# Embedded Speech Recognition In UPnP (DLNA) Environment

Jozef Ivanecký, Radek Hampl

European Media Laboratory
Schloss-Wolfsbrunnenweg 33, 69118 Heidelberg, Germany
{jozef.ivanecky, radek.hampl}@eml.org

**Abstract.** In the past decade great technological advances have been made in internet services, personal computers, telecommunications, media and entertainment. Many of these advances have benefited from sharing technologies across those industries. This influences how Digital Home Entertainment products are designed to follow the overall "Media Convergence" trend. Existing Universal Plug and Play (UPnP) or DLNA specifications are often used for these purposes. These specifications permit electronic devices to be simply plugged into home and local networks for access and exchange of shared data like music, video or photos. The number of media items in a user library can then easily exceed 10,000 elements. In addition, these specifications are used by manufacturers of consumer electronics to ensure interoperability of different consumer electronic devices.
In this paper, we describe our efforts towards introducing speech recognition to control electronic devices in UPnP (DLNA) environments. We give an overview of the content structure and media information available in the UPnP (DLNA) network. We also analyze the use of available information for speech recognition. The main focus will be on the possibility of designing and implementing a voice-enabled UPnP (DLNA) Control Point, and the introduction of one particular solution.

## 1 Introduction

In home entertainment today terms like MP3, AVI, MPEG, etc., appear very often. They indicate not only a new way of storing music, movies or pictures, but also gaining instant access to the entire music, video or picture selection no matter where they are stored. Distributed media content, as well as demand for interoperability among devices from different manufacturers, led to the Universal Plug And Play (UPnP) specification.

The goals of UPnP are to allow devices to connect seamlessly, and to simplify the implementation of networks at home (data sharing, communications and entertainment) and in corporate environments. This part of the UPnP specification is UPnP-AV (UPnP Audio and Video). The UPnP-AV standards have been referenced in specifications published by other organizations, including Digital Living Network Alliance (DLNA) and others.

The main components of the UPnP-AV (DLNA) network are [1]:

- **MediaServer** is a device that stores and shares digital media, such as music, pictures or movies.
- **MediaRenderer** is a device that is capable of rendering AV content. Examples of the use of a MediaRenderer include traditional devices such as TVs and stereo systems. Some other contemporary examples include digital devices such as MP3 players or picture frames. However, most of these examples are typically limited to render one specific content type (e.g. a TV typically renders video content).
- **ControlPoint** is a device that is capable of controlling the rendering of digital media streamed from MediaServers to a selected Media Renderer.

The main UPnP-AV (DLNA) components can be combined to form one physical device. For example, pairing a MediaRenderer with a ControlPoint allows one to browse the music content of the available MediaServers and request a MediaServer to start streaming selected files to the MediaRenderer for local playback. There are UPnP-AV (DLNA) components available for most operating systems and many hardware platforms. UPnP-AV (DLNA) devices can either be categorized as software-based or hardware-based. Software-based UPnP-AV (DLNA) devices can run on PCs.

The UPnP network can enable instant access to an entire digital media collection in today's homes. It is important to note that the digital content offered by a MediaServer is not sorted by the physical data structure on the disk, CD, etc., but by the media content itself. For example the music collection can be sorted by Artists, Albums, Genres, Years, etc. The content of such a database can easily exceed 10,000 items. Classical audio or video devices are usually equipped with relatively small displays. If they are acting as a ContolPoint, then selecting one item out of 10,000 can be cumbersome even if the data is well structured.

This is a challenge in which speech recognition can play a very important role. Selecting music or a movie from a very long list by voice is both fast and natural, but using voice control involves certain difficulties, such as automatic extraction of the music metadata, grammar generation, processing music or video metadata in other than the default language, intuitive dialog design, etc. In addition, the interface should be multi-modal, in order to allow standard interaction as well.

