MASTER THESIS

HAVi COMPONENTS IN DIGITAL TELEVISION

November 15, 2001
Cesar Garcia, Pablo
Helsinki University of Technology
Department of Computer Sciences
Telecommunications Software and Multimedia Laboratory

INFORMATION ABOUT THE THESIS

Author:	Cesar Garcia, Pablo
University:	Helsinki University of Technology
Laboratory:	Telecommunications Software and Multimedia

Name:	HAVi Components in Digital Television
Date:	20.11.2001
Num. Pages:	73

Supervisor:	Vuorimaa, Petri and Gómez Vilda, Pedro
Instructor:	Vuorimaa, Petri

Digital television broadcast started in Finland on 27th of August 2001. A new period in this entertainment field has already begun. Because of the importance of television in the society, the shift between analogue and digital has to be done with the viewers in mind.

The User Interface of any application provides the user the means to communicate with the system. Since it is what the viewer can see and interact with, it is a critical issue to provide a television friendly Graphical User Interface for digital television applications.

A Graphical User Interface framework provides a set "ready made" screen widgets. It facilitates the work of the developers, since they do not have to deal with the underlying system. In addition, these widgets could be implemented specifically for the purpose at hand (e.g., for digital television in this case).

The aim of the thesis is the implementation of a GUI framework, based on HAVi specification, for digital television. HAVi was followed because it is included in DVB-MHP, the digital television standard in use in the European countries.

This thesis is divided into two main parts. The first one studies the main aspects of digital television. The second part explains the implementation of the GUI framework, including a case study done in a digital television environment already developed.

The theoretical background focused on the development of digital television applications. First, the set-top box is studied, since it is the device needed at home. Then, the new services digital television provides are explained. Followed by the definition of a home network. After, the different standards and organisations involved in digital television are determined. Finally, how applications are implemented following DVB-MHP is studied.

The practical part explains the implementation of the GUI framework. The case study shows some benefits of this approach over using a general-purpose framework or developing application specific widgets. The use of a digital television oriented framework minimised the storage memory needed. Developing applications becomes more efficient, since the widgets are already done. Moreover, the visual presentation and the behaviour of the widgets are TV-friendly.

Finally, it is important to highlight another conclusion. The number of non-PC devices used for interactive applications is increasing. These devices have different needs. For this reason, a key idea is to determine a unified GUI framework and provide different implementations for each device at hand.

Language:	English
Keywords:	Digital Television, GUI, GUI Framework, Java

INFORMACIÓN SOBRE LA TESIS

Autor:	Cesar Garcia, Pablo
Universidad:	Helsinki University of Technology
Laboratorio:	Telecommunications Software and Multimedia

Nombre:	HAVi Components in Digital Television
Fecha:	20.11.2001
Núm. Páginas:	73

Supervisor:	Vuorimaa, Petri and Gómez Vilda, Pedro
Instructor:	Vuorimaa, Petri

Esta tesis se ha escrito en el *Telecommunications Software and Multimedia Laboratory* de la *Helsinki University of Technology*. El supervisor e intructor ha sido el *Professor Petri Vuorimaa* y el *catératico Pedro Gómez Vilda*. Su realización ha sido posible gracias al programa de intercambio *Erasmus*.

El broadcast de la televisión digital comenzó en Finlandia el día 27 de agosto del año 2001. Esto significa que una nueva época en la historia de la televisión ha comenzado. Debido a la importancia que este medio de comunicación tiene en la sociedad, es esencial que la transición, de analógica a digital, se haga teniendo en cuenta a los usuarios.

En un programa informático, la interfaz de usuario proporciona los mecanismos de comunicación con el sistema. Dotar al software de una interfaz gráfica de usuario amigable adquiere una importancia decisiva en el desarrollo de éste, ya que es aquello con lo que el usuario va a interactuar.

Un framework de interfaz gráfico de usuario proporciona un conjunto de elementos básicos de pantalla “ya elaborados” (e.g., botones y etiquetas). El framework facilita el trabajo de los desarrolladores debido a que proporciona un nivel de abstracción sobre la plataforma en la que trabajan. Además, estos elementos pueden ser implementados específicamente, dependiendo del objetivo final (e.g., la televisión

digital). El objetivo de esta tesis ha sido el desarrollo de un framework de interfaz gráfica de usuario, basándose en el estándar proporcionado por HAVi, para la televisión digital. La selección de las especificaciones HAVi como base se debe a que éstas forman parte de DVB-MHP (i.e., estándar de televisión digital en los paises europeos).

La tesis se divide en dos partes. La primera trata de las características principales de la televisión digital. La segunda explica el desarrollo del framework, incluyendo un case study sobre un entorno de televisión digital previamente desarrollado.

La parte teórica se centra en el desarrollo de aplicaciones para la televisión digital. Comienza explicando en qué consiste una set-top box (i.e. el dispositivo extra necesario para acceder a la nueva televisión). Continúa definiendo los nuevos servicios que ésta, gracias a la digitalización, va a proporcionar. Seguidamente, introduce el concepto de home network, ya que la televisión formará parte de ésta. Después determina las diferentes especificaciones y organizaciones involucradas en el proceso de digitalización. Esta parte finaliza con un estudio sobre cómo desarrollar aplicaciones siguiendo DVB-MHP.

La parte práctica explica el desarrollo del framework. El case study muestra los beneficios de este método sobre otras dos posibilidades: uso de un framework de propósito general o desarrollo de elementos de pantalla específicos para cada aplicación. La memoria se minimiza. El desarrollo de aplicaciones es más eficiente, puesto que los elementos de panatalla ya están implementados. Además, la presentación visual y el comportamiento de estos elementos son amigables, debido a que se han desarrollado específicamente para la televisíon digital.

Por último, es importante resaltar otra conclusión. La variedad de dispositivos que usan apliciones interactivas incrementa dia a dia, ya no es un monopolio del PC. Por esta razón, una idea clave para el futuro es desarrollar un framework unificado, con implementaciones específicas dependiendo de las características de cada dispositivo.

Lenguaje: Inglés
Palabras clave: GUI, GUI Framework, Java, Televisión Digital

ACKNOWLEDGEMENTS

This thesis was written in the Future TV research project, at the Telecommunications Software and Multimedia Laboratory at Helsinki University of Technology, Finland. I would first like to thank the National Technology Development Agency of Finland (TEKES) and all industrial partners for funding the project.

I also want to express my gratitude to Professor Petri Vuorimaa for the guidance and teaching given during the process. In special, when trying to show the trees among the forest. Thanks, also, to Professor Tapio Takala, because I probably would never started my thesis in this laboratory without his courses and help. I also have to thank my colleges, because in a good atmosphere is easiest to work.

Finland is a cold and dark place. Winter People has played a decisive role, because of their support, in the development of this thesis. Thanks to them I have not given up before ending the process.

My parents and brother, because from them I learnt the meaning of unconditional love. Moreover, because they have provided me the needed education to create my own set of unbreakable rules.

Finally, I want to thank Yoko Yamaguchi. Sometimes, in this imperfect world, two particle crash. In that moment a New World starts, which can be created as wished… darsena.

Helsinki, November 15th, 2001
Pablo César García.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF PICTURES

LIST OF TABLES

ABBREVIATIONS

ACC	Advance Audio Coding.
AFC	Application Foundation Classes.
AIT	Application Information Table.
API	Application Programming Interface.
ATSC	American Television Standards Committee.
AWT	Abstract Windowing Toolkit.
COFDM	Coded Orthogonal Frequency Division Multiplexing.
CPU	Central Processing Unit.
CA	Conditional Access.
CENELEC	European Committee for Electrotechnical Standardisation
DAVIC	Digital Audio Video Council.
DASE	Digital TV Application Software Environment.
DigiTag	Digital Terrestrial Television Action Group.
DTG	Digital Television Group.
DTT	Digital Terrestrial Television.
DTV	Digital Television.
DVB	Digital Video Broadcasting.
DVB-C	Digital Video Broadcasting standard for cable delivery systems.
DVB-S	Digital Video Broadcasting standard for satellite delivery systems.
DVB-T	Digital Video Broadcasting standard for terrestrial delivery systems.
EBU	European Broadcasting Union.
EPG	Electronic Programme Guide.
ETSI	European Telecommunication Standardisation Institute.
GUI	Graphical User Interface.
HAVi	Home Audio/Video Interoperability.
HDTV	High Definition Television.
HTML	HyperText Markup Language.
IEC	International Electrotechnical Commission.
I/O	Input/Output.
IRD	Information Resource Dictionary.
ISDB	Integrated Services Digital Broadcasting.

ISO	International Organisation for Standardisation.
JDK	Java Development Kit.
JMF	Java Media Framework.
JVM	Java Virtual Machine.
JRE	Java Runtime Environment.
MHEG	Multimedia Hypermedia Expert Group.
MHP	Multimedia Home Platform.
MPEG	Moving Picture Experts Group.
NorDig	Organisation specifying a common digital television platform for the Scandinavian countries.
NVOD	Near Video On Demand.
OCAP	OpenCable Application Platform.
OS	Operating System
OSD	On-Screen Display.
PAL	Phase Alternating Signal.
PID	Packet Identification Number.
PPV	Pay Per View.
QAM	Quadrature Amplitude Modulation.
QPSK	Quadrature Phase-Shift Keying.
RAM	Random Access Memory.
ROM	Read-Only Memory.
RTOS	Real Time Operating System.
STB	Set-top Box.
TS	Transport Stream.
UI	User Interface.
UML	Unified Modeling Language.
VOD	Video On Demand.
VSB	Vestigial Side Band.
XML	Extensible Markup Language.
YcrCb	Colour palette for conventional television systems.

1. INTRODUCTION

Regular television broadcasting services began in England and France in 1936 [1]. In June 1999, the Finnish Ministry of Transportation granted licenses for three terrestrial multiplexes to start operating digital television before 1st September 2001. At the same time it was preliminarily decided that the complete transaction from analogue to digital television would take place at the end of the year 2006 [2]. In 1936, the pictures in the system were composed of 405 lines and the viewer was only able to sit in front of the television and watch what the stations decided to broadcast. Today, this so-called "dumb" device is changing, the digital quality is coming, interactive applications are arriving, and the number of new services available creates a wide range of possibilities.

As mentioned above, a new period in the history of the television is beginning. A revolution in this entertainment field is coming about. It has to be taken into account that television has been a part of everybody's life for many years. So, habits, practices and traditions have already been developed in the society. Consequently, everyone involved in the process of providing this entertainment service, such as station workers, developers, and manufactures, cannot forget that it is oriented to all the population (i.e., people with different backgrounds). The main topic of this thesis is to highlight the importance of having an adequate television-friendly Graphical User Interface (GUI) when developing new applications for digital television.

This thesis has been written at the Telecommunications and Multimedia Laboratory of the Helsinki University of Technology, in the Future TV research group under the supervision of Professor Petri Vuorimaa. This thesis is divided into five chapters. The first chapter consists of an overview of the digital television. The second chapter talks about the different standards and organisations that are working on digital television. The third chapter is focused on how to proceed, when developing digital television applications. GUI aspect is emphasised. The fourth chapter is about the practical part of the thesis, where a GUI framework is developed and used is real applications. Finally, the last part of the thesis covers the conclusions and future improvements.

2. DIGITAL TELEVISION

Before starting this thesis it would be beneficial to explain what digital television really means. Taking a look at the term, it is easily understandable that it is formed by the juxtaposition of two different words, adding to television a new feature, which is digital. Television, from the station point of view, is composed of two main tasks, production of content and transmission of it to the viewers. Nowadays, in Finland, most of the contents is produced and stored using digital methods. Digitalisation of television, therefore, is referred to the transmission part [3]. Some of the improvements that this digitalisation will bring are the following [4]:

- Elimination of interference such as ghosting.
- A return channel is added, so interactivity is reality.
- Compression of the signal. More information can be squeezed in to the same bandwidth, so more TV channels and more services can be made available.

The compressed video data in digital television is actually a digital TV bit-stream. Consequently, it can be manipulated using software. This opens new opportunities to process the information. Effective encryption techniques facilitate conditional access and effective error correction. One may also add additional data such as subtitles, other languages, and so on [5].

On the other hand, there are some drawbacks connected to the convergence of television and digital technology. One of them is due to the necessity of greatly compress the signal for bandwidth reduction. Hence, appears the exigency to deal with misrepresentations of zeros and ones in the presence of transmission errors. Such errors, finally, could result in the total disruption of communication [6].

At the present, there are three different digital television standards in the world. In USA, the American Television Standards Committee's (ASTC's) Grand Alliance will operate under the aegis U.S. Federal Communications Commission (FCC) [7]. The Digital Video Broadcasting (DVB) is a common single television standard agreed by all the European countries. Finally, Japan has developed a unique digital system, Integrated Services Digital Broadcasting (ISDB) [8].

The European standard, DVB, defines three different broadcasting standards: DVB-C for cable transmissions, DVB-S for satellite, and DVB-T for terrestrial. The prime difference among them is the way the compressed signal is packaged for transmission [7].

All the standards related to the broadcast of the signal are based on MPEG-2 coding. MPEG-2 is an international digital compression standards set by the Moving Pictures Experts Group (MPEG) of the International Electrotechnical Comission (IEC) and the International Organization for Standarization (ISO). The main feature of this

technique is the capacity of transport audio, data, and video in the same stream. Not only the sound and the video particular of the television environment can be broadcast, but also data information such as subtitles, program schedule, etc.

This improvement involves that new requirements have to be accomplished by the viewers to have access to digital television. In the analogue era, a television and antenna were enough, a cable or a parabola antenna in other than terrestrial cases, to watch television. From now on, a new device called Set-top box (STB) is required to convert this TV bit-stream into analogue. In the beginning, the STB is going to be a separate device. This way, the investment needed to switch from analogue to digital by the users is not going to be prohibitive. In the incoming years, it is expected that the STB will be a part of the television set, so the users will have an integrated receiver.

This chapter is divided into three different sections. Section 2.1 studies the set-top box, from the hardware and the software point of view. In the next section, 2.2, the different services that digital television is going to provide to the users are explained. The last part, section 2.3, considers the home network, as an environment where different digital electronic devices will cohabit and communicate.

2.1 Set-Top Box

The digital STB has it roots in the “cable set-top box”, used to watch cable television. This first kind of STB was introduced to compensate TV-tuner deficiencies that occur because the broadcast and cable signal environments were different [9]. The cable STB was used to select between a wide range of TV channels, using a hand-held remote control. It was also equipped with special hardware, needed to prevent the access to the cable data to unauthorised viewers (i.e., unscramble the data). [10].

As digital-broadcasting technology matures, a new set-top box (i.e., the digital STB) arises. It serves the triple functionality of being a communications receiver, a computing device, and a consumer device for media entertainment [10]. This section is structured as follows. First, in sub-section 2.1.1, the functionality of the STB is explained. Then, in sub-section 2.1.2, the generic architecture of the STB is introduced. After, in sub-section 2.1.3, how the graphics are rendered is defined. Finally, in sub-section 2.1.4, the software stack of the STB is determined.

2.1.1 STB functionality

The STB is designed to receive and decode the broadcast television services. It tunes to the required channel, extracts and decodes the selected data, checks the access rights of the user, and outputs picture, sound, and other services as needed [11].

In the Scandinavian countries, a common STB specification is used, which is called NorDig II. It states the minimum hardware capabilities and the "Terminal Specifications" of the STB (i.e., the I/O function the STB has to fulfil) [10]. Consequently, based on this "Terminal Specifications" a generic architecture of the STB can be defined.

2.1.2 Architecture of STB

An overview of the block diagram of a generic STB is shown in the figure 1. Five major parts compose it. First of all is the Front End, which is in charge of converting the RF/IF input to the MPEG Transport Stream meeting the DVB specifications. Then, the Conditional Access sub-system, which controls the user access to the broadcast services using scrambling and encryption. The third part is the MPEG-2 decoder used to decompress the MPEG Transport Stream. The fourth part is the Output Signal Encoder performing the final signal ready to be sent to the television set. The last but not least is the Microcontroller necessary to control the correct working of all the other parts of the system [10, 11, 12, and 13].

