ISBN: 951-22-6171-5
Doctoral Dissertation

# Digital Television Applications

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission of Department of Computer Science and Engineering for public examination and debate in E-Hall, the Main Building of Helsinki University of Technology, Espoo, Finland, on the 15th of November, 2002, at 10 am o'clock.

Chengyuan Peng

Telecommunications Software and Multimedia Laboratory
Department of Computer Science and Engineering
Helsinki University of Technology
P.O. Box 5400, FIN-02015 HUT
Finland
Email: pcy@tml.hut.fi

Finland 2002

# ABSTRACT

Studying development of interactive services for digital television is a leading edge area of work as there is minimal research or precedent to guide their design. Published research is limited and therefore this thesis aims at establishing a set of computing methods using Java and XML technology for future set-top box interactive services. The main issues include middleware architecture, a Java user interface for digital television, content representation and return channel communications.

The middleware architecture used was made up of an Application Manager, Application Programming Interface (API), a Java Virtual Machine, etc., which were arranged in a layered model to ensure the interoperability. The application manager was designed to control the lifecycle of Xlets; manage set-top box resources and remote control keys and to adapt the graphical device environment. The architecture of both application manager and Xlet forms the basic framework for running multiple interactive services simultaneously in future set-top box designs.

User interface development is more complex for this type of platform (when compared to that for a desktop computer) as many constraints are set on the look and feel (e.g., TV-like and limited buttons). Various aspects of Java user interfaces were studied and my research in this area focused on creating a remote control event model and lightweight drawing components using the Java Abstract Window Toolkit (AWT) and Java Media Framework (JMF) together with Extensible Markup Language (XML).

Applications were designed aimed at studying the data structure and efficiency of the XML language to define interactive content. Content parsing was designed as a lightweight software module based around two parsers (i.e., SAX parsing and DOM parsing). The still content (i.e., text, images, and graphics) and dynamic content (i.e., hyperlinked text, animations, and forms) can then be modeled and processed efficiently.

This thesis also studies interactivity methods using Java APIs via a return channel. Various communication models are also discussed that meet the interactivity requirements for different interactive services. They include URL, Socket, Datagram, and SOAP models which applications can choose to use in order to establish a connection with the service or broadcaster in order to transfer data.

This thesis is presented in two parts: The first section gives a general summary of the research and acts as a complement to the second section, which contains a series of related publications.

**Keywords**: interactive service, digital television, middleware, user interface, content, interactivity, Java, XML.

# ACKNOWLEDGEMENT

It is not easy for a woman from a developing country to obtain a doctoral degree in computer science however, it had always been my dream. I worked as a software engineer for a company in China after graduating from Jilin University and following several years' programming, I began to feel that my work was becoming easy. I had become very interested in the challenge of carrying out research work and it was at this point that I decided to continue my post-graduate study within the Department of Computer Science and Engineering, Helsinki University of Technology, Finland.

I met with many difficulties at the beginning of my stay in Finland. In addition to the language barrier and money worries, the most difficult problems were studying and carrying out research under a different education system. When I began to doubt continuing my studies I attended a digital television seminar presented by Prof. Petri Vuorimaa. I was extremely interested in the topics discussed at the seminar and managed to secure an opportunity to work within the Future TV research group headed by Prof. Vuorimaa. It was under his guidance that I started my research into the development issues of digital television interactive services (i.e., my thesis).

I would like deeply to thank my supervisor Prof. Vuorimaa for his western style guidance, continuous support, and encouraging me to publishing research papers during my thesis work. Without his help, I would not have completed my thesis and achieved progress in all aspects of my research. In Feb. 2000, I published my first scientific paper under his encouragement and guidance. This was a very important first step for me towards the completion of my thesis and contributed to my knowledge in the area of digital television research. The most important gains for me have been obtaining the skills to carry out research i.e, learning to think, to discover, and to solve complex problems. All of these things are also valuable for my future career.

I would like to take this opportunity to thank Prof. Martti Mäntylä for his guidance in basic scientific aspects at the beginning of my post-graduate study which were very useful and helpful to my future research direction. I am grateful to Prof. Olli Simula for his valuable guidance in my minor subject study (neural network in machine learning). I would also like to express my thanks to Nokia Oyj Foundation for their support during my post-graduate study (2000-2001).

This manuscript was pre-examined by Dr. Pauli Heikkilä from Digita Oy of Finland and Prof. Seppo Kalli from Tampere University of Technology, Finland. I would like to express my sincere thanks to them for their valuable comments and constructive suggestions which significantly improved my thesis. I also wish to express my appreciation to Dr. Tony Daniels from Zarlink Semiconductor, UK, who helped me with language errors and gave valuable comments from senior software specialist point of view.

I would like to thank all my colleagues from past Future TV research group, especially Petri Koistila, Juha Vierinen, Pablo Cesar, and Artur R. Lugmayr (Tampere University of Technology), for their corporation, kind help and advice. I would like to thank TML engineer Ilpo Lahtinen for his help in the care of computer support, etc. allowing me to complete my work without delay. I would like to thank Sanna Patana and Ansa Laakkonen for their help during my work in TML laboratory. I would also like to take this opportunity to thank all the teachers and assistants who taught me at Helsinki University of Technology.

Finally, I wish to express my gratitude to my husband Bin Cheng and my son Genghua Cheng for their understanding and support. They had no complaining of my spending numerous weekends in the office.

Chengyuan Peng

Otaniemi, Finland
The 1st of July, 2002

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ABBREVIATIONS

| | |
|---|---|
| **AAC** | Advanced Audio Coding |
| **AC** | Audio Compression |
| **AIT** | Application signaling Information Table |
| **API** | Application Programming Interface |
| **ATSC** | Advanced Television Systems Committee |
| **ATSC-C** | ATSC-Cable |
| **ATSC-T** | ATSC-Terrestrial |
| **AWT** | Abstract Window Toolkit |
| **BAT** | Bouquet Association Table |
| **BPSK** | Binary Phase Shift Keying |
| **CA** | Conditional Access |
| **CAT** | Conditional Access Table |
| **CATV** | Cable TV Distribution Systems |
| **COFDM** | Coded Orthogonal Frequency Division Multiplexing |
| **CPU** | Central Processing Unit |
| **CSA** | Common Scrambling Algorithm |
| **DC** | Direct Current |
| **DDI** | Data Driven Interaction |
| **DTD** | Document Type Definition |
| **DECT** | Digital Enhanced Cordless Telecommunications |
| **DQPSK** | Differential Quadrature Phase Shift Keying |
| **DOM** | Document Object Model |
| **DSM-CC** | Digital Storage Media - Command and Control |
| **DSM-CC-UU** | Digital Storage Media - Command and Control User to User |
| **DVB** | Digital Video Broadcasting |
| **DVB-C** | DVB-Cable System |
| **DVB-S** | DVB-Satellite System |
| **DVB-T** | DVB-Terrestrial System |
| **EIT** | Event Information Table |
| **EPG** | Electronic Program Guide |
| **IP** | Internet Protocol |
| **GPRS** | General Packet Radio Service |
| **GSM** | Global System for Mobile Communications |
| **GUI** | Graphical User Interface |
| **HAVi** | Home Audio/Video Interoperability |
| **HDTV** | High Definition Television |
| **HTML** | Hyper Text Mark-up Language |
| **HTTP** | Hyper Text Transport Protocol |
| **IAV** | Intermediate Audio/Video |
| **ISDB** | Integrated Services Digital Broadcasting |
| **ISDB-C** | ISDB-Cable |
| **ISDB-S** | ISDB-Satellite |
| **ISDB-T** | ISDB-Terrestrial |
| **ISDB-TSB** | ISDB-Terrestrial Sound Broadcasting |
| **ISDN** | Integrated Services Digital Network |