In this paper we describe our effort toward implementation of a Voice-enabled UPnP Control Point, which serves as a first step towards reaching these goals. The rest of the paper is organized as follows: In Section 2 we give a brief overview over the content of MediaServer and all the available information relevant for the Speech Recognition. In Section 3 we focus on the processing and usage of available information for Speech Recognition. Section 4 briefly sketches the structure of the ControlPoint, with the focus on multi-modality. Section 5 describes the experiments and Section 6 offers a brief summary.

## 2 MediaServer content

As mentioned above, the digital media on the MediaServer is not sorted based on file structure, but is based on the content. To understand the structure we first

take a closer look at the possibilities offered by different digital media file types. One particular media file type will be then used to explain the MediaServer data structure.

### 2.1 Digital media content

Digital media data today is stored in a wide range of file formats. Music may be stored in formats like mp3, wma, ogg, etc. Pictures may be stored in formats like jpg, png, etc. Video may be stored in formats like avi, mov, mp4, etc. These file types vary not only in compression or quality level, but also in ability to store additional information related to the digital content.

One of the most popular music formats is MP3. Besides offering good compression and quality, it also offers the possibility of storing metadata, such as title, artist, album, track number and other information about the file content. The most widespread metadata formats are ID3v1 and ID3v2, and the recently introduced APEv2. Important ID3 fields for MediaServer as well as for speech recognition are: *Title, Artist, Album, Year* and *Genre*. As will be demonstrated later, there is no language or text-encoding information that is particularly important for speech recognition. The metadata part of the MP3 file can be created from different sources. It can be taken from the original format, if that is available, obtained from one of the internet music databases like CDDB or FreeDB, it can be added or modified manually, or it can even be left empty. This suggests that metadata extracted from MP3 files is unreliable.

The most popular format for pictures has become JPEG; this is used as a default format for most digital cameras. The *Exif* part of JPEG contains no especially useful information for the MediaServer or for speech recognition. The *Exif* includes technical data about the picture. The only usable information is the time when the picture was created. A similar situation exists with the other graphic formats.

With digital video formats there is a similar situation as there is with JPEG or other picture formats. The most popular video format, AVI, stores some technical information about the video, but other than the time, none of this information is useful for either MediaServer or speech recognition. The recent MPEG-4 offers support for additional information, such as title, chapter name, etc. but it is still rarely used.

### 2.2 MediaServer content

The Content Creation Subsystem of the MediaServer extracts the information from the imported data into the internal Content Storage Database (CSD), where it is sorted based on the content obtained from the processed files or from other available sources. The database can then be browsed and searched using a standard UPnP interface and stream the requested content from a MediaServer to the desired MediaRenderer.

MediaServer data is organized as a directory structure of *item*s into a hierarchy of *container*s. Both container and item have several properties, like *id,*

*title, creator*, etc. The top-level root container structure is created automatically based on the content supported by the MediaServer. The title of the top-level containers may differ from one manufacturer to another; this can be true even for MediaServers from the same manufacturer. Figure 1 shows one of the possible top-level container structures.


**Fig. 1.** Top level container structure.

As seen in (Fig. 1), *Photos* and *Video* containers each have subcontainers. One of these subcontainers — *Directories* was added to this particular MediaServer only recently to satisfy demands for some reasonable content structure of the pictures. Without a directory container the pictures were browsable only by their names (*img_4572.jpg, img_4573.jpg, etc.*), or by their dates.

Based on the semantics of metadata in the CSD, containers can provide access to a particular item (media object like song, picture or video) by album title, artist name, etc. The title of the container or item is derived from the content data, if available. Otherwise, the real file name or directory name is used.

The presented MediaServer data structure is browsable or searchable via ControlPoint's GUI[1]. After selecting a particular container or item, the associated URI is sent to the MediaRenderer with the PLAY request.

[1] Usual interface for UPnP devices.

## 3 Speech Recognition

As our ControlPoint is intended to run on hand-held devices, we decided to use a grammar-based embedded voice recognizer and allow minimal configurability of the ControlPoint by the user.

We identified the following steps as being of paramount importance in preparing data for the voice recognizer:

- Automatic parsing of the container structure and finding the needed containers.
- Automatic generation of grammars for the recognition system.

We will describe each task below.