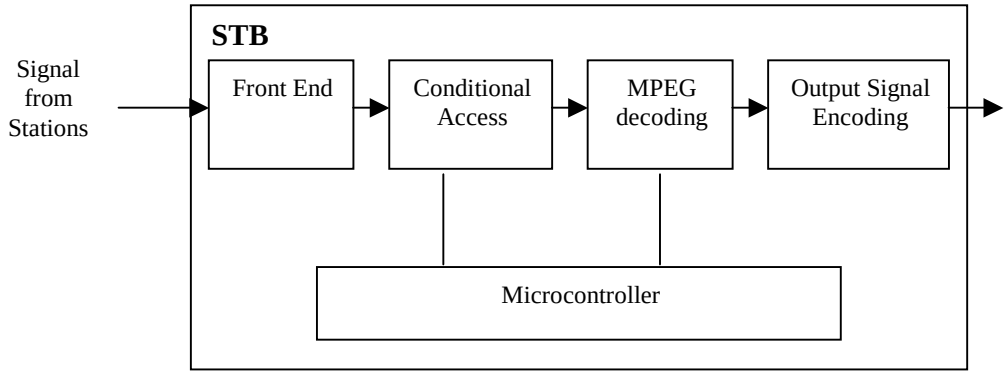


Figure 1. Block diagram of a set-top box [11, 12, and 13].

It is important to remark that the hardware architecture introduced here is the generic needed to accomplish the tasks the STB is expected to do. Accept the input, as a gathering of video, audio, and data, convert it, and send it to the television set or VCR ready to be watched. This architecture is based on the I/O functionality needed. The different manufacturers have to differentiate their own products, and thus, the final configuration is let to them. They can determine how to combine the blocks, the microcontroller in use, the storage capacity, how to improve the speed, and so on.

Front End

The first part in this block is the Tuner, which is in charge of tuning to the carrier frequency and converting the incoming signal to base-band. The base-band signal is then sampled to create a digital representation of the signal at the A/D. In cable

transmission, for example, normally the tuner accepts input from 120Mhz up to 862 MHz. Later, the digital transport bit-stream is recovered after the Demodulator and the Error Corrector work. The diagram of this part can be seen in the figure 2.

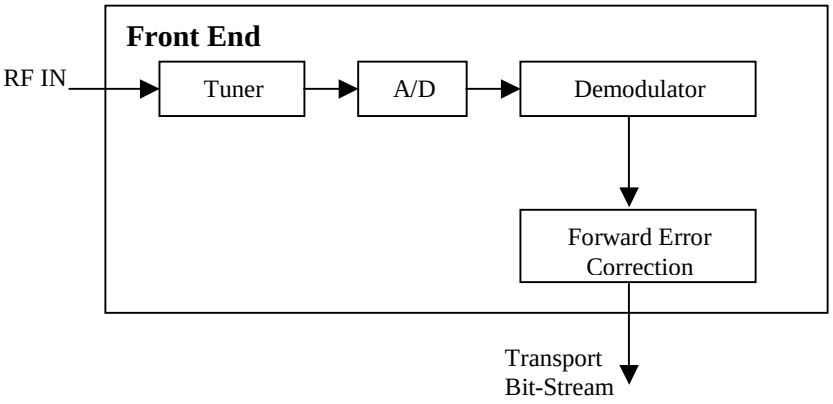


Figure 2. Diagram of the Front End [11].

The difference between the three broadcast standards, as said before, is in the way the compressed signal is packaged for transmission. So, the Front End in each case is distinct in the way that it has different channel tuners and decoders for each of them. This gives to the manufacturers the possibility of insert, in a single STB, the correct tuners and decoders for the three standards as the only difference to fulfil the specifications.

Conditional Access

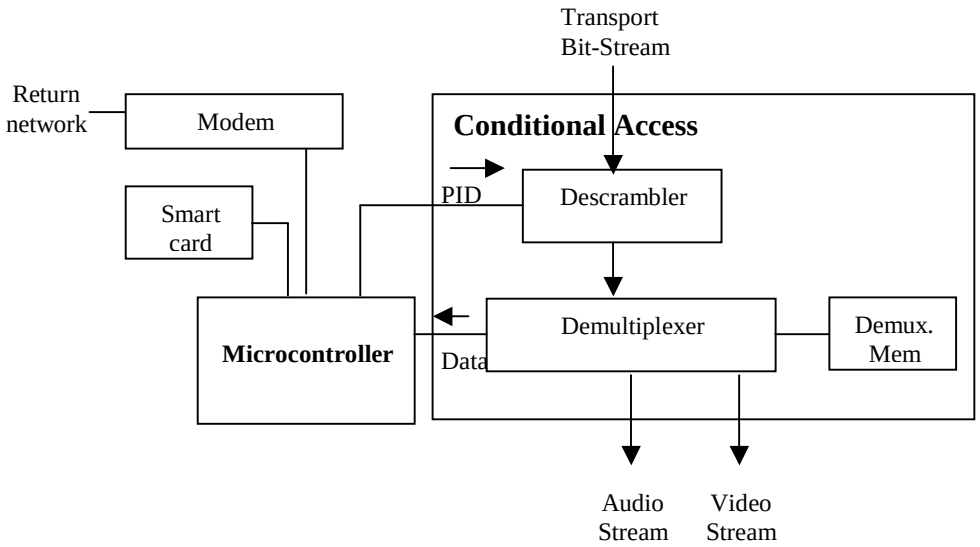


Figure 3. Diagram of Conditional Access [11].

As shown in the figure 3, the Conditional Access of the STB is divided in two different elements. The descrambler is used to select packets from the transport stream based on the Packet Identification Number (PID). This component identifies a packet as belonging to a particular program or carrying service information. In addition, it descrambles them synchronised by control words or keys under the control of a security model (e.g., Smart Card). Then, the packets have to be demultiplexed into audio stream, video stream, and data information. Finally, this information goes to the audio decoder, the video decoder, and the microcontroller, respectively.

MPEG Decoder

After the Demultiplexer has performed its action, there are three different kinds of data. The audio stream, which is decompressed by the audio MPEG decoder obtaining stereo audio to be sent to the television set. The data information is sent to the microcontroller obtaining the graphics and text that have to be overlapped with the video. At the same time, the MPEG video decoder decompresses the video stream.

The On-Screen Display (OSD) element is in charge of composing the video signal (digital Y, Cr, Cb format) overlapping with the graphics and text needed (e.g., subtitles). Also, in this block, an aspect ratio conversion is done. The MPEG decoder is depicted in figure 4.

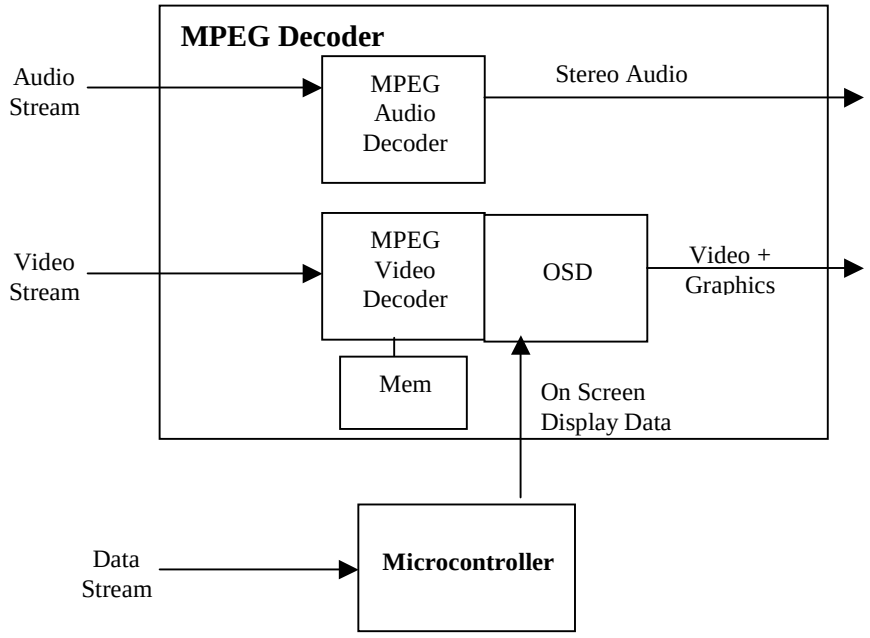


Figure 4. Diagram of MPEG Decoder [11].

Output Signal Encoder
The Output Signal Encoder converts the signal so that the television set can show it. This part of the STB is depicted in figure 5.

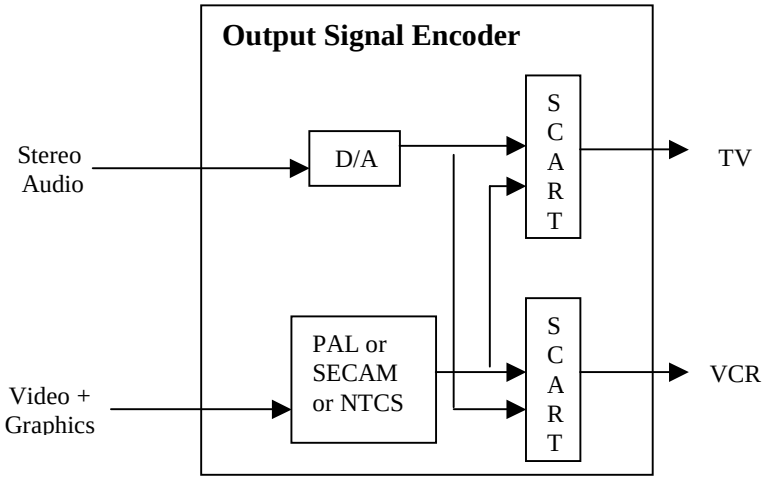


Figure 5. Diagram of Output Signal Encoder [11].

The Microcontroller
The microcontroller implements the software needed, it sets up the devices in the IRD, and controls the I/O ports. A list of the minimum capabilities, taken from NorDig II specifications, the microcontroller has to meet is given below [13]:

- 16 Mbytes of RAM.
- 8 Mbytes of Flash Memory.
- 4 Mbytes of video RAM.
- A real time clock/calendar running continuously.
- An internal timer for the possibility of automatically switches from stand by mode to the operational mode. The timer shall be initiated locally (accepted by end user).
- Supports Multimedia Home Platform (MHP) Application Programming Interface (API).
- Persistent mass memory capability.

2.1.3 Graphic Representation

The main theme of the thesis is the implementation of a television oriented Graphical User Interface (GUI). Making the users, when watching digital television, feel like watching the usual television. Hence, it is necessary to explain how the graphics and text are going to be rendered. The mechanism used to overlap the video, the graphics,

and the text is based on having the OSD information on a plane over a video. Then, some pixels of the plane are made transparent. This way, the video appears through the OSD [11].

The requirements that has to be fulfilled by the OSD following the Nordig II specifications are [13]:

- Resolution of 720 by 576 pixels and lower.
- At least 65536 colours per pixel including transparency. The actual presentation shall be specified as defined in the DVB-MHP specification.
- 4 logical display planes:
  - Video plane for full screen MPEG video.
  - Graphic plane I for MPEG I-still frames, JPEGs, GIFs, PNGs, and/or decimated live MPEG video.
  - Graphic plane II for MPEG for graphics (full screen).
  - Simultaneous overlapping displays of all pans.
- Blending of the graphics with video or stills backgrounds.
- 4:3 and 16:9 aspect ratios in dependence to the installation setting at the SCART I interface.

2.1.4 Software in the STB

It is very important to point out that the software resident in the STB has to be easily updated. In other case, the STB will become completely obsolete in a short time. To make it possible, the software should be stored in a conventional RAM or Flash Memory [9].

The stack of the software needed comprises an Operating System (OS), a Java Runtime Environment (JRE), a set of specific digital television oriented APIs (Application Programming Interface), and the applications running in the STB. The figure 6 shows the layers, which the STB needs to run digital television applications.

A brief explanation of the layers should be done at this time:

- OS: performs all the low-level processes and keep them as abstracts entities from the application's point of view.
- JRE: needed because Java was selected as language to be used to develop applications for digital television [14]. Necessary to run such applications.
- API: is a built-in programmer's toolkit for requesting data objects or services resident on a particular operating system [15].
- Applications: depending on the location, two types of applications can be distinguished. The first one is formed by the resident applications (i.e., stored locally) as the Navigator. The second one comprises all kind of downloaded applications (i.e., stored in a remote server) as a betting program, for example.

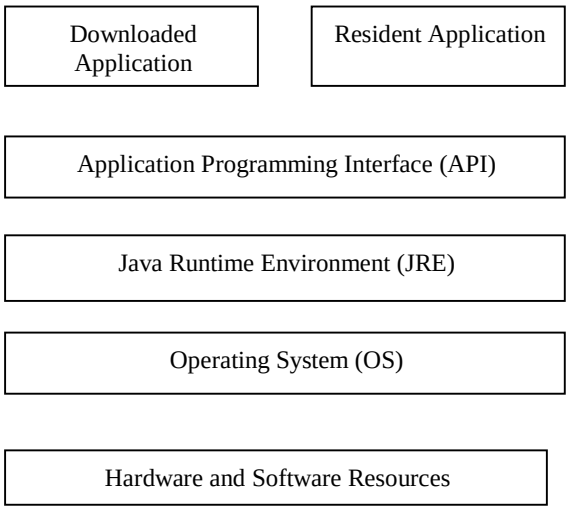


Figure 6. Software stack of the STB [16].

2.2 New Services of the DTV

A large variety of new ways of entertainment are coming about when the convergence between television and digital technology arrives to the users' home, such as [17]:

- Order a CD, while watching a concert on the television.
- Access to a several hundred of different channels.
- Use Internet on the television.
- Play an interactive game on television.
- Have access to Near Video on Demand (NVOD).
- Super Text TV service.

At this point, it is important to highlight why the stations have the possibility to increase the number and the quality of the services offered. The key factor is the compression of the broadcast signal, which can be done because of the digitalisation of it. This compression provokes a dramatic reduction of the bandwidth usage. Consequently, it gives to the station bandwidth enough to broadcast more information than in the analogue era. Also, it has to be remarked that if the return channel is added interactivity is reached.

These new services can be divided into three different categories: information navigators, Super Text TV and interactive-programs [18]. The first type, information navigators programs, are necessary because the viewer might find his way among the several hundred of digital channels that will be available [9]. The Super Text TV is used to provide all kind of information needed (e.g., the transportation schedule, the

news) to the user. Finally, the interactive programs, are applications, which require a return path to some central server (e.g., pizza delivery, shopping). The figure 7 depicts the new services the digital television may offer.

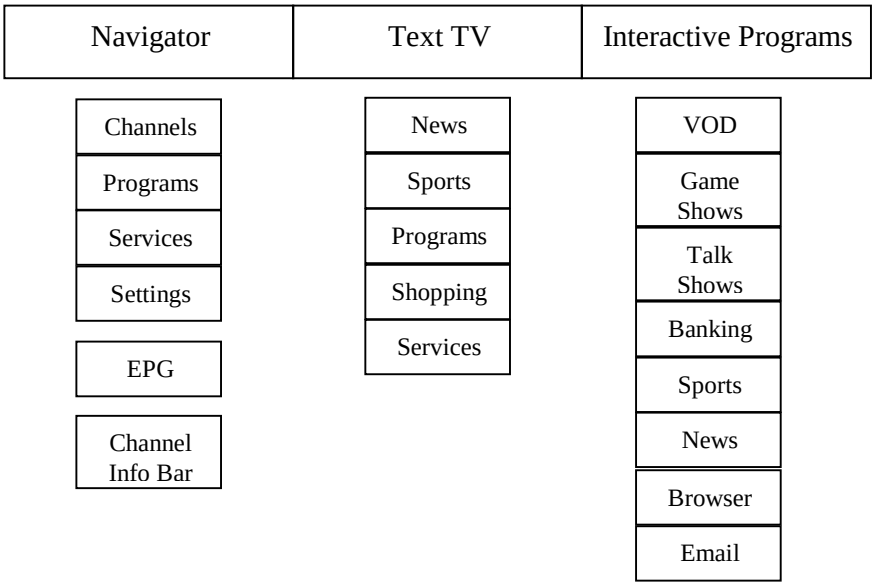


Figure 7. Digital television service architecture [18].

In this section, the following services are going to be studied. First of all, Navigator, Channel Info Bar, and EPG, which can be included as information navigators, are studied in sub-sections 2.2.1, 2.2.2 and 2.2.3, respectively. Then, in sub-section 2.2.4, Text TV (i.e., the digital version of the analogue Teletext) is defined. After, in sub-section 2.2.5, the browser, as other service digital television may provide, is analysed. Finally, in sub-section 2.2.6, different kinds of interactive programs that can be implemented for the digital television are examined.