| | |
|---|---|
| **JDK** | Java Development Kit |
| **JMF** | Java Media Framework |
| **LMDS** | Local Multipoint Distribution System |
| **MHP** | Multimedia Home Platform |
| **MMDS** | Microwave Multipoint Distribution Services |
| **MP@HL** | Main Profile at High Level |
| **MP@ML** | Main Profile at Main Level |
| **MPEG** | Motion Picture Expert Group |
| **NIT** | Network Information Table |
| **OFDM** | Orthogonal Frequency Division Multiplexing |
| **OSD** | On Screen Display |
| **PAT** | Program Association Table |
| **PES** | Packetized Elementary Stream |
| **PID** | Packet Identification |
| **PMT** | Program Map Table |
| **PSI** | Program Specific Information |
| **PSK** | Phase Shift Keying |
| **PSTN** | Public Switched Telephone Network |
| **PVR** | Personal Video Recorder |
| **QAM** | Quadrature Amplitude Modulation |
| **QPSK** | Quadrature Phase Shift Keying |
| **RAM** | Random Access Memory |
| **RF** | Radio Frequency |
| **ROM** | Read-only Memory |
| **RST** | Running Status Table |
| **RTOS** | Real-time Operating System |
| **SAS** | Subscriber Authorization System |
| **SAX** | Simple API for XML |
| **SDT** | Service Description Table |
| **SDTV** | Standard Definition Television |
| **SI** | Service Information |
| **SMATV** | Satellite Master Antenna TV distribution systems |
| **SMS** | Subscriber Management System |
| **SOAP** | Simple Object Access Protocol |
| **ST** | Stuffing Table |
| **TCP** | Transmission Control Protocol |
| **TC8PSK** | Trellis-Coded 8 PSK |
| **TDT** | Time and Date Table |
| **TOD** | Time Offset Table |
| **UDP** | User Datagram Protocol |
| **UI** | User Interface |
| **URL** | Universal Resource Locator |
| **VOD** | Video On Demand |
| **VSB** | Vestigial Side Band Modulation |
| **8-VSB** | Vestigial Side Band Modulation with 8 discrete amplitude levels |
| **W3C** | World Wide Web Consortium |
| **WAP** | Wireless Application Protocol |
| **XML** | Extensible Markup Language |

# PART ONE: SUMMARY OF RESEARCH

## 1 INTRODUCTION

Most existing terrestrial television transmissions are broadcast as analogue signals where the signal quality can be reduced due to location, obstacles, or interference from other sources (such as overhead electric cables). To receive the best possible signal, a rooftop antenna is required. Satellite television uses an external dish to receive its data, however, these signals are affected by weather conditions and pollution in the earth's atmosphere. Although cable television suffers little of the signal loss experienced by terrestrial and satellite television services, existing cable television services are restricted in the number of channels they can offer.

Digital television will ultimately replace the existing analogue systems and bring far more than significantly improved video and audio signal quality to television viewers. Digital television allows much more information (i.e. channels) to be transmitted and uses a new broadcasting technology to transmit services in binary format. Each channel is compressed and converted into a digital data stream using the Moving Pictures Experts Group (MPEG-2) compression algorithms [1]. This type of digital compression packs at least five times as many channels into a given distribution network bandwidth. MPEG only transmits the parts of a picture that changes from one frame to the next, rather than sending a completely new frame, thus reducing the amount of data that needs to be sent in order to reconstruct the original picture. Because the space needed for a digital channel is less than that for an analogue channel several digital signals can be transmitted side-by-side in the space previously occupied by a single analogue channel. Atmospheric interference has little or no effect on a digital signal as digital television receives high quality signals as binary coded data at the receiver with little loss of information.

Digital television produces sharper images than traditional analogue television and includes digital surround sound. Some service providers even have High Definition Television (HDTV) (depending on the standard adopted) and wide-screen programs. However, potentially the most interesting and exciting feature is that digital transmission creates the potential for interactive services. In combination with a return channel, digital television will be able to offer viewers a variety of enhanced and interactive services, from interactive soap operas to a high speed Internet over the air by combining TV with the Internet. Possible services include: an electronic program guide (EPG), video-on-demand (VOD), personal video recorder (PVR), pay-per-view, multi-camera-angle sporting events, home billing, home shopping, games, TV chat, digital Teletext, digital subtitles, etc.

A moving receiver cannot receive analogue television signals [2] however, with digital television moving receivers (i.e. located in cars, buses, trams, trains and even hand-held television sets) can receive clear terrestrial digital television signals and allow their viewers to make use of new interactive services. Also, digital technology and the convergence of various digital media will introduce many more possibilities, opportunities and challenges than today’s analogue television [3].

## 1.1 DIGITAL TELEVISION STANDARDS

Several different digital television standards are emerging from different world regions. The three main standards bodies include Digital Video Broadcasting (DVB), Advanced Television Systems Committee (ATSC), and Integrated Services Digital Broadcasting (ISDB). Table 1 presents a summary of the key parameters from the three resulting digital television standards. All proposed digital television systems use MPEG-2 technology for video and audio coding and for multiplexing to achieve an adequate throughput of the vast amounts of data required by HDTV or Standard Definition Television (SDTV).

| Standard | System type | Video coding | Audio coding | Modulation scheme | Channel bandwidth | Bit rate (Mbps) | Adopted countries |
|---|---|---|---|---|---|---|---|
| DVB | DVB-S | MPEG-2 | MPEG-2/1 digital sound | QPSK | 8 MHz | 38 | All European countries, Australia, New Zealand, Russia, etc. |
| | DVB-T | | | QPSK/QAM/OFDM | | 24<br>15 (Mobile) | |
| | DVB-C | | | QAM | | 38 | |
| ATSC | ATSC-T | MPEG-2 | AC-3 | 8 VSB | 6 MHz | 19.28 | North America, South Korea, Taiwan, Mexico, Argentina, etc. |
| | ATSC-C | | | | | 38.57 | |
| ISDB | ISDB-S | MPEG-2 | MPEG-2 AAC | TC8PSK/QPSK/BPSK | 34.5 MHz | 52 | Japan |
| | ISDB-T | | | DQPSK/QAM | 5.6 MHz | 21.47<br>4.06 (Mobile) | |
| | ISDB-C | | | 64QAM | 6 MHz | 31.644 | |

**Table 1. Comparison of parameters in different standards.**

All European countries have agreed to adopt the DVB standard [4] as DVB is one of the leading standard bodies in digital television. It has defined a satellite transmission standard (DVB-S), which is used by several satellite operators around the world. The DVB has also defined cable (DVB-C), terrestrial broadcast services (DVB-T) and Multimedia Home Platform (MHP) standards for receivers.

The DVB is based on SDTV and employs the MPEG-2 video compression and MPEG-2 or MPEG-1 digital sound [5]. Only stereo sound will be transmitted initially however, at a later stage the system can be upgraded to multi-channel surround sound [6]. Available screen aspect ratios include 4:3, 16:9 (wide-screen), and 2.21:1 [7] (HDTV mode is optional). The DVB provides no direct compatibility between HDTV and STDV modes, which means that if HDTV transmissions are broadcast, they cannot be received on standard receivers.

High-definition pictures are to be simulcast alongside standard-definition pictures and future receivers will convert interlaced transmissions into a 625, or 1250, progressive format. The screen luminance resolution is 1920 x 1080 for both 25 Hz and 30 Hz HDTV. The screen resolution mode at a frequency of 30 Hz SDTV ranges from 720 x 480, 640 x 480, 544 x 480, 480 x 480, 352 x 480 to 352 x 240. The screen luminance resolution modes at a frequency of 25 Hz SDTV have 720 x 576, 544 x 576, 480 x 576, 352 x 576 to 352 x 288 [7]. In addition, DVB sets a common standard for encryption but broadcasters are free to use a conditional access system of their own choice to control de-encryption in response to payment [6].

