist is made available as a WMV stream that runs all day. The stream works well in VLC.

The Voice
The whole playlist is made available as a WMV stream that runs all day. The stream works well in VLC.

RUUV TV
A rather large portion of the playlist has material available online. These streams are made available on broadcast or directly afterwards. The streams are in WMV format, and work well with VLC. Accessing the video streams for today’s playlist in an automated way can be done really easy.

Sky News
The whole playlist is made available as a WMV stream that runs all day. The stream works well in VLC.

Bloomberg UK
The whole playlist is made available as a WMV stream that runs all day. The audio does not work in VLC, so no recording is possible.

Direct8
The whole playlist is made available as a WMV stream that runs all day. The stream works well in VLC.
5.2.3 Conclusion

The TV-channels that were considered interesting were this either because they had a decent amount of recordable broadcast material available online. Or they had a really large part of their playlists available online. These TV-channels are: SVT1/SVT2/SVT24, ZTV, The Voice, Bloomberg UK, Direct8, RUV TV, BBC 24 and Sky News.

5.3 Conclusion

From the study above we can conclude that there is still a limited amount of TV-channels that offer a interesting amount of broadcast material available online. For an example out of the Swedish TV-channels being examined only the SVT-network, ZTV and The Voice were found interesting enough to actually be included in the prototype.

The examined channels that where considered interesting and that we will attempt to include in the prototype are: SVT1, SVT2, SVT24, ZTV, The Voice, RUV TV, Direct8, Bloomberg UK, BBC 24, Sky News.

The examination of the tools gave us the result that if we are to have a recording feature we are to use VLC. We concluded that only WMV streams not containing DRM and/or the latest version of Windows Audio, can be recorded. We also concluded that scheduling of recordings can not rely on any built-in functionality in VLC, we have to write our own. There is also the decision to use the xmltv xml-format for getting TV-channel playlists into our application. The tv_grab application will be our main source for these playlists, only writing our own playlist parser when tv_grab can not supply a playlist for the TV-channel in question.

6 Prototype

The purpose of the prototype is to demonstrate how the chosen open source tools can be combined with the TV-channels that where determined to be interesting to create a web TV-portal. For the development of the prototype a few things needed to be taken care of: as the language of choice is Python, a open source web framework for Python is needed. As the focus of this thesis is not to evaluate which Python web framework that is top of the line, no extensive research was put in to the matter. Below follows the choices made with a short motivation. Development of the prototype lasted 4 weeks and yielded about 250 lines of webtemplates and about 2000 lines in application code. In addition to this 1 week was spent documenting the prototype work.

6.1 Web framework

As web framework Django (http://www.djangoproject.com/) was chosen. The reasons for this being that Django is a framework used in actual production of several mayor sites. And it uses a variant of the Model-View-Controller (MVC) method, which is a favourable way of designing web applications. Last but not least Django uses the same license for all it’s components which means
no license-hustling which sometimes are the case when a framework consists of parts developed by different teams. Its main competitor was TurboGears, which is a very similar MVC-based framework. This advantages of Django is it’s autogenerated admin interface as well as the fact that’s it’s solid monolithically developed. The advantages of TurboGears is that it’s faster to catch on new technologies because it consists of different components developed by different teams. In the end Djangos advantages where considered more valuable.

6.2 Software design overview

The system consists of 6 different components: the URL dispatcher, the Model, the View, the Template, the Recording application and the Channel application. Each component interacting with one or more of the others. Using the Django web-framework, the system implements a variant of the Model-View-Controller (MVC) design pattern, called Model-View-Template (MVT). In this design pattern business logic, data persistence and data presentation are separated in different modules.

![Diagram of software design overview](image)

Figure 1

In figure 1 we see the scenario when a user types in a URL in his web browser. First the URL Dispatcher examines the URL and determines which View to be used and then calls that View. The View kicks in and does any data fetching or storing using the Model. The Model gets the data/storing/fetching requests and translates them into SQL statements and in the case of a fetch returns the result to the View. The View then does any logic needed and hands over the data to a suitable Template. The Template presents the data by generating a HTML page which is handed back to the web browser.
In figure 2 we see how the Channel Application feeds the database with information, which is later to be retrieved by a View. The Channel application will run at a interval, each time updating the database with information of today’s TV-shows and their associated broadcast material. A more detailed explanation of how the Channel Application works can be found in section 6.3.5.

In Figure 3 we see that the recording application schedules the recorder using at*. The recorder uses VLC to make the recording happen. Information on which shows to record are fetched from the Model. Which shows to be recorded is inputted by the user(s) on the webpage. A more detailed explanation can be found in section 6.3.7. In the following section we will refer to TV-channels with a video stream that runs all day and covers the whole playlist as CS Channels (Constant stream channels) and all other TV-channels as Non-CS Channels (Non-constant stream channels).