2.2.1 Navigator

The Navigator is the most important of the new services. It provides the user access to the system information, and allows the user to control the operation of the STB [13]. It can, also, be used to select a program among the immense number of channels available. To select a program one can use either a program guide or a channel guide [19]. Last but not least, the Navigator allows the user to create personalised program lists or favourite channels for an individual viewer [5 and 19], making easier to navigate the following times. A screen-shot of a Navigator, developed by the Future TV research group in the Telecommunications and Multimedia Laboratory of the Helsinki University of Technology, is shown in the picture 1.

Picture 1. Screen-shot of a Navigator application [19].

The Navigator is by definition part of the system (i.e., is loaded in the ROM memory of the STB or in the Flash memory) [13]. The design of the Navigator is a task that has to be done by the manufacturer [20].

The stations send the data, which is displayed via the Navigator by the STB. This task, as said above, can be done as a channel guide or as a program guide. Since several stations and several manufacturers of receivers will do their own implementations, the data has to have a specific structure. A standard called the DVB-SI broadcast specification states it [21].

It has to be remarked, at this point, one important issue when developing a Navigator for the digital television. A friendly user-interface and an intelligent way of showing the information are needed. It is very easy to get lost in a vast ocean of different possibilities. Also, it could be interesting to study, in a near future, the idea of having a search engine to afford the huge amount of information given to the user.

2.2.2 Channel Info Bar

The Channel Info Bar displays information about the current program. It is supposed to cover only a part of the television disturbing the less the user watching the program itself [18 and 19].

2.2.3 EPG

The Electronic Program Guide (EPG) is an application similar to the Navigator. It is meant to show the user the schedule of the different channels during the following week. The viewers can navigate through the programs offered. They can select a program, which takes place in a few days (within one week). When selecting, the user has two possibilities: either to be reminded by the receiver or request the STB to record it. Another difference with the Navigator is the impossibility of changing the settings of the receiver [18].

The presentation of the information of the TV screen, specified by ETS 300 707, is via the standard template shown in figure 8 [23]. Both manufacturers and broadcasters can determine presentation styles and create their own identities to differentiate their products.

The template has four different parts, header, event, message, and navigation areas, which are explained below [23]:

- Header Area: the broadcaster decides how to design this area. It is used to show the title of the guide.
- Event Area: the manufacturer defines the presentation style. It contains the list of programmes defined by the broadcaster.
- Message Area: is linked to the highlight programme or menu entry in the Event Area. It is used to show brief descriptions of the selected program. The broadcaster defines it.
- Navigation Area: the manufacturer defines this area. It is used for navigation purposes.

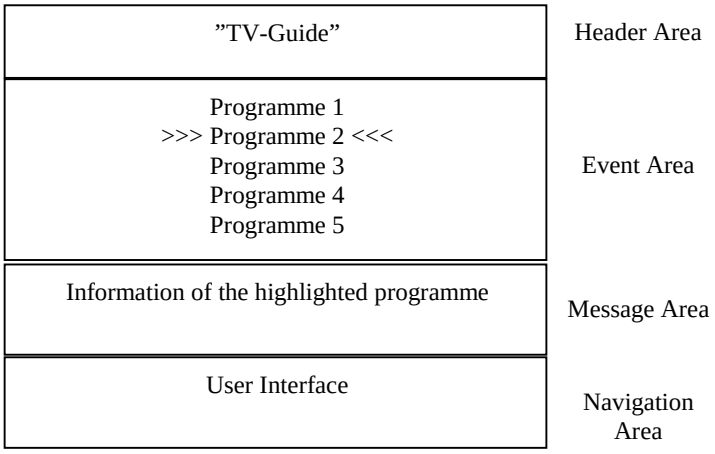


Figure 8. Standard template of the EPG [23].

A screen-shot of an EPG, developed by the Future TV research group in the Telecommunications and Multimedia Laboratory of the Helsinki University of Technology, is shown in the picture 2.

Picture 2. Screen-shot of an EPG application [19].

Consumer electronics manufacturers are responsible for implementing EPG decoders. A standard is needed to structure the information (as happened with the Navigator), because the stations have to broadcast the signal, but the receivers have to be able to display correctly the information. Manufactures and broadcasters worked together to draft a specification. The result is the ETS 300 707 [22].

2.2.4 Text TV

Super Text TV is going to play the role of the analogue Teletext service, which is popular in Europe [18]. The main difference between them is the way in which the data is transmitted. The method used in analogue television is by including the data in the vertical-blanking interval. This interval is the period of time needed for the picture tube's electron beam to retrace its path vertically up the screen [9]. In digital television, there are not vertical-blank intervals. Therefore, this text information is stored in data carousels and transmitted cyclically as transport streams within the MPEG-2 stream sent from the station [24]. As digital television will have more bandwidth, Super Text TV can contain more advanced graphics, still-image, sounds, and animation [25].

The content of the Super Text TV comprises all kind of information (e.g., news, sports, weather, and transports timetable), that the station expects to be useful for the users. It is important to remark that the information is delivered (broadcast), so there is no necessity of Internet connection to have access to it [18].

The information has to be structured into sections, in a similar way as a newspaper, in order to be accessible (e.g., sports, news, weather). The content information is divided into pages including text, pictures, and graphics. This produces a real advance from the Teletext service, which only allowed text.

Also, as in the case of the Navigator and EPG, a standard is approved, which is the ETSI ETS 300 706 [26]. The specification allows the use of hyperlinks in the Super Text TV, which are activated by the colour keys (i.e., red, green, yellow, and blue) of the remote controller. It makes the access to the information easier and faster than the usual teletext. A screen-shot of the Super Text, developed by the Future TV research group in the Telecommunications and Multimedia Laboratory at the Helsinki University of Technology, is shown in the picture 3.

Picture 3. Screen-shot of a Super Text TV application [24].

It is necessary the use of an interchange format to code the content of the different pages. The current digital television Super Text TV uses MHEG (format developed to be used in multimedia applications), but other formats can be used [18]. One very interesting option, which has to be pointed out, is the use of Extensible Markup Language (XML) to develop Text TV content [18, 24, and 25]. XML is a structural document description language, which is independent of its presentation. Hence,

XML provides the same content that can be displayed in devices with different features [27].

2.2.5 Browser

Nowadays, Internet is a tool widely in use. The digital television community, taking this into account, is interested in implementing a www-browser in the receiver. The browser is an application used to locate and display Web pages.

Normally, when one tries to use an application in a different environment than the presumed problems arise. This particular case, using a browser (computer oriented) in the television is not an exception. The problems can be resumed as the following [18, 27 and 28]:

- One-foot versus fifteen feet dilemma: the viewing distance in the case of a computer is longer than a PC.
- User interaction: TV uses a remote control, so navigation is normally restricted to the up-down-left-right arrows and the numerical keys. On the other hand, mouse and keyboard are common PC input devices.
- Resolution: TV screen has a much lower resolution than a computer monitor does. High-resolution drawings have to be avoided in order to elude the flickering.
- Number of users: usually, more than one person watches one television at a time, while PC is used by one person [29].
- Colour definition: high saturation should be avoided because the colour depth and fidelity are not as good in a TV screen than in a computer monitor. Highly saturated colours tend to bleed.

Normally, the Web sites are implemented using HTML language. Hence, the browser is able to translate this code into text and graphics. Technology moves on very fast. Therefore, it should be suggested the possibility of implementing an XML browser to the receiver in order to develop a STB adapted to the new technologies that are coming about [30].

2.2.6 Interactive programs

When talking about interactive applications two different interpretations can be used. The broadest consists of those applications that run in the receiver, which allow the user to control their behaviour (e.g., Navigator, EPG, and pizza delivery). The second one, which is applied in this Master Thesis, comprises those programs stored in the STB, which require a return path to some central server (e.g., pizza delivery, home banking, betting) [28].

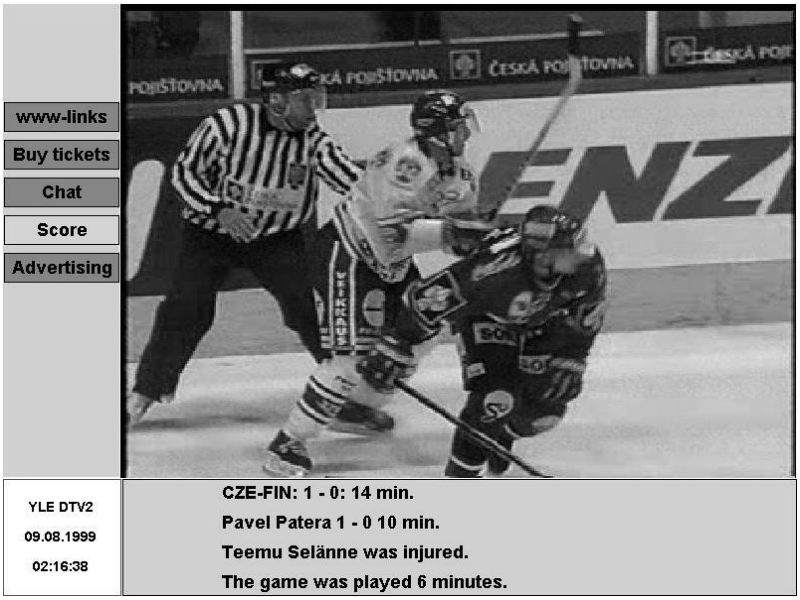
There is a wide range of interactive applications aimed to the digital television. Some of them are going to be studied below [5, 6, 9 and 31]:

- Video On Demand (VOD): allows users to access a movie from a remote video server at any time. It is like a video juke box, with a wide choice of movies from which the subscriber can select. The service provides full-VCR functionality (e.g., play, forward, and rewind).
- Near Video On Demand (NVOD): is a less expensive approach to the Video On Demand. In this case, the same movie starts on different channels in 15 minutes time increments. It only has sense in satellite or cable broadcast because of the wide bandwidth needed. Although, NVOD is not a real interactive application, in the beginning of the digital television (it doesn't seem possible to have VOD) can work.
- Pay Per View (PPV): allows viewers to purchase the right to view, at their own convenience, premium events (e.g., football match, Olympic games).
- Educational applications: comprises all kind of distance learning programs.
- Games: games life-like 3+D graphics, high speed of the movements of the characters, realistic background and the incorporation of life video. Interactivity allows players play the same game from different sites.

Other interactive applications consist of taking part on a game show. Betting to the user's favourite hockey team. Doing bank transactions and taking care of the bank account. Interact in a talking show. These applications will be enjoyed while seating in the sofa in front of the television set.

It should be said, as a conclusion about the interactive programs in the digital television, that there are a lot of new ideas and new approaches in the mind of the developers. But the problem arises, when trying to guess, which are going to be successful. That is because of the wide range of different people who are going to watch digital television, the different backgrounds and ages of them. Consequently, today there is not, yet, a clear picture of the kind of programs the viewers would prefer [27].

An example of an interactive application, developed by the Future TV research group in the Telecommunications and Multimedia Laboratory at the Helsinki University of Technology, is shown in the picture 4.



Picture 4. Screen-shot of an interactive application.

2.3 Home Network

The current trend in the entertainment field has been to convert all kind of electronic devices to digital (e.g., digital television, DVD, CD player). The next logical step, which is underway, is to network them in the user's home. There are many advantages with this approach; one of them is that the user can control all of the devices in a unified way. For example, the viewer of the television can, with the remote control, mute the television while the commercials and play, in the Hi-fi, a CD of his favourite band.

There are two important factors, which have to be taken into account, when trying to connect the different digital devices at home. The first one is that special feature requirements have to be fulfilled by the home network. The second one is concerning to the great variety of digital devices at home, which causes a situation of not-interoperability between them.

Regarding to the special requirements to satisfy. This requisites include timely transfer of high-data-rate audio/video (AV) streams, auto-configuration, self-management, and low cost cabling and interface [32]. The IEEE 1394 is a serial bus standard emerged in 1995. The main features, which turn it into the ideal bus to develop the home network, are [10]:

- High bandwidth capacity, up to 400 Mbits/sec.
- Capability of transfer in an "isochronous" mode.
- Hot pluggability.
- Communication mode done as "peer to peer".

The second problem, actual non-interoperability between devices, is due the existence of different vendors' ones with their individual specifications. Hence, multivendor interoperability between consumer electronic devices and computer devices is needed [33]. The Home Audio/Video interoperability, HAVi, architecture specifies a set of APIs, services and on-the-wire protocol assuring that products from different vendors can collaborate [33].

In this part of the thesis the two key factors, introduced above, are studied. First, in sub-section 2.3.1, the IEEE 1394 bus, which has the special features, needed to make the home network a reality. Then, in sub-section 2.3.2, the HAVi architecture is explained, which guarantees interoperability between the devices at home. Finally, in sub-section 2.3.3, a short explanation about why HAVi technology has been studied and not the rest home network technology is done.

2.3.1 IEEE 1394

One of the most important characteristics of the IEEE 1394 serial bus is the ability of supporting both asynchronous and synchronous communications. The asynchronous guarantees error-free delivery of data. This communication mode can not be used for multimedia purposes because the error-checking protocol needed introduces time delay. Hence, it is normally used for control commands. On the other hand, the synchronous communication ensures real time transfer where some errors can be tolerated [32]. In figure 9, the protocol stack defined in IEEE1394 is shown.

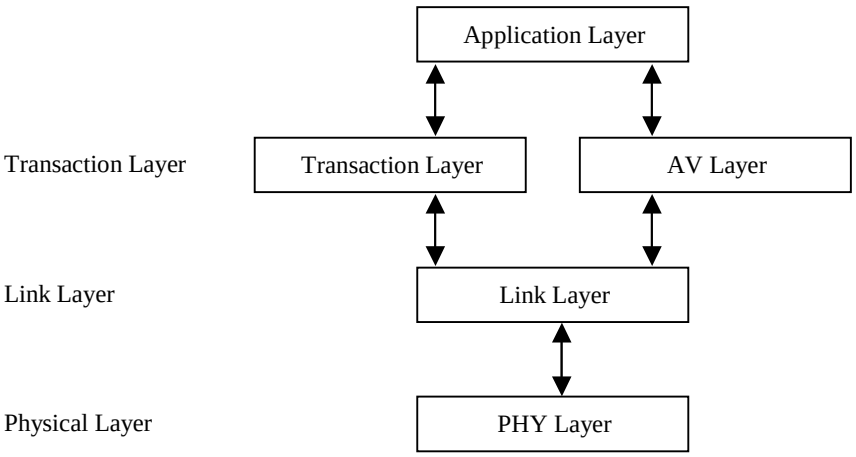


Figure 9. IEEE 1394 Protocol Stack [10].

Three different layers, below the application one, are defined in IEEE 1394 standard [10 and 33]:

- Transaction Layer is divided into:
  - Transaction Layer, in charge of the asynchronous packets.
  - AV Layer, which takes care of the synchronous packets.
- Link Layer: handles packet transmission and reception. It also controls the cycle for synchronous channels.

Physical Layer: can deal with asynchronous and synchronous packets. Converting the earliest into logical symbols and the latter to MPEG-2 transport stream when sending information.

2.3.2 HAVi Specification

The HAVi architecture is based on IEEE 1394 standard. It is quite complete and concrete with regard to consumer electronic devices. This will help different manufactures to build devices, which can inter-operate with each other [34].

The most important concept in HAVi architecture is the distinction between controller and controlled devices. HAVi controllers, by running a software proxy called Device Control Module (DCM), act as host for the controlled device. As well, a controller may itself behave as a controlled device being represented by a DCM [33].

DCMs can be classified taking into account three different features. The first one is how DCM is obtained: it can be embedded (resident software) or downloaded (get from an external source and dynamically added on the controller). The second one is whether DCM is implemented using native code or Java bytecode. The last characteristic is about the functionality provided by the DCM (i.e., if it is offered the standard HAVi API or a proprietary vendor specific API is also supplied) [34].

In the HAVi specifications four kinds of electronic devices are differentiate, where the first two may act as controllers. These are the following [33 and 34]:

- Full Audio/Video (FAV): provides a Java Runtime Environment, JRE. Hence, a FAV can update bytecode from other devices. It contains, moreover, a complete set of the HAVi architecture software elements. A STB can be included in this category.
- Intermediate Audio/Video (IAV): does not have a Java Runtime Environment. However, an IAV may provide native support control of particular home network devices. A DVD can fall into this category.
- Base Audio/Video (BAV): Do not host any of the HAVi architecture's software elements. Anyway, this devices can be controlled uploading the DCM bytecode (by a FAV) or via the native code (by a BAV). One example is a camcorder.