The ATSC standard has been universally adopted in North America [4] [8]. ATSC is a U.S. organization that defines standards for terrestrial digital broadcasting and cable distribution

[9]. The ATSC standard is based on computer display standards that call for pictures to be transmitted at a rate of 24, 30, or 60 Hz to match cinema projection standards (24 frames per second) and 60 field /30 frame National Television Standards Committee (NTSC) analogue TV system. The Digital Audio Compression (AC-3) standard was selected for the audio source encoding and the MPEG-2 standard for the video encoding. ATSC includes digital HDTV, SDTV, data broadcasting, multi-channel surround-sound audio, and satellite direct-to-home broadcasting components [10]. ATSC resolution ranges from 1920 x 1080, 1280 x 720, 704 x 480 to 640 x 480 with screen aspect ratios that include 16:9 (wide-screen) and 4:3 modes [11].

The ISDB standard has been adopted as Japan's own unique national standard [4]. The technical standards for digital broadcasting in Japan are grouped as the ISDB Family. It consists of ISDB-S (Satellite) for satellite broadcasting, ISDB-C (Cable) for cable TV networks, ISDB-T (Terrestrial) and ISDB-TSB (Terrestrial Sound Broadcasting) for terrestrial broadcasting [12] [13] [14]. They were developed as the Japanese specification in order to provide flexibility, expandability, and commonality between multimedia broadcasting services using each network. The ISDB standard combines the functionality of a personal computer and video recorder and allows the inclusion of SDTV and HDTV. This standard also uses MPEG-2 for video and audio coding as well as data multiplexing. The MPEG-2 AAC (Advanced Audio Coding) is employed as the audio coding system. ISDB employs MPEG-2 MP@HL (Main profile at high level) for 1080i and 720p, MP@H14 for 480p, and MP@ML (Main profile at main level) for 480i, respectively [12].

## 1.2 DVB DIGITAL BROADCASTING SYSTEM

In order to fully understand how interactive services work in the digital television environment, the following subsections introduce the main components of a typical DVB digital broadcasting system: broadcaster head-end system, receiver, return channel, DVB data broadcasting, and Conditional Access (CA) systems.

### 1.2.1 Broadcaster Head-End System

Figure 1 shows a general diagram of the broadcaster side in a DVB digital broadcasting system from a signal flow point of view [15]. The broadcaster is responsible for producing MPEG-2 transport streams that contain several television programs. These transport streams need to be delivered error-free to the transmitters (i.e., terrestrial towers) or Microwave Multipoint Distribution Services (MMDS); satellite up-links and cable head-ends [16] [17] [18]. These transport streams have no protection against error, which can cause serious effects [19], therefore error correction and channel coding (or modulation) is necessary before the signals are passed to the transmitters.

Video and audio encoders are responsible for compressing and encoding the video and audio signals. One or more of the following are fed into the MPEG-2 multiplexer: video and audio elementary streams generated from these encoders; private data; service information (SI); conditional access (CA) control and synchronization information. Within the MPEG-2 multiplexer an initial packetization of the video and audio elementary streams is performed to produce Packetized Elementary Stream (PES) packets. Then, transport packets are generated from the PES packets and finally a transport stream is formed by multiplexing the

transport packets with additional data from DVB-SI, CA control, and synchronization information [20].

Private data is a data stream, whose content is not specified by MPEG, which may be used to carry digital Teletext; program subtitles; data; additional service information specific to a particular network and commands intended to control modulation and network distribution equipment, etc. [20].

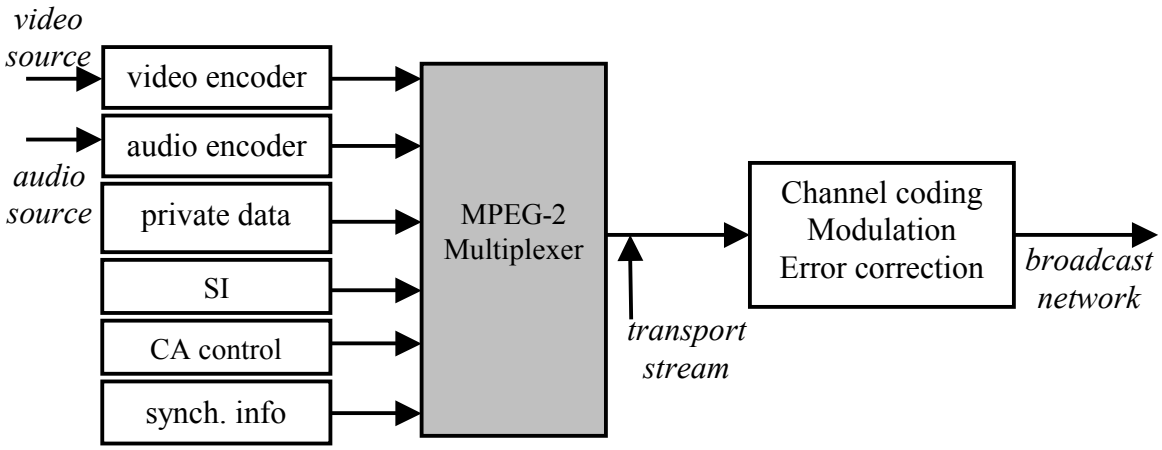


**Figure 1. Main components of broadcaster high-end system.**

Service information is also added to the broadcast stream to allow program tuning and selection and, once acquired, is stored in the SI database. DVB-SI includes four Program Specific Information (PSI) tables: Program Association Table (PAT); Program Map Table (PMT); Network Information Table (NIT) and Conditional Access Table (CAT). It also contains seven additional tables: Bouquet Association Table (BAT); Service Description Table (SDT); Event Information Table (EIT); Running Status Table (RST); Time and Date Table (TDT); Time Offset Table (TOT) and Stuffing Table (ST) [21]. PSI tables provide information to facilitate automatic configuration of the receiver; enable demultiplexing and decode the various program streams within the multiplex. The additional tables provide identification of services and events carried in different multiplexes [22]. SI data is conveyed as packets that have unique PIDs and these packets must be included periodically in every transport stream. The PAT always has a PID of 0, and the CAT a PID of 1. The demultiplexer must determine all of the remaining PIDs by assessing the appropriate table.

Synchronization of the decoding and presentation processes for audio and video at the receiver is a particularly important aspect of a real time, software based, multiplexer [23]. Consequently, a system of time stamps is specified to ensure that related elementary streams are replayed with the correct synchronization at the decoder. CA support is provided for the control of scrambling (i.e., for conditional access), which may be applied to one or more of the elementary streams [20].

The transport stream from the MPEG-2 Multiplexer is raw-serial binary data and is unsuitable for transmission for a variety of reasons. For example, runs of identical bits cause Direct Current (DC) offsets and lack a bit clock [24] meaning that there is no control of the

spectrum and that the bandwidth required is too great. Therefore, channel coding and modulation techniques are required. The purpose of channel coding is to increase the robustness and reliability of the digital information that is being transmitted over a noisy channel. In order to further increase robustness appropriate error correction is added to the transmitted data. The objective of modulation is to shift the message signal to the appropriate frequency therefore increasing the data transfer rate and making it suitable for transmission.

### 1.2.2 Receiver

Figure 2 shows a signal flow diagram for the main building blocks in a set-top box receiver. The Radio Frequency (RF) interface is connected to the incoming modulated signal. The tuner/demodulator block performs channel (frequency) selection, demodulation and error correction of the incoming MPEG-2 signal [25]. The tuner module is capable for accessing Quadrature Amplitude Modulation (QAM), Orthogonal Frequency Division Multiplex (OFDM), and Quaternary Phase Shift Keying (QPSK) network modulated data [19]. The baseband output signal from the tuner is forwarded to the demodulator whose function is to sample the analogue signal and convert it to a digital bit-stream. Once the bit-stream has been recovered it is checked for errors and forwarded to the demultiplexer and the output from the tuner/demodulator block is an MPEG-2 transport stream. The set-top box uses the return channel modulator to connect to interactive service providers.