* at is a very simple basic *nix tool included in most *nix distributions used to schedule commands.

6.3 System components overview

6.3.1 URL Dispatcher
Connects a certain URL with a certain View.

6.3.2 View
Describes what data the user will see.

6.3.3 Template
Describes how the data will be presented to the user.

6.3.4 Model
Database layer, handles data persistence.
6.3.5 Channel Application

Updates TV-channel information. For each Non-CS TV-channel being handled by the system there is a separate module, handling the TV-show broadcast material links for that particular TV-channel. This module is called a **interface**. The workflow for the Channel Application is the following:

1. First we download a XML-file in xmltv-format with playlist information for all TV-channels. This XML is created by tv_grab and some custom playlist parsers.

2. We parse the XML-file building a parse-tree.

3. From the database through the Model, for each TV-Channel we fetch the following information: if it is a CS or Non-CS channel, channel interface name, what string identifies the channel in the XML and when the playlist starts for that particular channel (start of the day).

4. For each channel that is a Non-CS channel we import the interface and from it we get information regarding broadcast material for all the shows in the channel's current playlist.

5. For each channel that is a CS channel we get the video stream information by determining which is the current show in the playlist, marking it as having broadcast material available online. For all shows that already has been broadcasted we mark as no broadcast material available. If the channel has recordingPossible set to true we mark all shows which has not been broadcasted yet as recordable but not yet available.

6. We traverse the tree and for each entry where we find a string that identifies a certain channel we add show data to the database through the Model, getting show name and start time from the XML and getting broadcast material from the interfaces.

In figure 3 we see a sequence diagram describing how this works in the case where we consider only one Non-CS TV-channel. However the procedure can be used for multiple TV-channels by, for each channel, repeating the sequence after parsing the XML file.
6.3.6 Recording Application

Starts recording of recordings, deletes old recordings and informs users when a recording is available. Scheduling of the recorder is done using the at tool.
since the scheduling functionality in VLC was determined to be of no use. The recorder uses VLC to record the stream to disk and convert it into a suitable format, for more information see 6.3.7.

**Non-Constant Stream channels**

Assuming we only deal with non-constant stream channels we get the following flow, for each recording not available:

For each recording available we do the sequence bellow where “schedule recording” is the same as starting the recorder immediately:

---

For the non-constant stream channels, the flow chart depicts the steps for each recording available or not available. If a recording is not available, it proceeds to check if the recording finish time reached and recording in progress, marking it as not in progress if true. If not, it checks if recording is possible, scheduling the recorder for the recording if true. For each recording available, it checks if the recording time of death reached, removing the recording file and database entry if true, otherwise sending notices to all users registered with the recording.
**Constant Stream channels**

For each recording not available we do the same as in Figure 5. For each recording not available we do the following, where “schedule recorder” means scheduling the recorder to start when the TV-show starts.

![Diagram](image)

Figure 7

### 6.3.7 Recorder

The recorder works in the following way (see Figure 8):

1. Calculate the length of the TV-show
2. Start VLC to dump the TV-show video to disk.
3. Sleep for the length of the TV-show
4. Send a kill signal to VLC
5. Mark recording as available in the database
6.4 System component indepth look

To get a better understanding on how the video stream information is gathered here comes two examples for how two Non-CS channel interfaces works underneath. These two examples gives us two cases: one where everything works easily after a single pattern and the other where a bunch of different patterns are required to be examined. These interfaces gather their information by parsing TV-channel home pages, mostly using regular expression and in some cases by
using more simple string operations.

6.4.1 RUV TV interface

In the case of RUV TV we have a playlist located at http://dagskra.ruv.is/streaming/sjonvarpid/ where each TV-show in today’s playlist has an entry stating show name and start time. If there will be a video stream made available for the show it is marked with an icon, and if there is a video stream available for the show it is marked with another icon as well as a link to the stream. This is an example on how a TV-show entry where a video stream will be available might look in HTML:

```html
<tr valign="top">
<td>StartTime</td>
<td><img src="/files/images/streaming/gray/wmp.gif" alt="" /></td>
<td class="description"><span title="">Show name</span></td>
</tr>
```

In the first `<td>` we see the start time for the show, in the next show we see the `<img>` which indicates that there will be a video stream available for the show. In between the `<span>` tags in the third `<td>` we have the Show name. If a link is available for the show the `<img>` in `<td>` will be another image and will be surrounded by `<a>` with a link to the video stream.