- Legacy Audio/Video (LAV): these are not HAVi compliant devices, having their own protocol. To control them is required that FAV or IAV act as a gateway.

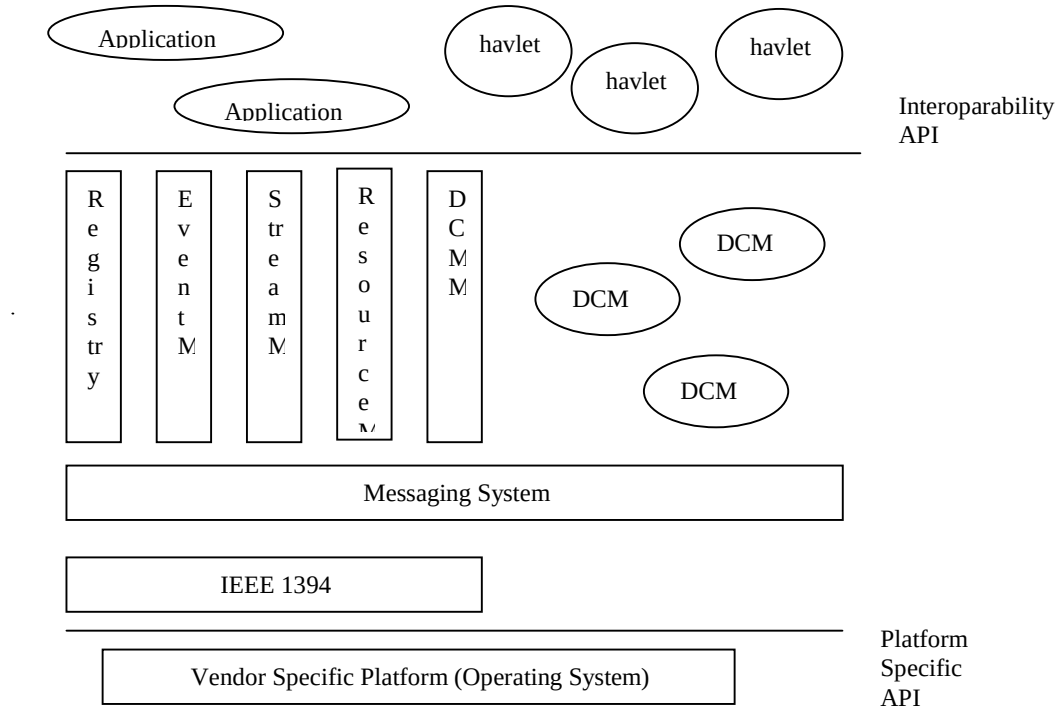


Figure 10. HAVi software architecture [34].

HAVi also differentiates between devices and Functional Component Modules (FCM). It defines a fixed set of APIs for FCM. Consequently, a typical consumer electronic device encompasses more than one functional component. The different FCM are [34]:

- Tuner: allow setting and getting service list (e.g., broadcast channels) or service list components (e.g., select English language).
- VCR: functions like play, record, and rewind.
- Clock: querying clocks and setting their values.
- Camera: control camera devices.
- AV Disc: playing, skipping tracks or manipulation of item lists.
- Amplifier: manipulate volume, mute, loudness, etc.
- Display: filtering functions for displaying video (e.g., contrast, colour).
- AV Display: combine Amplifier and Display FCM, as in a normal TV.

- Modem: establish connection over an outside network and transfer any amount of data (e.g., fax).
- Web proxy: offer access to Internet.

HAVi architecture includes the following software elements [33 and 34]:

- IEEE 1394: supports both synchronous and asynchronous mode.
- Messaging system: letting software elements invoke one another's services through APIs.
- Registry: provides a lookup service for applications, devices and their resources.
- Event Manager: lets software elements post and subscribe to asynchronous events.
- Stream Manager: lets software elements configure and control synchronous data streams.
- Resource Manager: lets schedule of actions and share of resources.
- DCM Manager: allows installation or removal of DCM units.
- DCM: consists of code for the module itself and code for each functional component within that device.

The stack of the HAVi software architecture is depicted in the figure 10. Services in HAVi are modelled as an object. Each object is a self-contained entity, called software element, accessible through a well-defined interface and executed within a software execution environment hosted by the device on which the object runs [33].

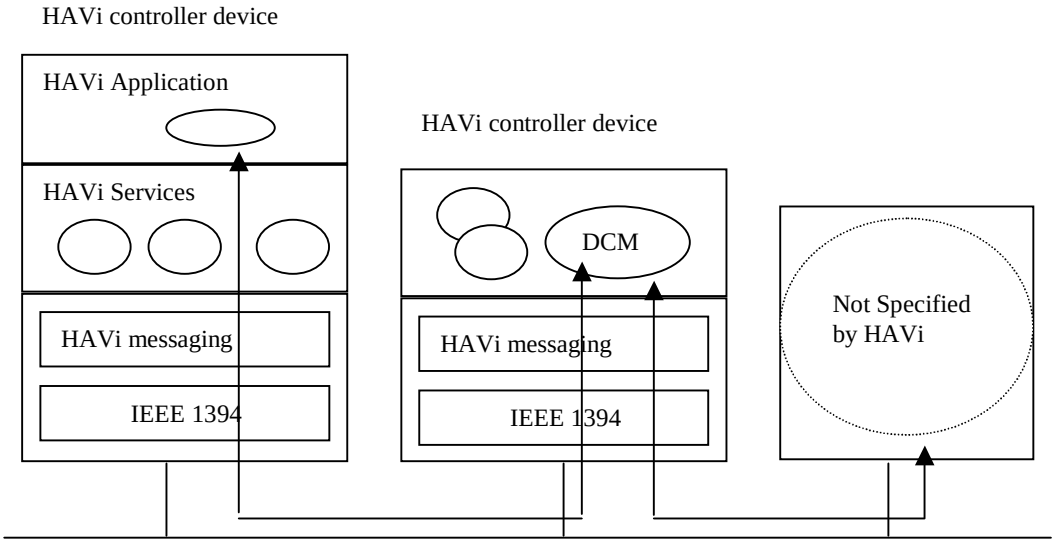


Figure 11. Communication in a HAVi network [33].

As a conclusion it can be said that HAVi provides a standard where the communication is done “peer to peer”. Any device can detect, query, and control any other device. Also, groups of devices can collaborate to provide enhanced services [33]. Therefore, there is not necessary to have a central controller. An example about the communication network in HAVi is shown in figure 11.

Finally, it is important to remark that the intention of HAVi specification is that no APIs will be updated or removed. All changes will be achieved by adding new APIs [35].

2.3.3 HAVi and home other network technologies

HAVi is not the only home network technology, which can be used. There are other technologies like HAPI, Jini, OSGi, Vesa Home Network, and UPnP. Since this thesis is focused in the development of a television oriented GUI, a study of all of them was out of the scope. Anyway, some differences of HAVi and the other technologies should be pointed out [33]:

- HAVi and Vesa Home Network explicitly focuses on home entertainment (e.g., digital television) and AV devices.
- HAVi is platform and language neutral (e.g., HAPI is for the PC environment).
- UPnP and Vesa Home Networking use Web technologies for device description and control. Taking into account that standards change rapidly, this feature can introduce potential pitfalls in a short time. HAVi does not.

3. STANDARDS AND ORGANISATIONS

The first intention of this chapter is to make clear who is who in digital television. Nowadays, in every technological area, many acronyms are produced, which usually originates a state of confusion and perplexity in the inexperienced. This chapter tries to provide a clear picture about digital television acronyms.

Standards are documented agreements containing technical specifications or other precise criteria to be used consistently as rules, guidelines, or definitions of characteristic [36]. Consequently, standards have two different options. First, being approved by a recognised organisation (e.g., ETSI, ISO). Second, being accepted as a de facto by the industry.

In the late 1940's, television standards engaged the emotional issue of national sovereignty. The negotiators were under political pressure to resist all sorts of foreign incursions into weakened economy of Europe and Soviet Union [1]. Nowadays, the picture is completely different. With the European Union idea maturing, a common European standard has been developed (i.e., DVB-MHP). Definitely, this is a good thing if one takes a look at the actual incompatible analogue systems (i.e., Secam and PAL) situation [7]. Also, there are other two standards in the world. Japanese, to be used in their territory (i.e., ISDB) and ATSC Grand Alliance to be used in USA.

In addition, it is important to remark the importance of other organisations. They have not developed standards for digital television, but anyway, their specifications are necessary for digital television regulations. These are MPEG, HAVi, DAVIC, and NorDig. MPEG is in charge of developing specifications for coded representation of digital audio and video [37]. HAVi has stated specifications about the home network [35]. DAVIC has created the standard for end-to-end interoperability of broadcast and interactive digital audio-visual information, and of multimedia communication [38]. Finally, NorDig is a co-operative organisation, which has specified a common platform for digital television within the Nordic region (i.e., Denmark, Finland, Norway, and Sweden) [39].

Finally, there are some other organisations related to the digital television. On one hand there are the standards organisations (e.g., ETSI, CENELEC, and EBU). On the other hand, two important groups called DigiTAG and DTG work to make easier the shift from analogue to digital terrestrial television. DigiTAG aim is to organise and facilitate the implementation and market driven introduction of digital terrestrial television services in all transmission media [40]. DTG, based in UK, was created to ensure that the new broadcast satisfies the standards and that new receivers work the way they should in UK [41].

Taking into account what is said in the previous paragraphs, this second chapter is divided into three parts. The first one, sub-section 3.1, discusses the standards

specified for digital television in the world. The second part, sub-section 3.2, is about other important specifications related to the digital television. Finally, sub-section 3.3 talks about organisations, which, in some way, are related with the outgoing of digital television.

3.1 Standards of Digital Television in the World

As said before, there are three different digital television standards in the world. In this thesis, most attention is paid to the European standard (i.e., DVB-MHP) over the rest, since it has been written in Europe. This section is structured as follows. First, in sub-section 3.1.1, DVB-MHP is studied. Then, in sub-section 3.1.2, ATSC Grand Alliance is introduced. After, in sub-section 3.1.3, ISDB is defined. Finally, in sub-section 3.1.4, the current situation across the world is analysed.

When defining the different standards in this thesis, there is a structure followed. First a little introduction about the organisation involved is given. Then the standard is defined. The categorisation is done taking into account the bandwidth of the channels to broadcast, the video resolution, the audio regulation, and the compression system to be used. It is very important to remark that the last issue (i.e. the compression system) is in all the standards MPEG-2.

3.1.1 DVB-MHP

DVB, formed in 1993, is a consortium of around 300 companies in the fields of broadcasting, manufacturing, network operation and regulatory matters that have come together to establish a common European standard for the move from analogue to digital broadcasting [42].

From the beginning, DVB decided to be a politically independent group. This way, no political decision could be taken to develop the standard. To assure it, the group refused a grant offered by the European Community funding [7 and 43].

The principles guiding the DVB group behaviour are the following [42 and 44]:

1. System should be market-led rather than technology-driven. It is not worthy to develop standards the market is not ready to accept, but use the last technology hints.
2. Interoperability. Ensuring the user all the systems in the market behaves the same way. Multivendor interoperability is assured.
3. Open. The standard is agreed, approved, and published by a recognised standardisation body. Then, they are available at a nominal cost, worldwide.

The group has developed DVB-S, DVB-T, and DVB-C, the standards for satellite, terrestrial and cable broadcast, respectively. The prime difference between them is the delivery transmission system. DVB-S uses Quaternary Phase Shift Keying Modulation (QPSK). DVB-C uses Quadrature Amplitude Modulation (QAM). Finally, DVB-T, the most complex delivery system, uses Coded Orthogonal Frequency Division Multiplexing (COFDM). The major difference between for the terrestrial system is the use of a multicarrier modulation system [44].

In Europe, the TV channel is decided to be 8MHz. As COFDM states, the channel is divided into several thousand of narrow sub-channels. The digital code is split into many streams, with one stream sent on each sub-channel [8]. Consequently, the data carriers in the COFDM can use QPSK or different levels of QAM modulation. This method stands up to multipath interference and unwanted reflections can be recognised and rejected.

In the early 1990s, there was a system, in Europe, called HD-MAC, which failed. It was a project trying to co-ordinate the launch of a hybrid analogue-digital High Definition Television (HDTV). Learning from it, DVB has chosen the standard-resolution television of 625 lines, 50 Hz interlaced pictures. HDTV is stated as optional until it becomes a market necessity [7]. In addition, the wide screen aspect ratio stated is 16:9.

The audio standard selected is MPEG-2 sound (i.e., six channels). Initially stereo sound (i.e., two channels) can be used instead, because of the lack of multichannel MPEG decoder. Stereo, anyway, can be upgraded to multichannel surround (i.e., another channel of rear information is mixed with the stereo pair) [7].

DVB has stated, also, the following [44]:

- Service Information System (DVB-SI): to provide a comprehensive identification system for the transmission parameters and service content.
- Conditional Access Common Interface (DVB-CI): to control the access to some programs to unauthorised viewers.
- Teletext transmission transport system (DVB-Text): to carry fixed-format teletext system in the MPEG-2 transport stream.

MHP is a subproject of DVB starting in 1997. While DVB activities are more related to physical and transport layer, MHP concerns with the upper system layer [15]. The scope of DVB-MHP is to define a common API, which provides a platform independent interface between applications from different providers and the manufacturer specific hardware and software implementation, which is called middleware [45].

DVB-MHP has defined three different profiles, which state functional requirements for supporting a wide range of application types. The profiles are enhanced

broadcasting, interactive broadcasting, and Internet access. The first two are already developed and MHP is working on the third one [45]. These profiles are depicted in the figure 12.

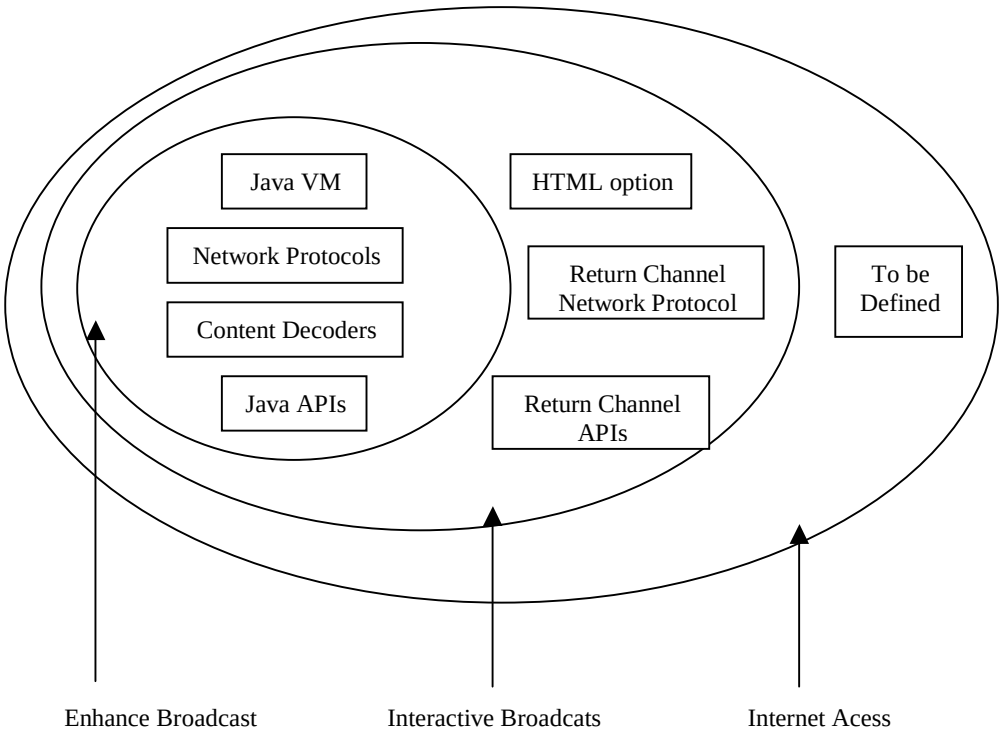


Figure 12. MHP Profiles [45].

The major components of the MHP are the following [45 and 46]:

- DVB-J platform: comprises a Java Virtual Machine and a set of specific television oriented APIs.
- Content Formats: including PNG, JPEG, MPEG, subtitles, and resident downloadable fonts.
- HTML: allowing the execution of HTML decoder written in Java, where there isn't an HTML decoder resident in the MHP receiver.
- Network Protocols: mandatory transport protocols, which are DSM-CC object carrousel (broadcast) and IP (return channel).