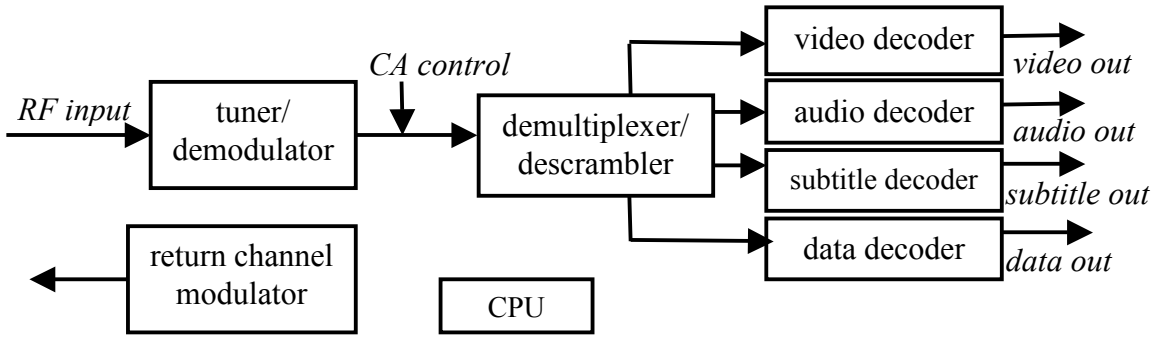


**Figure 2. A flow diagram of set-top box.**

The demultiplexer block synchronizes with the transport stream coming from the tuner/demodulator (or CA module) and selects the appropriate audio, video and/or private data elementary streams, according to the service selections made by the viewer. The demultiplexer block also contains circuits for descrambling of services subject to CA data in conjunction with a smart card [25]. The CA module is an external plug-in CA, which is attached via the Common Interface.

MPEG-2 uses an identifier (i.e., PID) to distinguish a packet as containing a particular format (i.e., audio, video, or data). The audio and video decoders complete processes for presentation (e.g., de-packetization, decompression, synchronization with related services, etc.). The subtitle decoder is used to decode and present the program subtitle data using the On Screen Display (OSD) buffer [26], while the data decoder is responsible for decoding system or private data (e.g., digital Teletext, SI, etc.). Subtitle and data decoders can be separate hardware modules or software.

The Central Processing Unit (CPU) is a microprocessor and the key system component in a set-top box. It manages all the internal units and attached external plug-in units [25] and its functions include initializing various hardware components; processing a range of Internet and interactive services; monitoring and managing hardware interrupts; fetching data and instructions from memory and running related programs, etc. [19].

### 1.2.3 DVB Data Broadcasting

Data broadcasting makes it possible to embed application data into the broadcast audio and video streams. The DVB data broadcasting system provides a means of delivering MPEG-2 transport streams via a variety of transmission media [27]. In addition, the DVB has extended the MPEG-2 systems specification (ISO/IEC 13818-1) to produce a full DVB-SI specification. Examples of data broadcasting are the download of software over satellite, cable or terrestrial links; the delivery of Internet services over broadcast channels and interactive TV, etc. Five different application areas with different requirements for the data transport have been identified, which include: Data Piping; Data Streaming; Multiprotocol Encapsulation; Data Carousels and Object Carousels [27]. Data Broadcasting will be a key technology for digital television applications.

Data Piping supports data broadcast services that require a simple, asynchronous, end-to-end delivery of data through the DVB compliant broadcast networks. Data broadcast is carried directly in the payloads of MPEG-2 transport stream packets [27].

Data Streaming supports data broadcast services requiring a stream-oriented, end-to-end delivery of data in either an asynchronous, synchronous, or synchronized way through the DVB compliant broadcast networks. Data broadcast is carried in PES packets, which are defined in MPEG-2 specification. Asynchronous data streaming is defined as the streaming of data without any timing requirements (e.g., RS-232 data). Synchronous data streaming is defined as the streaming of data with timing requirements in the sense that the data and clock can be regenerated at the receiver to provide a synchronous data stream. Synchronized data streaming is defined as the streaming of data with timing requirements in the sense that the data within the stream can be played back in synchronization with other types of data streams (e.g., audio and video) [27].

Multiprotocol encapsulation supports data broadcast services that require the transmission of datagrams of communication protocols via the DVB compliant broadcast networks. The transmission of datagrams is done by encapsulating the datagrams in DSM-CC data carousel specification (DSM-CC) sections (cf. figure 5), which are compliant with the MPEG-2 private section format [27].

The DVB data carousel is based on the DSM-CC data carousel specification (ISO/IEC 13818-6) and supports data broadcast services that require the periodic transmission of data modules through DVB compliant broadcast networks (cf. figure 5). The data transmitted within the data carousel is organized into modules, which are subdivided into blocks. Each block of all modules within the data carousel are of the same size, except for the last block of each module, which may be smaller. Modules are a delineation of logically separate groups of data within the data carousel, which can be clustered into a group of modules if required by the service. Likewise, groups can in turn be clustered into super groups [27].

The DVB Object Carousel supports data broadcast services that require the periodic broadcasting of DSM-CC User-User (U-U) objects through the DVB compliant broadcast networks. The DSM-CC is an extensive toolkit for handling the transfer of multimedia content. The Object Carousel has been selected to provide a broadcast filing system where a range of data content is broadcast cyclically, with the opportunity to update, add, or remove content as and when required. Data broadcast is transmitted according to the DSM-CC Object Carousel and DSM-CC Data Carousel specifications [19] (cf. figure 5).

### 1.2.4 Return Channel

Adding interactivity to the DVB broadcasting environment requires implementing a return channel as well as broadcast channel (cf. figure 3). A return channel is established between the viewer and the service provider and can be used to carry the viewer's commands and responses back to or download content from the service provider. Broadcast only services are sent via a downstream channel from the broadcaster to the receiver (cf. figure 3).

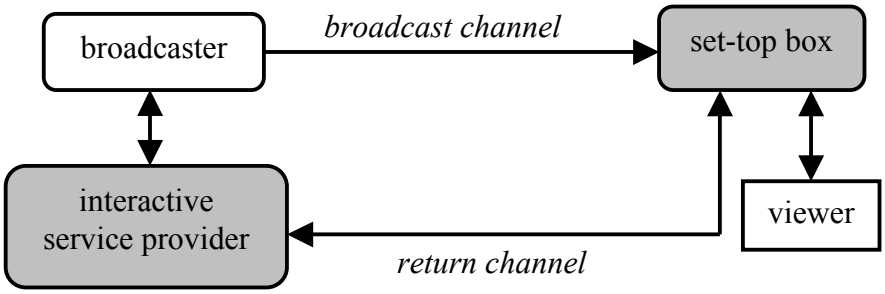


**Figure 3. A general model for interactive system.**

The DVB has defined a set of Network Independent Protocols (DVB-NIP) and a series of medium specific return channel specifications (i.e., network dependent protocols), e.g., DVB-RCC, DVB-RCCS, DVB-RCD, DVB-RCG, DVB-RCL, DVB-RCP, DVB-RCS respectively for CATV, SMATV, DECT, GSM, LMDS, PSTN/ISDN and satellite networks [28] [29] [30] [31] [32] [33] [34] [35]. Appendix A lists the abbreviations of return channels. The network dependent protocols work together with the Network Independent Protocols and the protocols defined in these standards provide a generic solution for a variety of broadcast only and interactive services through the use of DVB data broadcasting profiles [36] (cf. chapter 1.2.3).

The PSTN/ISDN is the most widely used technology for a return channel and is usually implemented via a narrowband communication link, such as PSTN/ISDN with a low-bit-rate of up to approximately 150 kbps [37]. All digital television receivers with interactive service capabilities will have either a built-in telephone modem; a port for a digital subscriber line connection or direct-broadcast satellite receiver or an Ethernet jack for a cable modem or home network setup.