### 3.1 Container Structure

First, we need to locate specific containers and, based on names of their items, create lists of albums, songs, pictures, etc. As mentioned above, container structure depends only on the design of the MediaServer. From the mime type[2] of the top level containers it is possible to determine what kind of media data is stored in the container, but this is the only useful information. For example, the path to the container with the "all songs" list for two different MediaServers can be as follows:

- Audio/All Audio
- Music/All Music/Songs

A problem can be caused by a different language of the user interface used by different MediaServers. One MediaServer for different national markets may have the containers' names in different languages. For example, the pictures can be stored in the container *Photos*, and in another case in the container *Bilder*[3].

Manual configuration would not be an option in this case. Setting one MediaServer with music, video and photos would mean having to set at least 5 container paths manually. For more MediaServers in the network the number of set container paths would grow linearly. To automate this task we analyzed the container structure of a few different MediaServers and implemented a simple algorithm to locate the specific container for each requested list (*Album list, Artist list, etc.*). This solution is not robust, as another type of the MediaServer may not be parsed correctly, but UPnP-AV (DLNA) may have been designed for Graphic User Interface only.

### 3.2 Container Content

The second step is a grammar construction. All grammars addressing dialog issues were created and compiled statically. Grammars for containers and items

[2] Part of ProtocolInfo in the container properties.
[3] For German version of a MediaServer.

are generated dynamically, as the content may change when adding or removing media data.

To process multimedia data from an unknown source means dealing with multilingual data without any information about the language as described in 2.1. Multilingualism also brings encoding issues. Considering the fact that the metadata in digital media files can be modified manually also, the detection of the correct code page is an issue in itself. The music collection used for our testing was a compilation of collections provided by several people. It contains music in about 10 languages and 6 encodings. Since we are using a monolingual system for the recognition, it was necessary to filter and modify the input test to pass the pronunciation generator in case the word is not part of the pronunciation dictionary.

The correct pronunciation was the second critical issue in content processing. We are using a combination of a large pronunciation dictionary with manually tuned pronunciation for frequent words and automatic generation. For each unknown word in case of either manual or automatic pronunciation we have to ask three questions:

- What is the language of the unknown word?
- How would the word be pronounced by a native speaker?
- How would the word be pronounced by a non-native speaker who does not speak the language well?

Despite the fact that we are using a monolingual systems (German or English), automatic pronunciation for a well-known artist, or for some common words, also worked well for words from a second language. This was also true in cases of known composers of classical music. For lesser-known names, or for other languages, the automatic pronunciation (especially with the encoding filtering) was not very reliable. For example, a band *Helenine oči* with a simple "*other to ASCII*" filter will result in *Helenine oci*. The German automatic pronunciation system does a good job with this version. The result is something like: [*h e: l n= i: n @ ? o: ts I*]. This version will work for a native German speaker who very likely does not know *Helenine oči*, but can see the band name on the display in the case of a multimodal ControlPoint. Someone who knows the band will pronounce it as [*h E l E J I n E: O tS I*]. To get the pronunciation for the word *oči* is also simple. Because the phonetically Slovak grapheme *č* is the same as German grapheme sequence *tsch*, by replacing each *č* with *tsch*, the Slovak (correct) pronunciation will also be created. However, for the word *Helenine* a simple solution in the case of automatic pronunciation does not exist.

The third issue for the container content is the existence of duplicates. There are three types of duplicates detected so far. The first type is caused by the simple fact that different MediaServers can contain the same data. The second type can be caused by a typo or an encoding issue in the content name. For example, *Herbert Grönemayer* once in UTF-8 and once in ISO-1 will cause this type of duplicates. For the MediaServer these are different artists, and will create two separated containers for them in the Artist list. The third type of duplicates is that of "empty" song name records in the ID3. If no song name has been

entered, the usual default is *Track 1, Track 2, etc.*. If there are several such CDs on the MediaServer, then there are several songs *Track 1, Track 2, etc.*.