Also, from the applications point of view, a security framework and the application model and lifecycle is defined. The security model covers both the broadcasting and data authentication (e.g., digital signature), and the return channel encryption (using TLS). In DVB-MHP, a DVB-Java application is called Xlet application. The lifecycle

of an Xlet is similar to that used by Java applets in Internet, with four different states (i.e., ready, running, paused, and dead) [45 and 46].

Finally, it is necessary to explain, what the APIs DVB-MHP defines. The APIs, as said above, are used by an application to access the platform, where they are running [47]. The APIs of DVB-MHP can be categorised into [45 and 47]:

- Core Java: a subset of packages from JDK1.1.
- JavaTV: Some packages selected from JavaTV API developed by Sun Microsystems.
- HAVi: The User Interface API from HAVi specifications.
- DAVIC: Mostly addressed to MPEG transport and related subjects (e.g., tuning, conditional access). Also, a number of extensions of the JMF (Java Media Framework).
- DVB: including packages to address the broadcast transport files, user settings and preferences, DVB service information, and DVB related extensions to the JMF.

As a conclusion, it would be interesting to talk about the work DVB and MHP are carrying on. One important subject DVB is working on is the Multipoint Distribution System (MDS), which actually is a wireless cable delivery system [44]. MHP group has the MHP 1.0, Enhanced profile, accepted by ETSI. MHP 1.1, interactive profile, is already accepted. Now the group has the intention of develop the last part of the MHP specification, Internet profile, finishing the work started in 1997 [45].

3.1.2 ATSC Grand Alliance

In almost all the countries of the American Continent the ATSC Grand Alliance standard will operate. ATSC is under the aegis of the US Federal Communication Commission (FCC). Consequently, the first obvious difference between DVB-MHP and ATSC, that can be point out, is the non-politically independence of the latter [7].

FCC has decided to allocate 6 MHz channel, less than the European standard (8 MHz). In addition, the modulation system is also different. ATSC Grand Alliance uses the 8-VSB (8 Vestigial Side Band) system. 8-VSB states that the lower side band is virtually eliminated and only the upper one is transmitted [8]. The sound system used is the Dolby Digital, formerly called AC-3 (i.e., six sound channels) instead of the stereo sound determined by DVB [7].

ATSC has given more importance to offer the viewer a high-resolution system instead of the one DVB uses (625 lines). The standard taken by USA is the computer display standard (i.e., call pictures to be transmitted at a rate of 24, 30, or 60 Hz and the 60-field/30-frame system). With a resolution of 1280-1920 pixels per scanning line and 720-1080 scanning lines. Also, it has to be said that the provision is made

for progressive scan display of 60 full resolution pictures per second instead of 60 interlaced fields' [7].

ATSC Grand Alliance has agreed a layered architecture. Four layers are defined: picture, compression, transport, and transmission. An overview about them is the following [48]:

- Picture: provides multiple picture formats and frame rates.
- Compression: using MPEG-2 video and Dolby AC-3 audio compression.
- Transport: based on MPEG-2.
- Transmission: using 8-VSB.

ATSC is, also, working on a television middleware standard. But in USA, alike in Europe, broadcasters and cable operators are developing their own specification. In 1997 broadcasters started the Digital TV Applications Software Environment (DASE). On the other hand, cable operators, forming CableLabs, are developing the OpenCable Applications Platform (OCAP).

Cable operators have agreed that DVB-MHP is to become the core of the OCAP. OCAP has two major components. One is the Execution Engine (EE), which provides a programmable environment. The other is the Presentation Engine (PE), similar to a Web browser, which will provide support for creating and using the Web's standardised markup and scripting languages.

From the beginning, ATSC offered CableLabs to join efforts with DASE, but the cable operators declined the offer and chose its own path. Now, both groups have initiated discussions to harmonise the two standards. Both standards share common functionalities and reliance on Sun Microsystems' Java to run applications, but have significant difference in transport level.

As a conclusion, the prime difference between ATSC Grand Alliance and DVB-MHP is the guidelines used to develop each of them. In the case of DVB-MHP, it is done as market oriented. On the other hand, ATSC is agreed to offer the viewer the best possibility technology can offer.

3.1.3 ISDB

The Association of Radio Industries and Business (ARIB) in Japan has developed a unified standard for digital television. It is called ISDB [49].

ISDB-T (i.e., terrestrial standard) uses a modified COFDM for transmission. With several thousand sub-channels in 13 groups carrying the TV signal in different resolutions. When reception is poor, the definition is less but still there is something to watch (In ASTC or DVB the pictures are good or non-existence). The system is called BST (Band Segmented Transmission) OFDM [8].

It also states the sound system to be ACC (Advanced Audio Coding). This coding is developed by the Franhoufer Institute (one of the developers of MPEG audio), as well as by Dolby, Sony, and AT&T [8].

Finally, ISDB-T is supposed to deliver information to three different types of receivers. The first one is the integrated receiver (i.e., television at home), with a HDTV display resolution. The second one is a mobile receiver, with standard television display resolution. The last one, is a lightweight portable, in which only sound and data services are transmitted [49].

3.1.4 Current situation world-wide

The actual situation in relation with the existing standards, in digital terrestrial television, in the different countries is depicted in the table 1. There are two different tendencies, European and USA model. Then, Japan stands alone with an own standard. It is interesting to remark the situation of Australia. Australia usually belongs to the USA group; this time it has chosen DVB-MHP as standard for digital television.

Argentina	ATSC	Waiting tests results with 8-VSB/COFDM
Australia	DVB	Includes Higher definition pictures & Dolby digital sound
Brazil	-	To decide
Canada	ATSC	Waiting test results with 8-VSB/COFDM
China	-	Both ATSC/DVB are being considered
Europe (all)	DVB	
India	DVB	With Dolby Digital surround system
Japan	ISDB	
Mexico	ATSC	Waiting tests results with 8-VSB/COFDM
New Zealand	DVB	
The Philippines	-	ATSC is being considered
Russia	DVB	
Singapore	DVB	With Dolby Digital surround system
S. Korea	ATSC	Waiting tests results with 8-VSB/COFDM
Taiwan	ATSC	Waiting tests results with 8-VSB/COFDM
Thailand	DVB	
U.S.	ATSC	Follows ATSC

Table 1. Situation of terrestrial television in the world [8].

3.2 Standards Related to Digital Television

There are four very important standards related to the digital television. Each of them is covering a different scope. The first one is MPEG, stating how to transmit video, audio, and data in the same stream. The second one is HAVi, specifying the home-network standard. The third one, to be studied, is DAVIC, in charge of specifying some APIs, which DVB-MHP has later included in its specification. The last one, only in the Nordic region, is NorDig, settling a common platform for digital television.

3.2.1 MPEG

MPEG is a working group of ISO/IEC, established in 1988. This work is studying how to improve the coded representation of the digital audio and video [37].

MPEG has developed the following standards [37]:

- MPEG-1: for storage and retrieval of moving pictures and audio on storage media. Products as Video CD and MP3 are based on it.
- MPEG-2: used on digital television and DVD.
- MPEG-4: for multimedia for the web and mobility.

Nowadays, MPEG group is working on MPEG-7, content representation standard for multimedia search, filtering, management and processing. It has also started to work on MPEG-21, the multimedia framework.

3.2.1 HAVi

Eight Consumer Electronics companies founded the HAVi organisation. They have defined the home-network architecture.

As the HAVi architecture was explained in the first chapter, here is important to express the part related with the DVB-MHP specifications. The User Interface (UI) part of the HAVi specification is included in the DVB-MHP standard. This way, HAVi UI API is a part of the DVB-MHP. This API is a subset of Java 1.1 with some additional functionality [34]. But, since the thesis' aim is to develop some parts of the API, this particular API will be studied in more detail in the next chapter.

3.2.2 DAVIC

DAVIC is a not profit-making association, founded in 1994 and based in Geneva. The task of this association is to promote digital services transmitted in broadband. It has created a standard for end-to-end interoperability of broadcast and interactive digital audio-visual information, and of multimedia communication [38].

There are some facts about the DAVIC philosophy that can be pointed out [50]:

1. Tools and not systems: specify non-system-specific components (tools) to ensure the interoperability.
2. Relocation of tools: components can be used in different systems as well as in different parts of the same system.
3. One functionality, one tool: components are defined to be unique.
4. Specify the minimum: only the very minimum is required to satisfy a multi-industry constituency.

Some of the APIs included in the DVB-MHP specifications are the following [47]:

- *org.davic.awt*: including the alpha channel.
- *org.davic.media*: including the language to be used and media events.
- *org.davic.mpeg*: including how to deal with mpeg stream.
- *org.davic.net*: including the tuning and resources management.

DAVIC next initiatives are two. The first one is to achieve TV Anytime and TV Anywhere. The second one is to complete the convergence of the digital television with Internet [38].

3.2.3 NorDig

NorDig is a co-operative organisation constituted on 5th November 1997, in Odense, Denmark. NorDig specify a common platform for digital television within the Nordic region [39].

Finland is going to use NorDig II specifications. It is intended for cable, satellite, and terrestrial networks. It covers the I/O specifications the STB has to fulfil as well as other issues such as the Navigator, the Teletext, the remote control or the interfaces for conditional access [13].

3.3 Other Organisations

There are two other kinds of organisations, which are important from the perspective of this thesis. On one hand, there are standard organisations. On the other hand, there are organisations helping on the shift from analogue to digital. The latter ones are DigiTAG and DTG.

3.3.1 Standards Organisations

The European Telecommunications Standard Institute (ETSI), is a non-profit organisation whose mission is to produce the telecommunication standards. DVB has given to them all the drafts to become standards [51].

The European Committee for Electrotechnical Standardisation (CENELEC) was set up in 1973 as a non-profit organisation. It has been recognised as the European Standards Organisation in its field by the European Commission in Directive 83/189/EEC [52].

Finally, the European Broadcasting Union (EBU) is the largest professional association of broadcasters in the world [53].

3.3.2 DigiTAG and DTG

DigiTAG was launched in 12 September 1996 in Amsterdam. The basis of this organisation is that DVB-T was successful specification, but not sufficient. The organisation is working on to make smooth migration from analogue to digital terrestrial television [40].

DTG is working in the UK to ensure that the DVB-T specifications are being followed in the way of developing the digital terrestrial television [41].

4 APPLICATIONS FOR DIGITAL TELEVISION

DVB-MHP selected HAVi Level 2 GUI framework for developing applications' User Interface [46]. The user interface design is a crucial issue, since it provides the means for the human-system interaction. The usage of a GUI framework for this task is beneficial. The developer does not have to deal with the underlying system, but can use a set of widgets already implemented.

There is a lot of literature centred on how to develop efficient User Interfaces depending on the system and user needs. Applications for digital television are oriented to a broad spectrum (i.e., heterogeneous group) of people. A GUI approach, since it reduces the effort needed to interact with the system, is the most suitable approach [51].

Java language was decided as the programming language to develop digital television applications [14]. There are many GUI toolkits implemented for Java. Abstract Windowing Toolkit (AWT) was part of the core packages [52]. As the use of Java grew, many other widget toolkits appeared. The most commonly used is Swing, developed by Sun Microsystems. Other includes JavaBeans Component Library (JBCL), developed by Borland, or Biss-AWT [53].

The toolkits mentioned in the above paragraph are general-purpose GUI frameworks. Hence, they cannot fulfil all the needs of specific devices, such as digital television. For this reason, a television oriented GUI framework was needed. DVB-MHP selected HAVi User-Interface because it can be used on a variety of CE devices. Furthermore, it is designed as a "TV-friendly" framework [46].

This chapter is structured as follows. First, in sub-chapter 4.1, a study about User Interface design is done, focusing on GUIs. Then, in sub-chapter 4.2, Java language and Java GUI frameworks are studied. Finally, in sub-chapter 4.3, HAVi, as a GUI framework for digital television, is explained.

4.1 User Interfaces

User Interfaces design is an essential area in the development of applications. It studies the way the user and the system communicate with each other. It is out of the scope of this thesis to do a deep study about it, but some of its main concepts should be pointed out.

The first concept refers to the different dialogue styles determined in human-computer interaction. The second one refers to the ideas behind a GUI. After understanding both, it will be clear the importance of using a GUI in digital television applications.

This section is structured in the following way. First, in sub-section 4.1.1, human-computer dialogue styles are studied, focusing on which is the most suitable for digital television. Then, in sub-section 4.1.2, the general aspects of a GUI are determined.

4.1.1 User Interface design, an overview

User interface design refers to the study of the interaction between the user and a system. It covers multiple fields, such as Cognitive Science, Mathematics, and Ergonomics. This sub-section focuses on direct manipulation as the dialogue style to be used, when developing User Interfaces for digital television.

Dialog is the process of communication between two or more agents. In human-system, as in human-human, communication, a dialogue is needed to achieve interactivity. In other words, the system has to understand what the user want to do and the user has to understand what the system can do. User Interface design defines several human-system dialogue styles. Two examples are command language and direct manipulation [54].

Direct manipulation refers to a dialog style, in which the user's actions directly affect what happens on the screen. Hence, there is a feeling of physically manipulating objects on the screen [55]. Shneiderman suggested this model in 1974. Since then, it is commonly used for systems oriented to a greatly diversified population, such as digital television, because of its ease of learning and use.

4.1.2 GUI

GUI supports direct manipulation dialogue style. The objects in the screen, which can be manipulated, represent interaction alternatives with the system. This way, the system shows content and gives options to the user by placing items on the screen. Then, the user can select among them, to determine what she wants to do next.

GUIs are built around a toolkit of widgets [56]. These widgets (e.g., Button or List) are "ready made" screen items. These are the objects mentioned in the above paragraph. The user interface designer has to make use of them, depending on the task at hand. First, she has to choose, which items are going to be used. Then, she has to decide, where to place them on the screen.

GUI elements can be divided into two categories: components and containers widgets. Container refers to a region, in which the components are placed. Usually, containers are components, but components do not have to be containers. *Canvas* is an example of a container, while a *Button* is an example of a component.

Two main ideas should be remarked, which make interactivity possible. The first one is referred to what the user see (i.e., the visual presentation of the widget). The

second one is referred to how a widget reacts to the user interaction (i.e., behaviour of the widget). These are called look and feel respectively.

As explained before, a GUI is composed of some objects on the screen. It gives options and shows content to the user. Hence, every GUI widget needs of both look and feel. A widget has to be displayable on the screen and behaves in a certain way. It has to be remark that a widget can also have static behaviour (i.e., only show content).

4.2 Java

Java is a programming language based on the "object oriented" paradigm. Java started as a language developed to address the problem of building software for networked consumer devices [57]. Hence, portability and reliability were the main goals.

Java, also, offers a GUI framework. A GUI framework is a set of classes and interfaces, which are used to develop GUI easily. It includes widgets, which can be placed on the screen. It also includes layout managers, which controls the layout of the widgets within their container. Finally, it includes the means to paint.

As studied in sub-section 4.1.2, the look and the feel are the main important aspects to achieve interactivity. Hence, Java includes both of them. Look is offered by rendering methods of the widgets. The widgets' behaviour can be controlled by event-handling mechanisms.

This section is structured as follows. First, in sub-section 4.2.1 an introduction of Java language is done. Then, in sub-section 4.2.2, GUI frameworks for Java are studied. Followed, in sub-section 4.2.3, by how rendering is done in Java. Finally, in sub-section 4.2.4, the event-handling model used in Java is explained.

4.2.1 Introduction to Java

Java was developed with two main goals in mind, portability and reliability. The key factor of Java is the use of a Java Virtual Machine (JVM) [58]. JVM is an abstract computing machine. It has an instruction set and uses various memory areas. It knows nothing about Java, but only understands JVM instructions (i.e., bytecode).

Because of the use of the JVM, Java is a platform-neutral language. A Java compiler generates a Java *Class* file. This file contains bytecode, a symbol table, and other ancillary information. Since, *Class* files use bytecode, it is not dependent on the underlying system instructions. In conclusion, JVM provides a level of abstraction from the Operating System, hence portability.

In addition to JVM, the Java platform core classes are needed to run Java applications. These classes and the JVM forms the JRE mentioned in sub-section 2.1.4. These classes are grouped forming packages. In other words, a package consists of a number of compilation units and has a hierarchical name. Some examples of the core packages are *java.lang*, which includes the core language classes, and *java.io*, which focuses on input and output and some file system manipulation [57].