We can easily extract information regarding the availability of a video stream for a link by making a simple regular expression.
By examining the second match group we can determine if there will be a stream available or if there is a stream already available.

This interface is a good example when everything is easy and smooth, as all TV-shows for which there exists or there will exist a video stream follows the same simple pattern which can be captured by the above regular expression.

6.4.2 SVT1 interface

In the case of SVT we have a playlist for today on http://svt.se/svt/jsp/Crosslink.jsp?d=8744&lid=tv-guide. Each show has an entry on the form:

```html
<tr class="...">
<td class="time">StartTime</td>
<td class="content"><h2><a title="Visa / Dölj information" href="javascript:SgToggleObjects('...');">Show name</a></h2>
<div style="display:none;" id="...">
<p>Show description</p>
</div></td>
<td class="icons"><a title="Till Hemsidan" href="homepageurl"><img src="somegif.gif" alt="Till Hemsidan" width="20" height="20" /></a>
<a title="Se Video" href="videourl.htm","largevideoplayer",790,600,'scrolling=no,resizable=no,status=yes')"> <img src="someothergif.gif" alt="Se Video" width="20" height="20" />
</a></td>
</tr>
```

As we see in the third <td> there is a description for the show, this description might contain information on which episode the show is or which was it’s original
air date. We can extract the episode number by using the following regular expression on the show description:

```
Del.(\d+).av.(\d+)
```

In the same way we can get the original air date using the regular expression:

```
Fr.n.(\d+)/(/\d+)
```

This information is not always given, but when given it is essential for us, but more on that later.

If there is any video for the TV-show there will be a link in the `<td class="icons">` with the title “Se video”. This link takes us to a framed page consisting of two different frames, here called “video page” and “navigation page”. On the “video page” most oftenly there will be a link to a video stream, however it is not always the case that it is actually the same episode as the playlist tells. This is why the air date and episode number from the playlist is so important, we don’t want to link a TV-show to a video stream of the wrong episode. However we cannot always use the airdate/episode number to determine if it is correct or not, since it is not always given and also different TV-shows uses different patterns for naming their episodes. So we have to focus on a few widely used patterns and ignore the rest to make the interface somewhat maintainable.

Example on how a video stream entry looks on the “video page”:

```
<ul class="media" title="Mediafiler">
<li class="video">
<a href="videostreamurl.asx">videostreamurl.asx</a> (VideoFormat)
</li>
</ul>
```

Example on how a episode entry might look on the “navigation page”:

```
<tr id="tr3">
<td id="li3" class="link">
<div class="video">
<a target="cont" title="" href="videopageurl" onclick="chC(3);shL('cont');">Show name arbitraryInformation</a>
</div>
</td></tr>
```

Here are few widely used patterns to determine if a video stream is of the right episode or not.

**Date pattern #1:**

The “video page” will be searched for video stream entries of the form “somethingYYMMDDsomething.format”. If such is found we will assume that a video stream will be available in the future, because it seems like it follows a pattern. Then we check if this YYMMDD is the same as the air date extracted from the show description, if that is the case we have found the video stream. If no air date was found in the show description we use today’s date and repeat the just mentioned.
**Date pattern #2:**

As date pattern #1 with the exception that we look for a date on the form “somethingYYYY-MM-DD” and follow the same procedure as in Date Pattern #1.

**Weekday pattern:**

Here we will look at the “navigation page” to see if we can find an entry with today’s weekday as arbitrary information. If it is found then we assume that a video stream will be available, and if it’s past 18:00 we will assume that we have found the correct video stream. This pattern seems to be used only for the shows Uutsiet and Oddasat, whose video streams for today are available at 18:00.

**Episode pattern:**

Here we look at the “navigation page” to see if we can find an entry matching the regular expression:

```
href="(["\"]*)\.*oncl\.*\.(del|avsnitt|Avsnitt|Program) (\d:\d|\d)\.*\</a>
```

If the regular expression is found we assume that a video stream will be available. And if the third match group matches the episode number extracted from the show description we have found the correct video stream.

If none of the above patterns are found and the show isn’t listed in our “does not provide full episodes” list* and the word “trailer” isn’t present in the video stream link on the “video page” we just grab the first video stream entry and use it as the video.

As we can see there are several problems with SVT1, there is no real standard on how to determine if an episode is correct or not. As far as the investigations can tell there isn’t any clashes between the different patterns but it is not unthinkable that there might be a show with the name “Monday” which is aired on a Monday and has it’s episodes named by the “episode pattern”. If we check the “weekday pattern” first the system will believe it is a “weekday pattern” day and thus give the wrong video stream in this scenario.