### 1.2.5 CA System

The main objective of the CA system is to control a subscriber's access to digital television pay-per-view services and secure the operators revenue streams. CA systems: limit access to subscribers who have valid contracts with a specific network operator can access a service as

well as allowing network operators to directly target programming, advertisements, etc. Restricting access to a particular service is accomplished by using cryptography, which protects the digital service by transforming the signal into an unreadable format (i.e., encryption). Once the signal is encrypted it can only be decrypted by a digital set-top box using a decryption key. Set-top boxes generally incorporate the necessary hardware and software subsystems to receive and decrypt these signals. These components comprise a de-encryption chip; a secure processor (e.g., smart card) and appropriate hardware drivers (e.g., Common Interface). The smart card contains the necessary keys needed to decrypt the encrypted services [19].

Conditional Access is not fully specified in the DVB but a series of tools have been defined. The key tool to the entire DVB CA package is the DVB Common Scrambling Algorithm (CSA) for secure scrambling of transport streams or PES's. There are two CA interoperability scenarios envisaged in the DVB: SimulCrypt and MultiCrypt [38] [39] [40].

SimulCrypt is a mechanism whereby a single transport stream can contain several CA systems. One way of providing viewers with access to programs is to enable different CA decoder populations (potentially with different CA systems installed) to receive and correctly decode the same video and audio streams. An alternative is when a contract or negotiations between different program providers is established, enabling viewers to use the specific CA system built into their set-top boxes, irrespective of the fact that these programs were scrambled under the control of one of several CA systems.

MultiCrypt revolves the specification of a Common Interface which, when installed in the set-top-box, permits the viewer to switch manually between CA systems. Therefore, when the viewer is presented with a CA system, which is not installed in his box, he or she simply switches cards.

A typical end-to-end CA system consists of several subsystems: the Subscriber Management System (SMS); Subscriber Authorization System (SAS) and receiver CA control system [19]. SMS is ultimately responsible for handling all customer data and sends requests to SAS, which encrypts the requests and delivers the code to receiver CA control system (e.g., smart card). The code includes messages, which enable the descrambler to make the program legible.

## 1.3 MULTIMEDIA HOME PLATFORM (MHP)

MHP is an open DVB standard, which defines a whole set of technologies to implement digital interactive multimedia services in the home. MHP includes the home terminal (e.g., set-top box, TV set, or PC), its peripherals and the in-home digital network [41], and covers three application areas (i.e., enhanced broadcasting, interactive broadcasting, and Internet access) [42]. The primary goal of the MHP is to enable the birth of horizontal markets for digital television and multimedia services where there is open competition between content providers, network operators or platform manufacturers at each level in the delivery chain. Another goal is to exploit the potential for convergence between broadcasting, the Internet and consumer electronics.

The architecture of the MHP is defined in terms of three layers: resources, system software and applications (cf. figure 4). Typical MHP resources are MPEG processing, I/O devices, CPU, memory and a graphics system. The system software uses the available resources in order to provide an abstract view of the platform to the applications, which are controlled by the application manager.

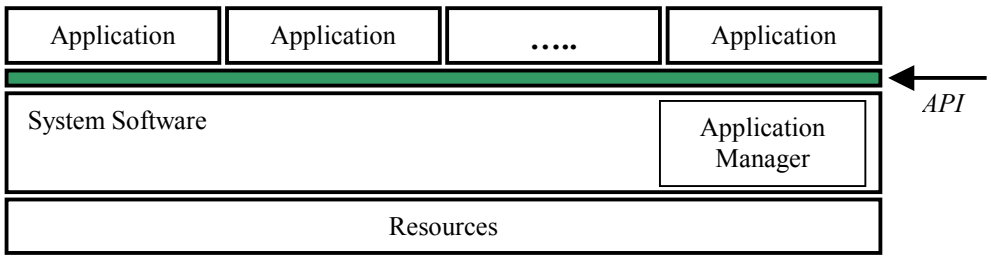


**Figure 4. Basic architecture of the MHP.**

The MHP defines a generic interface (i.e., API) between interactive digital applications and the terminals (cf. figure 4). The API provides an abstraction layer between different provider's applications and the specific hardware and software (i.e. device drivers) details of different MHP terminal implementations. MHP enables digital content providers to address all types of terminals ranging from low-end (e.g., small amount of RAM, no local storage capacity, limited computational and graphics capability) to high-end set top boxes, integrated digital TV sets, and multimedia PCs. Applications from various service providers will be interoperable with different MHP implementations in a horizontal market. Digital television applications use the APIs to access the actual resources of the receiver, including: databases, streamed media decoders, static content decoders and communications. Key applications are based around the DVB-Java platform and therefore the API plays a crucial role in MHP.

### 1.3.1 MHP in General

MHP defines transport protocols; application lifecycles and signaling models (both for DVB-Java and DVB-HTML applications); security models; content formats; the DVB-Java platform; plug-ins; the graphical reference model and minimum platform capabilities, etc.

The MHP transport protocols deal with the Network Independent Protocols so that the MHP unit can communicate through different network types. These protocols provide a generic solution for a variety of broadcast only and interactive services using DSM-CC User-to-User, data carousel and object carousel protocols (cf. chapter 1.2.3). They also support IP over the interaction channel as well as over the broadcast channel through the Multiprotocol Encapsulation (cf. chapter 1.2.3, figure 5, and figure 29) [42].

Security framework covers four security areas, i.e., authentication of applications; security policies for applications; authentication and privacy of the return channel communications and certificate management [42]. The security framework enables a receiver to authenticate an applications source and a typical system uses three different security messages (i.e., Cryptographic hash codes, Signatures and Certificates). Content formats include static and broadcast streaming formats (e.g., GIF, PNG, JPEG, MPEG-2 Video/Audio, subtitles, etc.); resident and downloadable fonts; color representation and MIME types.

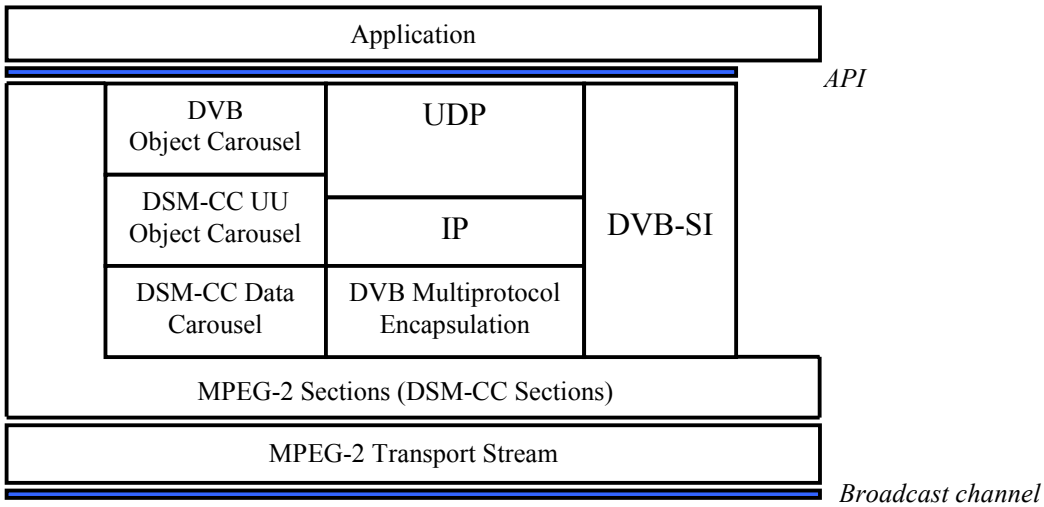


**Figure 5. Broadcast channel protocol stack.**

The DVB MHP specification provides the basic definitions needed for integration of DVB-HTML applications into the MHP unit. The DVB-HTML application and signaling models are defined but it is not the main MHP application. One reason is that HTML content developers have very limited control over how screen layout is rendered. In addition, local computation is not possible, making most types of interactivity impossible without a persistent two-way connection and a head-end infrastructure to support this interactivity. Games are exceptionally difficult to program due to the lack of an environment that can support local calculations. Although there is a wealth of content available that was authored using HTML based technologies on the Web, the vast majority of this is unsuitable for television use due to the unique display requirements of television [43].