## 4 ControlPoint

Using more then one device, including a personal computer, as part of the home entertainment setup is more or less today's standard. This increases the complexity of such systems. In order to minimize the complexity, it is necessary to see the home entertainment system not just as a set of separate devices, but rather as a complex network where users do not need (and do not want) to know the details concerning data storage and distribution, or about the capabilities of each device, etc.

Designed and implemented, the Voice-enabled UPnP Control Point addresses these issues. It simplifies dealing with multiple devices connected to the network of a home entertainment system, and it eliminates the need to configure individual devices. After startup, all devices (MediaServers and Media Renderers) are automatically found. The full scan of the each MediaServer is performed during the next step. The scanned content from different MediaServers is merged together by categories. These categories currently are: *Artist, Album, Song, Photo, Video, Internet Radio*. The user can switch among the categories by voice command, and then select the desired item from the respective category. In case of *Artists, Albums* and *Photos*, the selected item is the container. The content of an entire container is played. For *Video* and *Internet Radio* one selected item is played. For the *Song* the selected song starts to play, and it continues with the next song on the list.

Song selection by voice is also possible within the previously selected container (*Artist, Album*). A set of commands is active independent of the current context like category switching and basic control (*next, previous, stop, random, etc.*). The main category lists are generated and compiled once after the MediaServers scan. The reason for this is a timing issue; the grammars for the required subcontainers are created and compiled on demand.

Beside several MediaServers, several MediaRenderers can be available as well. They usually have different capabilities (can render just certain media types) and can have different locations as well. The MediaRenderers configuration is the only manual procedure required from the user for correct functionality. For each media type the user should define a primary and an alternative renderer. After the *play* command without any target specification the desired content for a particular category is sent to the default renderer for that category. If some additional information is specified, e.g., *Play Keisers Orchestra in the kitchen!*, the alternative renderer is selected. The user may select any renderer via the GUI combo box based on his or her own preferences.

## 5 Experiments

To test and evaluate the implemented ControlPoint we created a UPnP network with two MediaServers and three MediaRenderers of different capabilities. The content of MediaServers was as follows: 1014 songs, 114 artists, 98 Albums, 28 picture directories and 57 videos. The title names were in 10 languages and 6 encodings.

The content scanning and grammar compilation, as well as the entire UPnP infrastructure needed for the speech recognition, was running as expected after some tuning. The most interesting aspect (for us) was information about how well speech recognition in such diverse environments would work. For testing purposes, 10 users were playing freely with the system and all their activities, as well as speech data, were logged. The stored logged data were later evaluated. It was possible to use 1278 recorded utterances for recognition evaluation.

The speech recognition test was performed on the Samsung Q1 portable device with far-field directional microphone. The talking distance was about $40cm$. For speech recognition we used 2 different embedded engines with a 16kHz German system. The results of both embedded engines were similar. The average sentence error rate was 27.46% (30.58% WER).

We analyzed the results to find the weakness of the system. As expected, the main issue is a title in other than the recognizer's language. The user, especially in case of English titles, is using English pronunciation. For this purpose the use of a bilingual system including English would be beneficial. This is true for today's popular music. In the case of classical or alternative music, a real multilingual system will be more useful.

## 6 Summary

In this paper we have described various aspects of the development of a voice enabled UPnP-AV (DLNA) ControlPoint. We have explored the UPnP-AV (DLNA) specification and analyzed the possibility to enable voice control of UPnP-AV (DLNA) systems, and have identified available and missing features of the environment for successful speech recognition implementation. Experiments in a real testing environment demonstrated the feasibility of our approach, but also unveiled the need for further research in language detection of written short text and multilingual speech recognition in order to create a system of acceptable accuracy in an environment not initially designed for a voice user interface.

## References


1. UPnP specification. Available online on http://www.upnp.org
2. J. Ivanecký, V. Fischer, S. Kunzmann. French–German Bilingual Acoustic Modeling for Embedded Voice Driven Applications. In *Proc. of TSD 2005*, Plzeň, 2005.
3. S. Kunzmann, V. Fischer, J. Gonzalez, O. Emam, C. Gnther, E. Janke. Multilingual Acoustic Models for Speech Recognition and Synthesis. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Montreal, 2004.