JRE includes some packages, as the ones mentioned in the paragraph above. These packages can be used, when developing applications. This thesis will focus on the GUI packages. A GUI package should provide widgets, both containers and components, classes needed to support these widgets and their rendering, and event handling mechanisms for them.

4.2.2 Java GUI frameworks

AWT was included in Java core platform packages. After, some other GUI packages appeared. Some examples are Sun's Swing, and Microsoft's Application Foundation Classes (AFC). When developing new frameworks, two decisions have to be taken. The first one, whether is convenient to extend AWT, as Swing, or develop from scratch, as JFC. The second one studies how to render the widgets.

Widget can be, depending on how it is rendered, heavyweight or lightweight. Heavyweight is one that is associated with its own native screen resources, commonly known as a peer [59]. Lightweight is one that "borrows" the screen resource of an ancestor. Hence, it has not native resource of its own (i.e., it is called peerless).

In the following paragraphs both AWT, which uses heavyweight widgets, and Swing, which uses lightweight ones, are studied. After, a comparison between both approaches is done.

AWT
In AWT, for every widget, there is a corresponding native code counterpart, called peer object, which manages a native component in the underlying OS. For this reason, every time a widget is created, the peer object works with the OS to create one of its interface widget.

Swing
In Swing, on the other hand, each component extends *java.awt.Component* and draws itself on its parent container (i.e., the window). It is a pure Java widget, which does not need any peer object.

Comparison
Although Swing widgets do not need any peer object, a native screen resource is needed at some level. In other words, these widgets have to be painted somewhere. In Swing, this is done at the window level. Hence, the number of interactions between Java and the OS is reduced. In the three next paragraphs, what implies the use or not of peer objects is studied.
It is obvious that the use of a native peer to render speeds the performance [60]. But since AWT creates one peer per widget the speed is, at the same time, slowed down. First, there is too many interactions needed between Java and the OS [61]. In addition, the number of classes the JVM has to manage is much more than in Swing (i.e., the widgets and the peers versus window peer and widgets).

In addition, the use of native peers creates platform specific limitations, which directly affects portability, one of the goals of Java language. The native widgets from one OS to another vary in appearance and behaviour. Hence, porting AWT from platform to platform is not as easy as it should be.

Finally, the look and feel of a widget, and the transparency of some of its parts should be remarked. In a heavyweight approach, the widgets reflect the look and feel of the underlying OS. On the other hand, in a lightweight environment, the developer can decide the look and feel depending on the needs. Moreover, the lightweight component can have transparent areas since it paints itself, while the heavyweight cannot.

4.2.3 Java Graphics

To paint widgets in Java, the rendering code should be placed inside a particular method. The method is called *paint(java.awt.Graphics g)* and it is located in *java.awt.Component* [62]. When it is time to paint, the toolkit will invoke this method. First, the class *java.awt.Graphics* should be studied. Then, how to place code inside *paint(Graphics g)* and when it is invoked is considered.

java.awt.Graphics class provides the framework for all graphics operations within AWT. It plays two different roles. First, it is the graphics context. A graphics context stores all the information needed to draw, such as background and foreground colours, font, dimension and location of the clipping rectangle. Second, it also provides methods for drawing simple geometrical shapes, text, and images [63].

In sub-section 4.2.2, there was a study about heavyweight and lightweight widgets. The former relies on the paint subsystem of the native platform. Hence, the peer is in charge of painting and each widget does not have to override *java.awt.Component*'s *paint(java.awt.Graphics)* method. In the latter, *paint(java.awt.Graphics)* method has to be overridden to determine the actual look of the widget, making use of the drawing methods provide by *java.awt.Graphics* class.

Finally, in Java, the widgets are added to a container. The container represents a region, in which they can be painted. These widgets, once they are added, become "children" of the container. To make visible the GUI, the container invokes the *paint(Graphics g)* method of each of its added widgets.

4.2.4 Java Event handling

Interactivity is reached, when the user can modify the state of the running program. The Java run-time system creates events, when a user action occurs. An event is actually an object storing some information (e.g., event ID, who has originated it). After the event is created, the application has to process it.

Java language defines several models of event handling. In the beginning, Java 1.0 Inheritance Event Model was proposed. In this model, the event was sent, to the component that generated it, to be processed. If the component didn't consume the event, it was propagated further up the inheritance hierarchy until it was either consumed or the root of the hierarchy was reached [64].

Some drawbacks from Java 1.0 Model were the following. All the components of a GUI had to handle events, either to consume or to let them go up the hierarchy. Events were delivered to components, which were not interested on them at all, wasting CPU cycles on uninteresting events. For the reasons introduced in the paragraph above, Java 1.1 Delegation Event Model was introduced.

Java 1.1 Event Model is based on the concept of an "Event source", which generates events, and "Event Listeners", objects interested in receiving events [65]. A widget is an "event source". It maintains a list of listeners. A listener is in charge of handling the event. A widget provides methods, which allow the listeners to add or remove themselves from the list. When an event reaches the widget, it delivers the event to all the registered listeners.

Sometimes, the events available are not enough for the task in hand. In Java, new events types can be created [66]. If one wants to create a new event type, she has to do the following. First, create the new event type class. Second, create a listener for the new type. Third, modify the component, so a listener of the new event can be added. Finally, create an object, which can manage multiple listeners (i.e., a list of listeners).

4.3 HAVi Framework

A digital television oriented GUI framework was needed to develop applications. Java, as mentioned above, was selected as programming language. The problem was that all the Java oriented GUI frameworks available were general purpose. Hence, digital television had some special features these frameworks did not cover. For this

reason, DVB-MHP selected HAVi Level 2 User Interface [46]. So, special requirements of a digital television GUI could be implemented.

HAVi is an organisation founded by eight Consumer Electronic organisations, as mentioned in sub-sections 2.3.2 and 3.2.1. They aim is to develop a common API to achieve the interoperability between electronic devices. HAVi Level 2 User Interface has two versions. Version 1.0 appeared on January 18th 2000. Version 1.1 is from May 15th 2001. In this thesis, HAVi 1.1 is studied since it is part of the DVB-MHP specification 1.1.

HAVi, unlike AFC, decided not to start from scratch. Hence, it uses JDK 1.1 *java.awt* package as its core. For this reason, some of the classes from the package are HAVi compliant and the most are not [46]. It was decided to maintain the main classes, such as *java.awt.Component*, since they are needed to build lightweight widgets. Also, classes, such as java.awt.Color and java.awt.Font, are required for general drawing and painting. Finally, the layout classes are also retained to provide flexible layout on output devices.

HAVi, like Swing, is a lightweight framework. A lightweight framework, as studied in sub-section 4.2.2, means that the widgets do not rely on a peer class, but are pure Java implemented. Hence, the visual presentation and the event handling of the events are not determined by the underlying system, but are decided by the developer. Also, HAVi includes some specific television concepts (e.g., remote control input, television screen definition, applications' containers, and the transparency of the widgets).

This section is structured as follows. First, in sub-section 4.3.1, the possible user inputs are determined. Then, in sub-section 4.3.2, HAVi screen is explained. Then, in sub-section 4.3.3, the containers needed for applications are studied. After, in sub-section 4.3.4, matte concept is introduced.

4.3.1 User Input

Usually, the remote control is the input device required for television. In HAVi, remote control events are supported extending *java.awt.event.KeyEvent* class. Although Keyboard and Mouse events are PC specific, the former is fully maintained and the latter is optional.

In the Nordic countries, the remote control capabilities are defined in NorDig II specification [13]. The keys can be divided into three categories: numeric entries, basic TV functions, and digital TV functions. Numeric entries include 10 digit keys, labelled 0 to 9. Basic TV functions include Power, Program up and down, volume up and down, and TV key, which puts the IRD directly into conventional TV state. Digital TV functions include a pointing system to navigate on the OSD, OK key to confirm a choice, application keys to access directly to a service (i.e., navigator,

Teletext, EPG/Guide, default interactive application), back key to return to the previous state, and four coloured keys for non-dedicated functions. Figure 13 depicts the remote control capabilities.

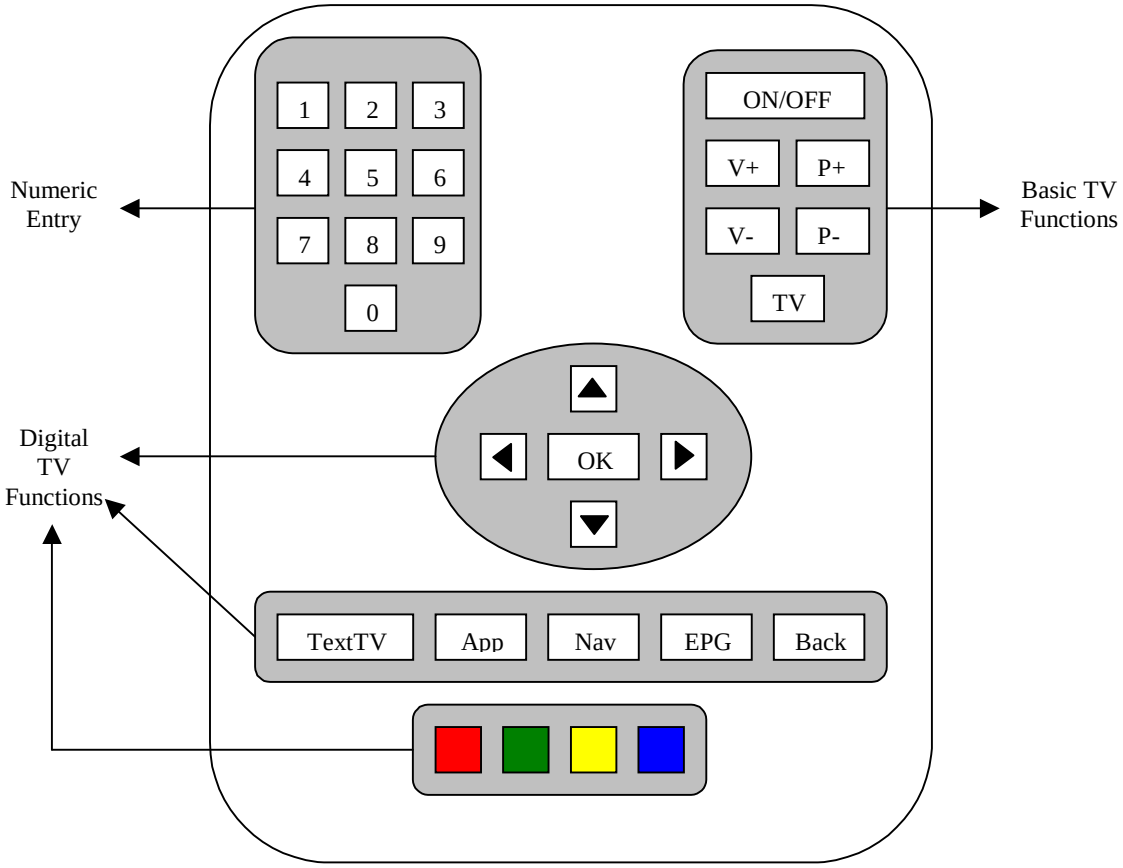


Figure 13. NorDig II remote control specification**.**

HAVi defines two kinds of remote control input events. Six coloured keys (i.e., not dedicated keys) form the first set. Dedicated keys form the second group (e.g., Power key, Volume Up, Teletext). Hence, the remote control functionality defined by NorDig II can be mapped with the remote control events determined by HAVi.

4.3.2 Screen

In HAVi, three different layers form the final television output. The first one, graphic, is where the applications are placed. The second one, video, is where the actual video content is located. Finally, the third one, background, is the area that is behind the running graphics and video and not covered by those. Following HAVi

specification, the background can be defined as a single colour or even as a still image.

HAVi names the layers mentioned in the paragraph above display devices. Hence, graphic, video, and background display devices are integrated in order to acquire a screen to be shown. It is possible determining relationships between them. Moreover, one can determine a set of constrains between video and graphics. For example, an application running some video might want the video display device to fit the running video. In addition, it can be decided to request transparency between graphic and video display devices.

It is important to take into account that the current set-top boxes are able to output in a number of different configurations. Hence, an important requirement HAVi needs to fulfil is to determine the current configuration and detect modifications on it. HAVi uses configuration classes for these purposes. A configuration class represents distinct possible configurations of a display device.

A configuration class is associated to each display device (i.e., graphic, video, and background). Hence, applications can request the available display device configuration. Also, they can ask for configurations matching a set of constrain. In addition, to detect modifications on the configuration, HAVi defines screen events and listeners.

4.3.3 Scene

Scene represents the displayable area on the screen where the applications can be rendered and interact with the user. Scene is the main top-level component of the application. Hence, as discussed in sub-section 4.2.2, although HAVi is a lightweight framework, Scene needs of a native peer. Scene is transparent, but is in charge of painting all the added children.

Applications can ask for the preferred location and size on the television screen via templates. Template is a class, which indicates the size and the location of the scene. Once the application has requested it, a scene factory tries to find the closest option to the needs.

4.3.4 Matte

Matte is defined as an object carrying the transparent information associated to an image. In HAVi, every widget, including container and components, have an associated matte. Matte is useful for many purposes, such as grouping widgets. When a container is grouped, its matte influences all its members appearance. But, when it is ungrouped, its matte only influence the region not covered by its members.

5. IMPLEMENTATION

Peng et al. [67] and Sivaraman et al. [68] have proposed a digital television environment, which is MHP compliant. It includes hardware, system software, a Java Virtual Machine (JVM) and set of APIs needed, an application manager, and three different services covering the service architecture proposed by Vuorimaa [18]. This environment lacked of a digital television GUI framework implementation.

When developing digital television GUIs applications without a specific GUI framework, the developer has two options. The first one is to use a general purpose one. The second one is to create application specific widgets. Both of these options have some drawbacks.

If a general-purpose framework is selected, the widgets to use will not be TV-friendly. In addition, a package of this kind is huge, because it provides much more functionality than needed for digital television. If application specific widgets are implemented, these cannot be used in other applications. Moreover, the developer has not only to develop the GUI, but the widgets to use every time a new application is required.

The current research aim was to develop a GUI framework oriented to digital television. Following this way were decided due to two reasons. First, the environment, mentioned in the first paragraph, lacked of it. Second, the drawbacks of either using a general-purpose framework or developing application specific widgets mentioned in the last paragraph. Then, the available environment was used for testing purposes.

This section covers both parts of the process. It is divided into two sub-sections. The first one, sub-section 5.1, explains the actual implementation of the GUI framework. The second one, sub-section 5.2, discusses a case study of the framework.

5.1 GUI Framework Implementation

The framework implemented was named *ftv* package. *ftv* followed HAVi specification. Hence, it was a lightweight framework based on *java.awt* package. *ftv.FTVComponent*, which is the top level class of the component widgets, extends *java.awt.Component*. *ftv.FTVScene*, which was the container, was a subclass of *java.awt.Container*.

The two main features of *ftv* were separation of look and feel, and definition of specific new events. The look and the feel were separated by the use of a look class, which was associated to each widget. Each widget required of a look class in order to be rendered. In addition, new events were defined, forming *ftv.event* package.

This section is structured as follows. In sub-section 5.1.1, the container is described. In sub-section 5.1.2, *ftv.FTVComponent* is studied. In sub-section 5.1.3, the classification of the different behaviours, which the widgets can have, is done. In sub-section 5.1.4, the possible states the widgets can reach during run time are defined. In sub-section 5.1.5, the different available look classes are discussed. Sub-section 5.1.6 describes how the events are handled. Finally, in sub-section 5.1.7, the different components developed are explained.

5.1.1 Container

Container refers to the place, where the components are added. This way, the added widgets become "children" of the container. In *ftv* package, the container, *ftv.FTVScene*, extended *java.awt.Window*. Its main tasks were handling window events and painting its child components. Since, *ftv.FTVScene* is transparent, it painted the "children" by calling the *paint(Graphics g)* method of each contained widget.

ftv.FTVScene also supported Z-ordering of the widgets. This capability is required, since the widgets in *ftv* are explicitly allowed to overlap each other. The Z-ordering capabilities, regarding to widgets, are the following: addBefore, addAfter, pop, popInFrontOf, popToFront, push, pushBehind, and pushToBack. This methods, were implemented by implicitly reordering the child components within *ftv.FTVScene*.

5.1.2 FTVComponent

ftv.FTVComponent extends *awt.Component*. Since, new events were defined in *ftv*, besides *java.awt* ones, *processEvent(java.awt.AWTEvent evt)* was overridden to facilitate the generation of events in *ftv.event* package from *java.awt.AWTEvents*. The events used in this way from *java.awt.event* were not consumed and, hence, propagated to the super-class.