A "plug-in" is a block of functionality that can be added to a generic platform in order to provide interpretation of application and content formats not specified by the MHP [42]. The choice of which plug-ins to use is totally in the hands of the end-user in order that he can have a choice of sources of service. Plug-in applications may stay resident, where the design of the platform implementation allows. There are two possible types of plug-in implementation, the first uses implementation-specific code (e.g., in native code or using implementation-specific Java APIs) and the other is described as an interoperable plug-in.

### 1.3.2 DVB-Java Platform

Java technology plays a vital role in creating and executing the new interactive multimedia services for set-top boxes. The core of the MHP is based on a platform known as the DVB-Java platform, which includes a virtual machine (as defined in the Java Virtual Machine specification from Sun Microsystems); a number of software packages to provide generic application program interfaces (APIs) to a wide range of features of the platform and a series of reusable and television-specific libraries (i.e., APIs). The latter are a subset of core class libraries from the Java platform and a set of libraries, which are an extension to Java platform (e.g., JavaTV, HAVi user interface, and DAVIC APIs). MHP applications access the underlying platform resources only via these specified APIs. Appendix B lists the APIs defined and supported in the MHP [42].

The Java virtual machine resides on the set-top box and executes Java bytecodes. Java bytecodes do not need to be fully resident, as they can be downloaded from the network, when required. When no longer needed, the java byte-code is automatically freed from the receiver's memory through the garbage collection mechanism provided by the Java virtual machine.

The DVB-Java platform is an ideal base for developing next generation interactive services as DVB-Java applications are inherently cross-platform portable. Application developers need to author the code only once and then it can run on any terminal regardless of the underlying operating system and CPU. The Java platform overcomes the limitations of HTML based technology [43], i.e., it enables advanced levels of interactivity, dynamic and broadcast-quality graphics as well as local computation. This provides an unlimited range of possibilities for content authors without requiring access to a return channel and broadcaster (service provider) head-end infrastructure. The Java platform is also among the most secure and reliable environments as it avoids the use of pointers, therefore eliminating many bugs and the risk of memory leaks.

## 1.4 DISCUSSION

Digital television will enhance the viewer's overall viewing experience by providing a wide range of interactive services. The key question is: Which digital television specific interactive services should be provided for the viewers? Essentially, there are two contrasting views on how interactive television should develop. One school of thought favors the walled garden approach, (i.e., never offering full Internet navigation) as they believe that people have PCs at home or the office for this purpose and that viewers requirements from interactive television services can be equated directly to leisure time. A second school favor the Web browser approach, (i.e., providing viewers with a seamless transition between television and full Internet browsing) as they believe that the walled garden approach will soon be overwhelmed by the Web's strength. The walled garden approach is fine if interactive television broadcasters want to attract people with no knowledge of the Web and I tend towards this belief. Ultimately, the approach adopted will affect design and development strategies of interactive services and system architecture of the set-top box. Chapter 2 and chapter 4 present discussions about the differences.

Digital television represents a fundamentally new technology compared to the computer. Interactive television services will create new opportunities and challenges for television content developers, advertisers, system engineers, and viewers. Specifically for application developers, in order to add more information, more control, more convenience and more fun to the television viewing experience, they must understand some of the differing characteristics, constraints and requirements between desktop computing and set-top boxes. Although a set-top box has many of the same components found in a desktop PC, e.g., a microprocessor, memory (some boxes even have built-in storage devices), an operating system, and other software, it has its distinctive features:

- Set-top box chip sets will handle audio, video, and data processing under limited memory and local data storage (e.g., hard drive).
- A set-top box has its own specialized Real-Time Operating System (RTOS) and the Java platform, which runs Java applications on top of this RTOS (in MHP).

- Applications code and data can be stored in flash memory chip even after viewers switch to a new channel or turn off the set. The manufacturer can update these embedded applications remotely.
- A TV tuner is built into a set-top box allowing it to access QAM-, OFDM-, and QPSK-based networks.
- Limited input devices (e.g., remote control).
- A return channel is needed, however it has limited bandwidth.
- Network communications protocols for both broadcasting and return channel are different.
- CA controls are necessary for accessing pay-per-view services.
- Usability issues are important.
- Digital television should provide television specific services and entertainment. The most important new functions should relate directly to TV watching.
- Design and implementation of interactive services should adapt to different platforms (e.g., PDA, mobile phone, laptop as well as set-top box).
- Utilizing the legacy services and content inherited from existing analogue TV.

In addition, developing digital television applications should meet some requirements. In general, including the following.

- Ubiquity of service
- Low cost
- Rich graphics and multimedia user experience
- Fast response
- More functionality and better quality in terms of services, connections and features
- Easy to use user interface (online help or extensive manuals are intolerant)
- Scalability, and interoperability and portability of the application source code
- Small footprints
- Reliability

## 1.5 RESEARCH PROBLEMS

Interactive services should facilitate interoperability and scalability requirements [44]. Interoperability means that the hardware, software, and interactive services from different manufactures, vendors, and service providers will run together in a set-top box. Portability means that an implementation should adapt to different set-top box platforms and some interactive services should be able to run in both low-end (demands small footprints) and high-end set-top box models. To achieve these goals, the system architecture of a set-top box must be well designed. Chapter 3 discusses this in further detail.

The biggest research problems are the presentation of a user interface as well as user interface design and usability issues. Many constrains are added on the look and feel of digital television interactive services and the interactive content must complement the large amounts of information and handle underlying video and background audio. In addition, the user input device is currently limited to a remote control. PC centric implementations have relied on bulky, memory-intensive graphics libraries that have added extra cost to the products therefore, some appropriate implementation methods are essential. They must be

studied and designed to solve these problems (cf. chapter 4). Also, adaptation problems, e.g., different screen size and aspect ratios, different platforms, etc. must be solved.

The content of information services (e.g., digital Teletext) must be well represented, organized and transmitted to ensure the interoperability and portability. No standard is available and key problems include the delay and the unpredictable order of the content arrivals [45]. Chapter 5 discusses this in greater detail.

Latency is one important aspect of measuring an interactive services' performance, because long delays are unacceptable to television viewers. Delays can occur in almost all the processes in digital broadcasting system, e.g., transmission, download, start-up, switching between services, page search, content rendering, network access, server response, etc. For example, in the broadcast environment lost packets have a noticeable effect because the client has to wait for the next broadcast of the missed packet or even the whole file. This can cause a significant delay before the content can be presented to the end-user. Consequently, application developers should consider latency measurement in their problem-solving methods.

It is also necessary and efficient to control the broadcast parameters to ensure optimum performance of the application for example, bandwidth (broadband or narrowband), bandwidth allocation (i.e., how much bandwidth is allocated for interactive services), connection methodology (connectionless or dial up), average access latency (tuned in or connect), scene load time, computing model (broadcast or client-server), service capacity (constrained or open-ended), server load, client caching (anticipatory or recently used), user interaction processing (set-top box or/and server), etc. [44].

## 1.6 SUMMARY

The objective of this chapter was to give an overview of DVB digital broadcasting system underpinning digital television applications and to outline the key research problems. This chapter introduced the benefits of digital broadcasting technologies versus the inefficiency of analogue television and three main digital television standards were compared in order to understand DVB standards. A digital television application touches almost the whole digital broadcasting system, therefore both broadcast head-end systems and receivers were introduced. MHP and its Java platform, return channel for interactive services and CA system were also described.