* A list of SVT shows that only provides trailers or short clips. This list had to be introduced since there seemed to be a pattern that some TV-shows always were short clips and some always were full episodes without any good method to separate these from each other.

### 6.5 Result

The resulting prototype has been tested and is verified to work with the Non-CS channels SVT1/SVT2/SVT24 and RUV-TV through their interfaces. There is also support for the CS channels BBC 24, The Voice, ZTV, Sky News, Bloomberg UK and Direct8. The XML file which provide playlist information
for all the channels is generated by tv_grab, except for the entries for Direct8, The Voice and RUV-TV. For these there are specific playlist parsers and XML-generators, because they do not exist in tv_grab. For all video streams playable in VLC there is a recording option. Each TV-show being recordable is marked in the playlist with a icon, clicking the icon will prompt the user to enter their e-mail address. If a recording order already exists for the TV-show the user e-mail is added to that recording order. If there is no such recording order one is created and the user e-mail is added to it. When the recording is finished a e-mail with a URL to the recorded stream is e-mailed to the user. The recording is deleted after 48 hours.

**User view:**

Here we see the user interface, different channels and their playlists. TV-shows with a video stream is marked with a link and shows that are recordable are marked with a red *.
Admin view:

The admin view is basically the Django admin interface, which is a web interface for viewing, adding, deleting and editing database entries.

6.6 Known prototype problems

6.6.1 Time storing

The present situation is that all time information in the database is saved in UTC +2 (CET with Daylight saving) format. This is rather inappropriate. All times should be saved in UTC format for easy conversion to the time-zone the user is located in.

6.6.2 Daylight saving

The playlist parsers and xmltv generators for RUV-TV (ruvxml.py), Direct8 (direct8xml.py) and The Voice (thevoicexml.py) does not handle daylight saving, and will therefore give wrong starting and ending times when daylight saving is changed.
Daylight saving is not handled by the Channels Application (updateChannels.py) when the XML-file time information is converted to UTC+2 format. In the XML-file a start or end time has the form “YYYYMMDDhhmmss offset” where the offset tells how many hours plus or minus from UTC the time is. When converting the offset is subtracted from the time and then 2 is added to get the time in UTC+2 format, daylight saving is not considered and thus wrong starting and ending times will appear when daylight saving is changed.

The purposed solution is to use a package that implements the Python tzinfo() class correctly to handle the daylight saving when converting and creating the xml files.

6.6.3 TV-show updates for Non-CS channels

The current situation is that a update of the shows in the database is performed in the following way: today’s playlist is read from the XML-file and today’s playlists video stream information is gathered from the interfaces. This might give us the following problem:

A user requests a recording for a TV-show that, in the playlist, is marked as recordable but does not yet have a video stream available. The video stream for this TV-show is not made available until the next day (maybe the show is the last in the playlist for that day). This show will never be recorded because when the next update of video stream information is made it is on next day’s playlist and therefore it will be unknown to our system that a video link has been made available.

There are two purposed solutions to this:

1) Make sure that only TV-shows where the video stream is already available is marked as recordable even if we know that a video stream will be available later. This way the situation will never occur.

2) Make a change to the data model so that a TV-show entry contains information about episode number and is bound to several different start times, this way a recording will start if the TV-show has a re-run some day later.

7 Conclusions

Judging from the pre-study and the development of the prototype it can be concluded that there is a lot to be wished for; only a few of the mayor Swedish TV-channels actually offer a decent amount of their playlists’ broadcast material online. In many cases only short clips are offered, and often the TV-channel’s own web-page gives a hard time determining if a video stream is a clip or full episode. Usually it is also hard to determine which episode a available TV-show is. Hopefully more and more material will be made available online as time passes. When it comes to the recording feature it has a decent coverage of the examined TV-channels. Probably a even larger coverage can be seen in a not to distant future, as VLC gets support for what it’s lacking or a better suited tool than VLC emerges.

It can be said that the prototype that has been developed lives up to the expectations of what a service like this should be able to offer given the current
situation. Of course with the exception of the problems mentioned in section 6.6, but remember it is a proof of concept prototype, nothing more. There are of course some design choices that might be questioned. For an example the choice of using tv_grab and the xmltv format can be questioned. In the case where we have non-constant stream channels the work the interface does could easily be extended to parsing the playlist as well, thus making tv_grab redundant. And for constant stream channels the interface could easily just be a playlist parser that also handles the calculations for determining which show is the current and so on, which also makes tv_grab redundant. Making tv_grab redundant would make us less dependent on the third-party software tv_grab, which in some cases is poorly maintained. The drawback would be that it requires us to redo work that has already been done in tv_grab and in the case for constant stream channels it would duplicate a lot of code for no good reason.