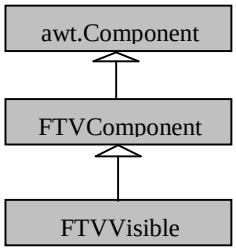


Figure 14. FTVVisible, of *ftv* package, diagram.

ftv.FTVVisible extended *ftv.FTVComponent*, it indicated that a widget is displayable. Since in digital television GUI all the widgets should be displayable, all of them were originated from *ftv.FTVVisible*. Figure 14 is an UML diagram of *ftv.FTVVisible*.

5.1.3 Feel

The feel of a widget denotes its behaviour. That is, the way the widget reacts to the user interaction. In *ftv* package, as mentioned in 5.1.2, is originated from *ftv.FTVVisible*, Hence, every widget can be displayed on the screen. In addition, three more different behaviours are available: navigable, actionable, and switchable. In *ftv*, these behaviours are represented by interfaces. Figure 15 shows and UML diagram of these interfaces.

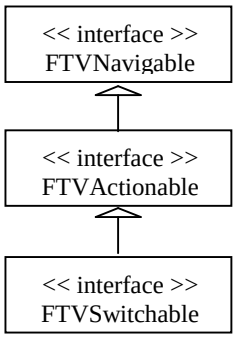


Figure 15. Possible widget behaviours in *ftv* package.

Navigable
Denotes a component, which can acquire the focus. That is, the user can navigate among a set of navigable widgets.

Actionable
Denotes a component, which can be selected. It has an associated functionality. The functionality is started when the widget is chosen. An example is a button.

Switchable
Denotes a component, which can be selected and retain its internal state information. An example is a toggle button.

5.1.4 State of the widget

Depending on the behaviour associated to a widget, it can reach different states during run time. There are up to four different states: normal, focused, actioned, and actioned focused.

Normal
The state when the widget is only shown on the screen. Since every widget is visible, all of them can use this state.

Focused
The state when the focus is on the widget. That is, when the user has navigated over it. Only navigable (or its subclasses) widgets can use it.

Actioned
The state when the widget is selected. Restricted to actionable (or it subclass) widgets.

Actioned focused
The state when the widget has been selected, so the associated action has been started, and it gets the focus. This state can only be used by switchable widgets.

In addition it is important to remark that, in *ftv*, different states can have different content associated. Hence, the look class renders the content depending on the current state. In *ftv*, different colour borders identify different states. Figure 16 depicts the different states of a switchable widget, since it is the most general case.

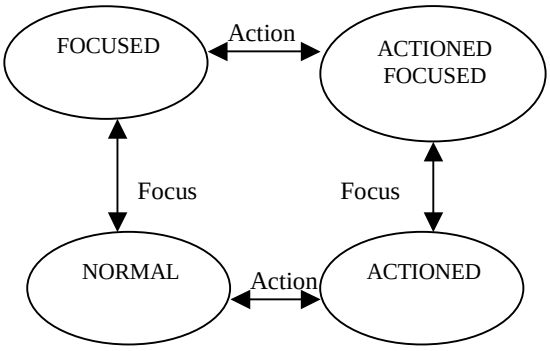


Figure 16. States for Switchable widgets in *ftv* package.

5.1.5 Look

Look refers to the visual presentation of a widget. Each widget needed of an associated look class. This class was in charge of rendering it. When a widget is created a default look, depending on the kind of component, was associated. But, *ftv* package allowed widgets to select a different look class. This way, a developer can choose the visual presentation of the widgets depending on the application at hand.

When a widget needs to be painted, its *paint(Graphics g)* method is invoked. This method calls *showLook(Graphics g, FTVVisible visible, int state)* method of the

associated look class. In this method, *visible* refers to the widget to render and *state* to the current state of the widget.

Double buffering is a technique used to avoid screen flickering. It was used in *ftv* package to a better rendering of the images. To handle double buffering, one must first draw the widget's contents in an off-screen image and then transfer the contents of the image to the visible window. In *ftv* package, instead of using this technique in every look class, it was used at *ftv.FTVVisible* level. Figure 17 shows the Java code used.

```
Graphics offg;
Image offImage = null;
Dimension dim = size();
offImage = createImage(dim.width,
dim.height)
offg = offImage.getGraphics();
offg.setColor(this.getBackground());
offg.fillRect(0,0,dim.width, dim.height);
offg.setColor(this.getForeground());

// ftvlook is the look class associated
ftvlook.showLook(off, this, state);
g.drawImage(offImage, 0, 0, this);
```

Figure 17. Double buffering code in *ftv* package.

In *ftv*, the default looks available were animate, graphic, text animate, text, and time. Animate look defines the way to show a sequence of images. Graphic look determines how to show a single image. Text look is used to display text. Text animate look displays a sequence of text. Finally, time look renders a clock. Figure 18 shows an UML diagram of these looks.

In *ftv*, the look classes worked on the following way. First, the borders were painted, if needed. Then, the content is rendered. If the content is text, a text layout manager is used. In *ftv*, the text layout manager made use of a text formatter. It formats the text to the space given depending on the text to render and font selected.

Developers can also create specific looks depending on their needs. To test this feature, an additional look was implemented, *ftv.FTVGraphicTextLook*. It extends *ftv.FTVGraphicLook*. This new look paints both a background image and a text over it. An UML diagram of this class is shown in figure 19.

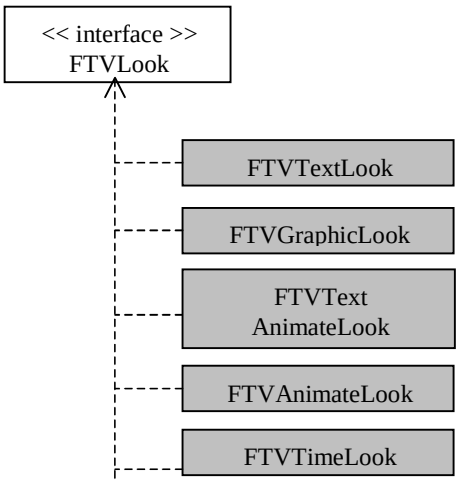


Figure 18. Default looks diagram of *ftv* package.

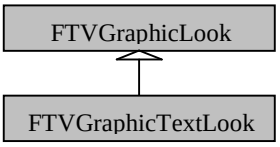


Figure 19. FTVGraphicTextLook of *ftv* package.

5.1.6 Event handling

Since *ftv* is based on JDK 1.1, it uses delegation event model, as explained in subsection 4.2.4. Hence, each widget maintains a list of listeners. It provides methods, which allow the listeners to add or remove themselves from the list. When an event reaches the widget, it delivers the event to all the registered listeners.

In *ftv* package, new events were defined, such as *ftv.event.FTVActionEvent*. Hence, new listeners were needed, such as *ftv.event.FTVActionListener*. Although the list of listeners can be maintained as a *java.util.Vector*, a special class was implemented, called *ftv.FTVEventMulticaster*, for this purpose. The widgets, depending on their feel, were allowed to add these new listeners. For example, navigable widgets implemented *addFTVFocusListener(FTVFocusListener l)* method.

ftv.event.FTVFocusEvent includes, besides AWT's lost and gained focus, transfer focus. Transfer focus determines that the focus have to be transferred from the current widget to another specific one. This way, the user can navigate among navigable widgets by the use of some code keys, determined by the developer. In *ftv*, a listener, who reacts to key code inputs, was added to navigable widgets. Also, the

widgets stored “When” (i.e., the key code that provokes the focus transfer) and “who to” (i.e., the widget to which transfer the focus) in a hash-table.

Finally, it has to be pointed out, that in *ftv* there are two sets of events, *java.awt* and *ftv*. This means that widgets will receive two events, the original *java.awt*, and the generated *ftv*. Ignoring the former ones and handling the latter ones solved this problem.

5.1.7 Toolkit widgets

The developed components can be divided into animate, graphic, and text widgets. Animate refers to those widgets, which make use of a player to sequence the information to be shown. Graphic refers those that only store images. Text defines those that store *String* Java type.

A consequent notation was used to differentiate the behaviour of the widgets. Static widget signifies that it is only visible. Navigable widgets are not static. Buttons are actionable widgets. Finally, Toggle Buttons is a switchable widget.

Animate widgets
Animate widgets, as mentioned above, need a player. The player is a thread, which has some features. It can be started and stopped as desired. Also, it can be played for a specific number of times or being an infinite loop. The playing mode can be both repeat or invert the order of the sequence each time the animation restarts. Finally, the speed of the animation can be stated. For rendering, the player uses the look class associated to the widget.

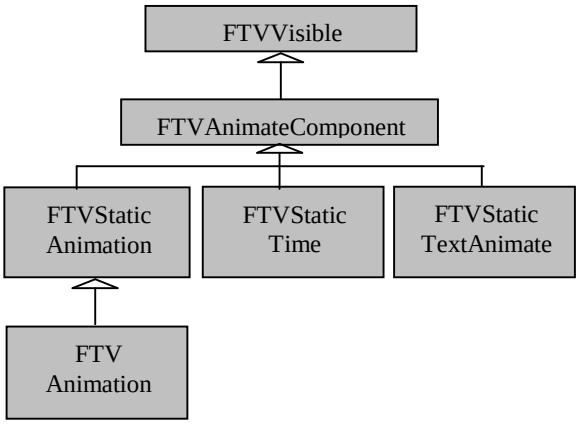


Figure 20. Animate widgets of *ftv* package.

Since HAVi defines animation only for image content, a class extending *ftv.FTVVisible*, which interacts with the player, was implemented. This class was called *ftv.FTVAnimateComponent*. The static animate widgets comprise a static

animation, static text animation, and static time. All of them extends *ftv.FTVAnimateComponent*. HAVi was modified because of the importance of both clock and text animation. A clock is necessary to show the current time to the user. Text animation can be used to show latest news, for example. Figure 20 is an UML diagram of the animate classes implemented.

ftv.StaticTextAnimate and *ftv.FTVStaticTime* store text information because, in Java, a date can be transformed to text. For this reason, their look class used the text layout manager and the formatter mentioned in sub-section 5.1.2.

Graphic widgets

Graphic widgets are those, which store an image. They can be static icon, icon, graphic button, or toggle button. Figure 21 shows an UML diagram of them. Toggle buttons can be grouped forming a toggle group. This group behaves as a group of radio buttons, a maximum of one button is chosen at any time. If decided, the group can, also, behave as checklist, when at least one button is chosen at any time.

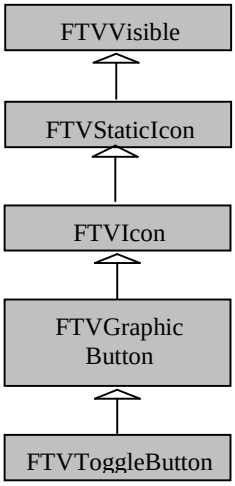


Figure 21. Graphic widgets of *ftv* package.

Text widgets

Text widgets are those that store *String* Java type. When creating the widget, the font, the text layout manager and, the background and foreground colour can be selected. Figure 22 shows a diagram of these widgets, which includes static text, text, and text button.

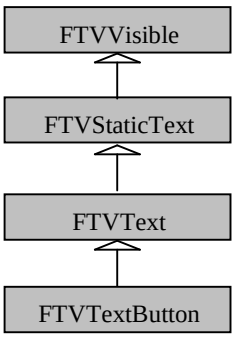


Figure 22. Text widgets of *ftv* package.

5.2 Case Study

As mentioned in 5, a MHP compliant digital television environment was available before starting the current research. The objective of this research was oriented to fulfil the lack of a GUI framework implementation oriented for digital television. As explained in 5.1, the package *ftv* was implemented. After, it was tested with the available environment.

This section is structured as follows. First, in sub-section 5.2.1, the environment used is explained. Then, in sub-section 5.2.2, the modification of the existing applications to make use of *ftv* is determined. Finally, in sub-section 5.2.3, the obtained results are discussed.

5.2.1 Digital Television Environment

The available environment covers the layers mentioned in 2.1.4. These layers were hardware, OS, JRE, APIs and applications. Figure 23 depicts the environment. In this specific case, the options taken were [67 and 68]:

- Hardware: a board PC, 233 MHz processor, 32 MB RAM memory, and 64 MB ROM memory was used as a set-top box.
- OS: Linux without X-Windows system.
- JVM: Kaffe using framebuffer for rendering [69].
- APIs: core Java API, some specific API needed such as *xml* or *jmf*.
- Application Manager controlling the applications, while the system was running.
- Navigator: resident application (cf. 2.2.1).
- Digital Teletext: resident application (cf. 2.2.3).
- Interoperable Application: downloaded application from Internet. This application provides the user some services, such as Internet access

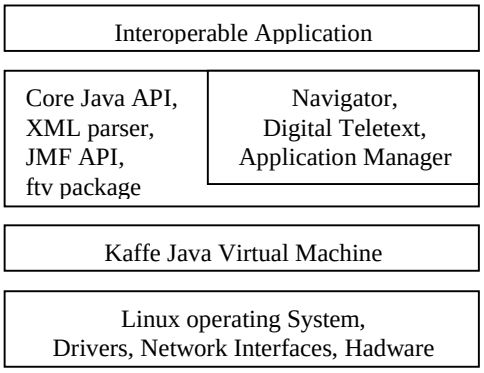


Figure 23. Environment used for the implementation [67 and 68].

5.2.2 Test

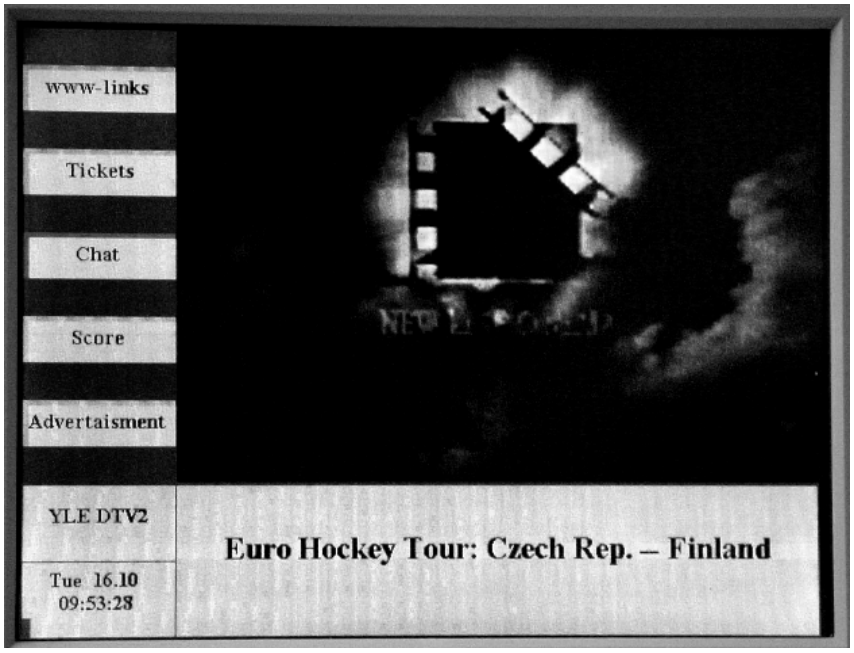
In the environment, *ftv* package was included as the GUI framework at the API level. Hence, the applications had to be modified to make use of it. Once the new environment was running, some storage memory results were taken.

For test purposes, it was decided to modify the Interoperable Application. In addition, minimal changes to the Application Manager had to be done, since it provided the container for the applications. It worked in the following way, Application Manager started the Interoperable Application, when the user pressed a specific key.