The main issues involved in development and deployment of digital television applications relate to: an applications user interface; application content; system architecture running the applications and communication protocols, etc. Chapter 2 will introduce the applications developed in the thesis and subsequent chapters (chapter 3, 4, 5, and 6) will summarize the methodology and solutions to the problems of digital television applications.

# 2 APPLICATIONS

The concept of an application in this thesis differs from that of services. The Java TV API characterizes television programs as services [46], which is an abstraction that provides a common way to refer to a wide variety of content that may appear in a broadcast environment. For example, a service can refer to a regular TV program with its synchronized audio and video or to an enhanced television broadcast that contains audio, video and a DVB-Java application that is synchronized with the broadcast. The application in this thesis means the DVB-Java application (i.e., Java program).

## 2.1 TYPES OF INTERACTIVE SERVICES

MHP supports many kinds of interactive services, which can be categorized as three types according to the level of interactivity, i.e., services with: local interactivity (broadcast-only); one-way interactivity and two-way interactivity.

Broadcast-only services support purely local interactivity, in which viewers can interact with an application running on the set-top box. No upstream communication occurs with the server. The code and data can be downloaded from broadcast network carried in transport stream multiplex and stored in the set-top box's Flash memory. Examples are EPG, digital Teletext, local games, VOD, PVR, etc. The local interactivity also includes tune to selection, browse, configure and control (e.g., subtitles), etc.

With one-way interactive services a physical upstream return channel is required when the viewer communicates with the server. These type of services only broadcast user responses and examples include: advertisement direct response, opinion polling and voting, etc.

Two-way interactive services can send user responses to the server and content can be correspondingly individually addressed to the user [44]. Examples include: email, Web browsing, TV banking, TV shopping, Interactive games, Interactive television (applications synchronized to TV content), education and interactive quiz shows (the viewer at home can compete along with the studio contestants), etc.

In the following subsections three representative applications developed for this thesis will be presented. Each represents one of the above types, i.e., Navigator, digital Teletext and an interactive sports program. DVB subtitles are also described although it is a TV program related service which: adopts new technology; has local interactivity and can be implemented in Java.

## 2.2 NAVIGATOR

Viewers will need help when they navigate the hundreds of channels that will exist in the digital television universe. The Electronic Program Guide (EPG) is the core digital television service who's primary function is to provide the viewer with an overview and schedule of current and upcoming television programs (transmitted in broadcast transport streams). With the help of an EPG it is possible for the viewer to select channels through, for example, a favorites list, an alphabetical list, a provider name list and a transponder list (i.e., channel lists by frequency). Usually, an EPG also changes the channel corresponding to the viewer's

selection. With this type of application, interactive performance and short start-up times are critical to a positive user experience.

Many broadcasters have launched or are planning digital television services using the DVB system. If EPG content is to be represented according to the format of DVB-SI it is feasible for viewers to download EPG software from any vendor having a newer and/or better EPG.

The EPG service developed for this thesis is called 'Navigator' and was designed as a resident application with local interactivity. No return channel is required and the content used was DVB-SI. The service information required is carried together with program MPEG-2 transport streams and downloaded from the data carousel via the broadcast channel in reply to viewer's request. Therefore, the server load on the broadcasters side does not depend on the number of active service users.


**Figure 6. Navigator main menu.**

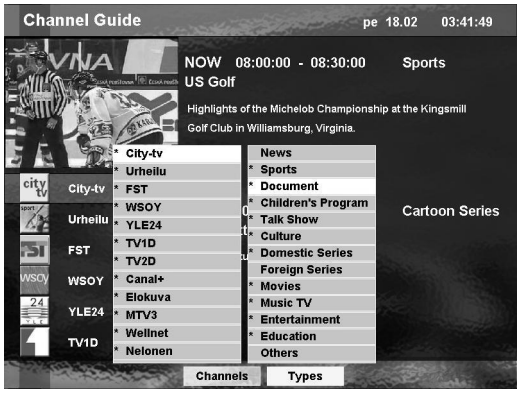


**Figure 7. Channel guide**.

The Navigator performs other functions as well as browsing television programs including: tuning (select) television programs, configuring the set-top box, downloading software, activating other services available in set-top box and controlling subtitles. Figure 6 shows the Navigator main menu user interface and Figure 7 shows a user interface for a typical channel guide. Figure 8 is the screenshot of the EPG in Navigator and Figure 9 shows the screenshot of an information bar for a user selection.


**Figure 8. Program guide.**


**Figure 9. Info Bar user interface.**

The Navigator was designed to run on both high-end and low-end receivers therefore, all the set-top boxes will have a Navigator application. In the case of a low-end set-top box, only a small amount of local storage is required to save a viewer's profiles. The Java bytecode size was about 112 KB and the start-up latency six seconds. The time was used mainly by transparent user interface components (cf. figure 6 and figure 9) and once started, the average browsing delay was approximately 1-2 seconds.

The Navigator had two versions; the first used local DVB-SI (cf. [P1]), while the second used DVB-SI (cf. [P2]) from a remote server. The server implementation is outlined in publication [P2], while publications [P1] and [P2] have detailed descriptions of the design, functionality and implementation of the Navigator

## 2.3 DIGITAL TELETEXT SERVICE

Digital Teletext is an information service, which appears to run unattached to any normal television programs. It may resemble current analogue Teletext services in content but have a much-enhanced look and feel. It provides information on topics that include: news, weather, sports, EPG, cinema listings, online newspapers, online TV shopping and advertisements, etc. The distinctive feature of this application is presenting vast amounts of textual information.

There are no common solutions to content representation, presentation, and portability. One idea is that users could open a Web browser window to reveal digital Teletext information as well as the Internet. However, the Web browser has large code overhead and would have to compete for the very limited screen and input resources.

Another idea is to implement the digital Teletext service as a stand-alone application and this approach was used in this thesis. It was designed as a resident application with both local and two-way interactivities. The pages are still information units and represented as XML pages accompanied by their corresponding Document Type Definitions (DTDs), which are used to define the data structures of different page types used by the digital Teletext service. The pages are transmitted via MPEG-2 transport streams instead of the Vertical Blanking Intervals (VBI) used in analogue transmissions.


**Figure 10. Main menu of Digital Teletext.**

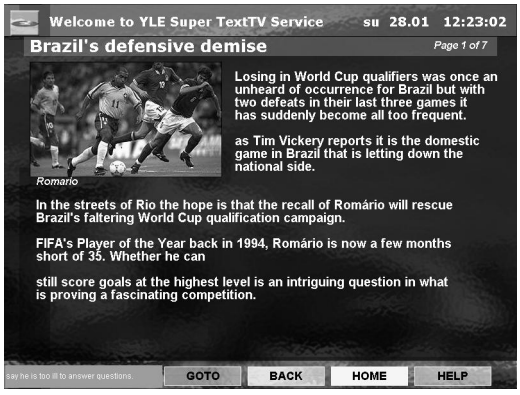


**Figure 11. Page from sports.**

Digital television user interfaces are simple to navigate and provide a rich graphical multimedia user experience, i.e., text, enhanced graphics, images, forms and animations, etc. (cf. figure 10-13). The navigation menu is used to index all the subjects instead of accessing using three-digit numerical codes (cf. figure 10). Simply use the up, down, and select buttons on your remote control to highlight and select. The viewer can also move backwards and forwards with the help of colour buttons and no matter which page you are viewing, you can access a helpful "pop-up menu" which takes you to an index of all sections simply by pressing the blue coloured button.