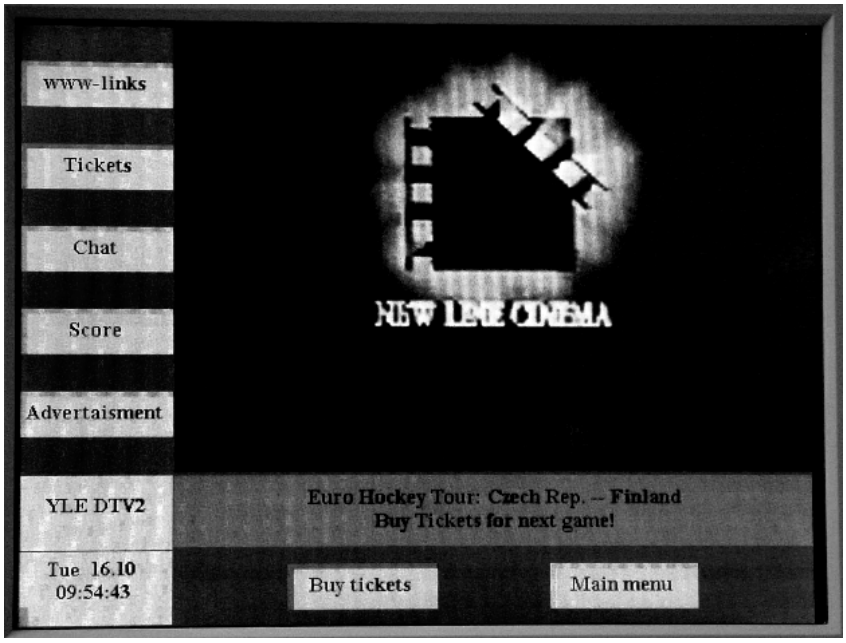
As mentioned is sub-section 5.1.1, the container in *ftv* package was *ftv.FTVScene*, so it was used as the new container. It handled Window events, and also, *ftv* widgets could be added to it. Then, the Interoperable Application was re-implemented, maintaining the functionality, making use of *ftv* package.

The application layout was divided into four differentiate regions. In the top-left, a toggle group, which behaves as a checklist of five toggle buttons, was placed. These toggle buttons did not use the default look. Instead, *ftv.FTVGraphicTextLook* was associated to each. The user could select the service desired via them. In the top-right region, the video was placed. In the bottom-left region, there were two widgets. On the top, a static text indicated the current channel. At the bottom, there was a static time widget. Finally, the bottom-left part informed of the current program by a static text. A screen-shot of the main screen of this application is shown in picture 5.

Top-left toggle buttons controlled the services available. When the user selected one of them, the bottom-left part of the screen was replaced to offer different options. Graphic buttons, static text, toggle groups, and static animated text were used for this purpose. Picture 6. shows the “Buy tickets” service, as an example.



Picture 5. Main screen of the Interactive Application using *ftv* package.



Picture 6. "Buy tickets" service of the Interactive Application using *ftv* package.

5.2.3 Results

Once the environment was running, some results were taken. The most important result was the ease of developing an application, when a GUI framework is available. In addition, it has to be remarked the importance of having digital television oriented widgets for GUI purposes.

In section 5, the possible options a developer has when developing applications without a specific GUI framework were introduced. On one hand, one can develop application specific widgets. On the other hand, one can use a general-purpose framework. The two next paragraphs will study the benefits of using a GUI framework oriented to the device at hand, in this case digital television.

Firstly, the *ftv* package has advantages over a general purpose GUI framework implementation. The storage memory is minimised, also the widgets are platform specific, in this case, television oriented. Table 2 shows a comparison of storage memory benefits, the packages where compressed using the jar functionality. In the environment, the application manager used a container from *Swing*, which is not needed anymore.

Package	Storage Memory (KB)
Swing.jar	2420
ftv.jar	96

Table 2. Storage memory comparison between *ftv* and *Swing* package.

Secondly, *ftv* has, also, benefits over application specific widget implementation. The most obvious is that the widgets can be reused for other applications. In addition, applications are easier to implement, since they do not have to create their own widgets. When a specific look is needed, it can be easily created and associated to an existing widget. Finally, the storage memory of the application is minimised, since its set of widgets it is not included. The interactive application used its own widgets. Table 3 compares the storage memory before and after the use of *ftv* package.

Interactive Application	Storage Memory (KB)
Before using *ftv*	77
After using *ftv*	26

Table 3. Storage memory comparison of Interactive Application before and after using *ftv* package.

6. CONCLUSIONS

Television will not be the same after the digitalisation comes about. All the population is used to analogue television, so it is a responsibility for everyone involved in this field, to make the shift as smooth as possible. One essential part of digital television services is the GUI. It acquires special importance, since it is the means the user has to interact with the system.

A TV-friendly, user oriented GUI is necessary for digital television services. For this reason, it is beneficial to provide the developer of applications a GUI framework. This way, a developer does not have to deal with the underlying system. In addition, the visual presentation and the behaviour of the widgets does not depend on the system, but can be decided depending on the needs.

The objective of this thesis was to implement a digital television oriented GUI framework. It was called *ftv* package. Since DVB-MHP selected HAVi, *ftv* followed this specification. *ftv* separates look and feel, it defines new events, and it is a lightweight framework based on *java.awt* package. In addition, a running digital television environment was available, which lacked of a digital television oriented GUI framework. Hence, it was used for test purposes.

There are many benefits of using a specific GUI framework. The first one is the availability of TV-friendly widgets. That is, the widgets are digital television oriented. Then, the development of the applications' GUI is easy, since the widgets are already developed. In addition, different visual presentations of the widgets can be used depending on the needs. Also, the storage memory needed is minimised from a general-purpose GUI framework, because unused classes are avoided.

Digital television is one of the new devices, which can be used for media rich applications. Other examples are mobile phone and PDA. Usually, a general-purpose GUI framework, such as Swing, is used when developing GUI for these devices. This approach has two problems. First, some unnecessary classes, which can be avoided, are included in the framework. Second, general look and feels instead of device specific ones are provided. Here, a device-oriented GUI framework implementation concept is proposed.

Finally, it has to be said that *ftv* can be modified having in mind other devices. Since the look and feel is separated, the new look classes can be implemented for other devices. In addition, new device-oriented events can be created. So, each device can store the widgets, the events and the look classes needed. The final objective is to define a unified GUI framework for all kind of devices, which will be the topic of future studies.

REFERENCES

[1] Morton D., "The Electrical Century, Viewing Television's History," *Proceeding of IEEE*, Volume: 88, Issue: 7, July 1999.

[2] Digital Television Group (DTG), "General Situation, Finland," *DTG Reference Library*, referred on 22.1.2001, available at http://www.dtg.org.uk/reference/index.html.

[3] Koistila P., "Content Production in Digital Environment, Master Thesis," *Helsinki Univerity of Technology*, December 2000.

[4] 625: Andrew Wiseman's Television Room "Digital TV – Beyond the Hype," referred on 22.1.2001 available at http://625.uk.com/digital/index.htm#whatitis

[5] Nave R., and Tsuria Y., "Advanced Use of TV Set Top Boxes," *International Broadcasting Convention*, pages: 436-440, 1997.

[6] Anastassiou D., "Digital Television," Proceedings of the IEEE, Volume: 82, Number: 4, pages: 510-519, April 1994.

[7] Fox B., "Digital TV Comes Down to Earth," *IEEE Spectrum*, October 1998.

[8] Fox B., "Digital TV Rollout," *IEEE Colloquium*, pages: 12/1-12/2, 1999.

[9] Ciciora W. S., "Inside the Set-Top Box," *IEEE Spectrum*, Volume: 33 Issue: 4, pages: 70-75, April 1995.

[10] Datta T., "Digital Convergence: The Set Top Box and Its Digital Interface," *White paper, Enthink*, July 1998.

[11] Hurley T., "Evolution of the Digital Set Top Box," *International Broadcasting Convention*, (conf. Publ N. 428) pages: 277-282, 1996.

[12] Droitcourt J.-L., "Integra Architecture- Anatomy of the Interactive Television Set-Top Box, How it Works, and What it Means to the Consumer," *International Broadcasting Convention* (conf. Publ N 428) pages: 272–276, 1996

[13] NorDig information central, "Nordig II Version 0.9," *referred on 22.1.2001* available at http://www.nordig.org/

[14] Stilgic J., "DVB Selects Java Technology For Its Digital Television Broadcasting Standard," *Java Press Release*, November 1999.

[15] Jacklin M., "The Multimedia Home Platform-On the Critical Path to Convergence," *IBC Show Daily*, September 1997.

[16] Evain J. P., "The Multimedia Home Platform. An overview," *EBU Technical Review*, spring 1998.

[17] Tan Q., Zhou M., Li J., and Yao D., "A Brief Overview of Current TV Set-Top Box Developments," *Systems, Man and Cybernetics, 1996, IEEE International Conference on*, Volume: 3, pages: 2127-2132, 1996.

[18] Vuorimaa P., "Digital Television Service Architecture," *Conference on Multimedia and Exposition, ICME2000, New York City, NY, USA*, July 30-Aug. 2, pages: 1411-1414, 2000.

[19] Peng C., and Vuorimaa P., "A Digital Television Navigator," *Proc. the Internet and Multimedia Systems and Applications, IMSA'2000, Las Vegas, USA*, Nov. 19-23, pages: 69-74, 2000.

[20] Pullinen M., "Billing in Digital Television, Master Thesis," *Helsinki Univerity of Technology*, March 2000.

[21] ETSI ETS 300 468, "Specification for Service Information (SI) in DVB Systems," *European Telecommunications Standards Institute*, January 1997.

[22] ETSI ETS 300 707, "Protocol for a TV-Guide Using Electronic Data Transmission," *European Telecommunications Standards Institute*, 1997.

[23] Tarrant D. R., "An European Standard for an Electronic Programme Guide," *International Broadcasting Convention*, pages: 447–455, 1997.

[24] Peng C., and Vuorimaa P., "Digital Teletext Browser," *the 9th Int. Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, WSCG'2001, Czech Republic*, February 5-9, pages: 120-125, 2001.

[25] Vuorimaa P., and Sancho C., "XML Based Text TV," *Web Information Systems Engineering, WISE2000, Hong Kong*, June 19-20, pages: 109-115, 2000.

[26] ETSI ETS 300 706, "Enhanced Teletext Specification" *European Telecommunications Standards Institute*, 1997.

[27] Vierinen J., and Vuorimaa P., "A Browser User Interface for Digital Television*," the 9th Int. Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, WSCG'2001, Czech Republic*, February 5-9, pages: 174-181, 2001.

[28] Kraal C. J., "Developing for Interactive Services' in digital TV," *International Broadcasting Convention*, (Conf. Publ. N. 428) pages: 230-235, 1996.

[29] Eronen L., and Vuorimaa P., "User Interfaces for Digital Television: a Navigator case study," *the 5th International Working Conference on Advanced Visual Interfaces, AVI2000, Palermo, Italy*, May 24-26, pages: 276-279, 2000.

[30] Vuorimaa P., Ropponen T., and von Knorring N., "X-Smiles XML browser," *the 2nd International Workshop on Networked Appliances, IWNA'2000, New Brunswick, NJ, USA*, Nov. 30-Dec. 1, 2000.

[31] Droitcourt J.-L., "Understanding How Interactive Television Set Top Box works…and what it will mean to the consumer," *International Broadcasting Convention*, pages: 382–394, 1995

[32] Thomson D., "IEEE 1394: Changing the Way We Do Multimedia Communications," *IEEE Multimedia*, Volume: 7, Issue: 2, pages: 94-100, April-June 2000.

[33] Lea R., Gibbs S., Dara-Abrams A. and Eytchison E., "Networking Home Entertainment Devices with HAVi," *Computer*, Volume: 33 Issue: 7, pages: 35–43, July 2000.

[34] GMD Fokus, "R & D – Competence – Centers – MAGIC – HAVI," *GMD Fokus*, referred on 14.3.2001, available at http://www.fokus.gmd.de/research/cc/magic/projects/havi/CONTENT-HTML

[35] HAVi Organisation, "HAVi-Home Audio/Video Interoperability," referred on 14.3.2001, available at http://www.havi.org/

[36] ISO Organisation, "Introduction to ISO," referred on 28.3.2001, available at http://www.iso.ch/infoe/intro.htm

[37] MPEG Group, "MPEG Home Page," referred on 28.3.2001, available at http://www.cselt.it/mpeg/

[38] DAVIC Association, "DAVIC: What is DAVIC," referred on 28.3.2001, available at http://www.davic.org/WHATIS.HTM

[39] NorDig Organisation, "NorDig.org Questions and Answers," referred on 28.3.2001, availbale at http://www.nordig.org/QandA.html

[40] DigiTAG Group, "DigiTAG Company Profile," referred on 28.3.2001, available at http://www.digitag.org/

[41] Digital TV Group, "Welcome to Digital Television," referred on 28.3.2001, available athttp://www.dtg.org.uk/welcome/index.html

[42] DVB, "About DVB-Background," referred on 30.3.2001, available at http://www.dvb.org/about/about.html

[43] Fox B., "The Digital Dawn in Europe," *IEEE Spectrum*, Volume: 32, Issue: 4, pages: 50-53, April 1995.

[44] Wood D., "The DVB Project: philosophy and core system," *Electronics and Communication Engineering Journal*, Volume: 9, Issue: 1, pages: 5-10, February 1997.

[45] Piesing J., "The DVB Multimedia Home Platform-MHP," *Interactive Television IEEE Hong Kong*, (Ref. N. 1999/2000), pages: 149-152, 1997.

[46] DVB Project Office, "DVB: Multimedia Home Platform," Revision 15, February 2000.

[47] MHP, "DVB-MHP-What is MHP?," referred on 30.3.2001, available at http://www.mhp.org/what_is_mhp/what_is_mhp.html

[48] Grand Alliance HDTV, "Grand Alliance HDTV System Specification," *version 2.0*, December 7, 1994

[49] Uehara M., Takada M., and Kuroda T., "Transmission Scheme for the Terrestrial ISDB System," *IEEE Transactions on Consumer Electronics*, Volume: 45, Issue: 1, pages: 101-106, February 1999.

[50] Donnelly A., and Smythe C., " A Tutorial on the Digital Audio-Visual Council (DAVIC) Standardisation Activity," *Electronics and Communication Engineering Journal*, Volume: 9, Issue: 1, pages: 46-56, February 1997.

[51] Preece J., Rogers Y., Sharp H., Benyon D., Holland S., and Carey T., "Human-Computer Interaction," *The Open University*, 1994.

[52] Eckstein R., Loy M., and Wood D.: Java™ Swing, *O'Reilly Associates*, 1998.

[53] Tai L.-C., "The GUI Toolkit, Framework Page," referred on 24.10.2001, available at http://www.free-soft.org/guitool/

[54] Booth P. A., "An Introduction to Human-Computer Interaction," *Lawrence Erlbaum Associates Ltd*, 1989.

[55] Shneiderman B., "Designing the User Interface: strategies for effective human-computer-interaction," *Addison Wesley Longman*, 1998.

[56] Olsen D. R., " Developing User Interfaces," *Morgan Kaufman Publishers*, 1998.

[57] Flanagan D., "Java in a Nutshell," *O' Reilly & Associates, Inc*, 1996.

[58] Lindholm T., and Yellin F., "The Java™ Virtual Machine Specification," *Sun Microsystems, Inc*, 1997.

[59] Fowler A., "Mixing Heavy and Light Components," referred on 31.10.2001, available at http://java.sun.com/products/jfc/tsc/articles/mixing/

[60] Borland Developer Support Staff, "AWT vs Swing," referred on 31.10.2001, available at http://community.borland.com/article/0,1410,26970,00.html

[61] Knudsen J., "Light as a Feather," referred on 31.10.2001, available at http://java.oreilly.com/bite-size/java_0499.html

[62] Fowler A., "Painting in AWT and Swing," referred on 31.10.2001, available at http://java.sun.com/products/jfc/tsc/articles/painting/

[63] Sundsted T., "Using the Graphics Class," referred on 31.10.2001, available at http://www.javaworld.com/javaworld/jw-11-1996/jw-11-howto_p.html

[64] Sun Microsystems, inc., "Java™ Foundation Classes: Now and the Future," referred on 1.11.2001, available at http://java.sun.com/products/jfc/whitepaper.html

[65] Sundsted T., "Java and Event Handling," referred on 1.11.2001, available at http://www.javaworld.com/javaworld/jw-08-1996/jw-08-event_p.html

[66] Retelewski T., "Create New Event Types in Java," referred on 1.11.2001, available at http://www.javaworld.com/javaworld/javatips/jw-javatip35.html

[67] Peng C., Cesar P., and Vuorimaa P., "Integration of Applications Into Digital Television Environment," accepted to the 2001 *Int. Workshop on Multimedia Technology, Architecture, and Application*, Taipei, Taiwan, Sept. 26-28, pages: 266-272, 2001

[68] Sivaraman G., Cesar P., and Vuorimaa P., "System software for digital television applications," accepted to *the IEEE Int. Conf. on Multimedia and Expo*, Tokyo, Japan, August 22-25, pages: 784-787, 2001.

[69] Transvirtual, “Wherever you Want to Run Java, Kaffe is There,” *White Paper*, 1999.