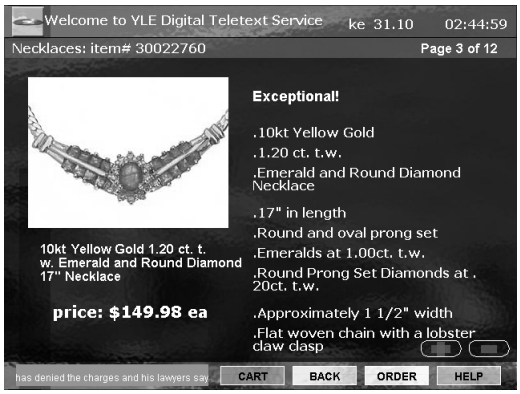


**Figure 12. Page from TV shopping.**

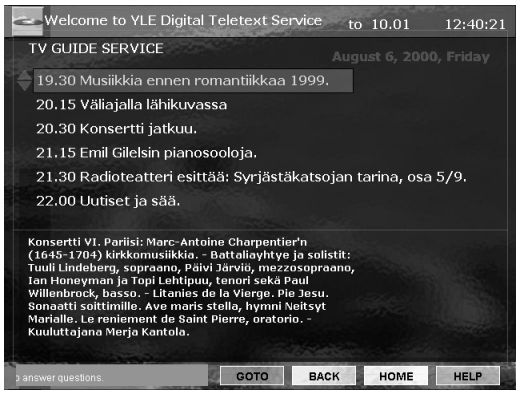


**Figure 13. Page from TV guide.**

This application was designed so that all the information was represented in XML format. It is able to parse XML pages and operate on both high-end and low-end set-top boxes and can cache a few frequently used pages to accelerate speed. With digital Teletext services the viewer can browse large amounts of information offered by service providers. The information ranges from news, sports, weather and TV guides to stock market data, transportation and TV shopping, etc. The start-up latency was approximately five seconds, which was taken by the user interface initialization and pages used were stored in local memory. Publications [P3] and [P4] give a detailed description of the services while publication [P4] adds the two-way interactivity to the service if a return channel is installed.

In many countries cable TV providers offer reasonably priced high-bandwidth Internet connections via cable modems. Users could consequently access the entire WWW with a traditional web browser that is incorporated into the TV at high speed. In such a scenario an interactive digital Teletext service may seem necessary as one can directly access all the services offered by the WWW through the TV. Although the WWW will have a promising future on a TV platform the existence of a digital Teletext service is reasonable for several reasons:

- The code size of a digital Teletext application (hundreds of Kbytes) is much smaller than a Web browser (tens of Mbytes). The memory consumption of a digital Teletext is also much smaller than a Web browser.
- A digital Teletext is low cost - even low-end set-top box can run it.
- The user interface and navigation are easier to use than Web browsing. For example, digital Teletext does not have hyperlinks, which are difficult to navigate via a remote control.

- The page transmission rates will be higher than the Web page over a standard modem.
- Pages from the broadcast object carousel are transmitted periodically so that there is no server overload and network traffic.
- Content authoring and production of digital Teletext services will have limited differences compared to that of analogue Teletext.

## 2.4 INTERACTIVE PROGRAM

Program-specific applications (i.e., interactive television) represent an important category of digital television services. They are created for and deployed with specific service audio-video programs. Examples include an application deployed with a game show that allows viewers to play along at home; an application that provides interactive information such as scores, statistics and different camera angles while watching sporting events and a movie that is accompanied with editorial content like reviews, actor biographies and related e-commerce opportunities. These applications have several key requirements. For instance, application code must be synchronized with the audio and video and the receiver hosting the application must be able to suspend the operation when the viewer changes channel.

The application code and data are usually stored in the object carousel. The application signaling information (AIT) is carried with the program in the MPEG-2 transport stream. When the viewer activates the interactive program by pressing "app" button on the remote control, application code and data will be dynamically loaded into the set-top box's memory, which can be released after the viewer quits the service. In an interactive program, application code usually requires handling of video (i.e., synchronization video with data content, resize video, etc.) as well as interactivity.


**Figure 14. Main menu of ice hockey.**


**Figure 15. Chat of ice hockey.**

In this thesis an ice hockey sports program was developed and presented in publication [P5]. The ice hockey program handled both one and two-way interactivity (e.g., chat, check scores, advertisement, etc.) and synchronization with video (cf. figure 14). The Java bytecode size was about 80KB and the start-up latency was around 3 seconds. Transparency was needed when displaying chat board on video (cf. figure 15).

## 2.5 SUBTITLES

Digital television subtitle services do not belong to the category of interactive applications, however digital television uses a new digital subtitling system, which provides interactivity. The subtitle data is carried with the television program and exists in the private data (which is transported with the MPEG-2 transport streams [47]). The DVB bit-map method operates by converting each subtitle row into a graphics image and transmitting it as a bit-map object. The biggest difference from analogue television subtitles is that the viewer can control the visibility and languages of digital television subtitles. DVB subtitling relies on DVB subtitle compliant solutions at both ends of the transmission chain, i.e. both the subtitle encoding and decoding processes have to be DVB subtitling compliant.

Subtitles are important because of the benefits to the hard of hearing community and also for language translation. Digital broadcasting is becoming a global business and satellite transponders have no respect for geographic regions or international borders. Hence a broadcaster will often target TV channels at a viewing population who do not necessarily share a common language. DVB subtitle compliant receivers can be configured to display subtitles in a range of languages.

The essential task of implementation is to ensure the interoperability of the code and to minimize memory usage. The greatest problems experienced when decoding digital television subtitles are delay and content synchronization [48]. Delay is a particular problem experienced during real-time subtitling, when a subtitler simultaneously creates and transmits the words from server to the receivers.

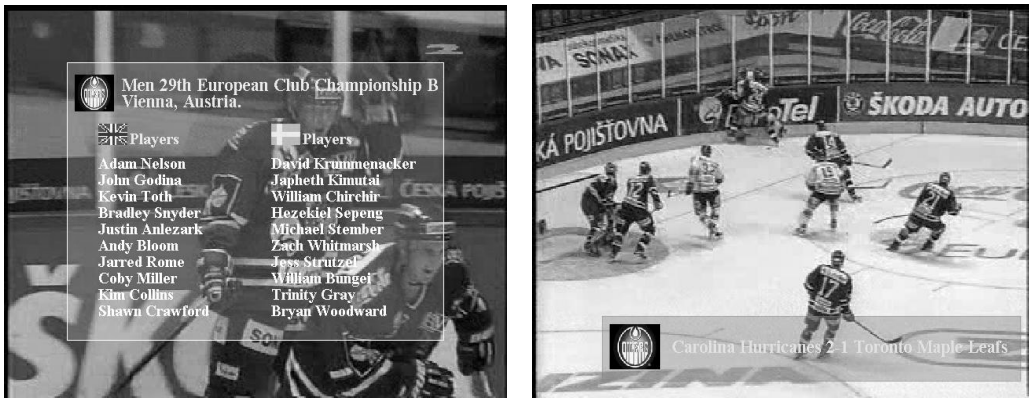


**Figure 16. Subtitle examples**

Figure 16 shows screenshots of typical subtitle examples. Publication [P6] discusses the details of the DVB subtitling system and gives decoding approaches for implementation in a set-top box. The approach presented in the thesis uses a software (i.e., Java) solution instead of hardware chip decoding so that the interoperability can be ensured.

## 2.6 SOFTWARE RESOURCES

The application source code is available on-line and can be downloaded from:

http://www.tml.hut.fi/~pcy/thesis/code.zip

# 3 SYSTEM ARCHITECTURE DESIGN

The objective of the system architecture design was to provide a reference implementation of set-top box middleware for running interoperable applications. Figure 17 shows the diagram of the system architecture which was defined in terms of three layers: hardware and software resources; middleware and applications. Typical hardware resources are MPEG processing (i.e., video, audio, and data decoders); CPU; memory; a graphics processor (i.e., OSD); modem and network interface; tuner and demodulator; demultiplexer and decryptor; smart card reader; remote control and storage devices, etc. [19]. The software resources include all device drivers as well as the RTOS etc.

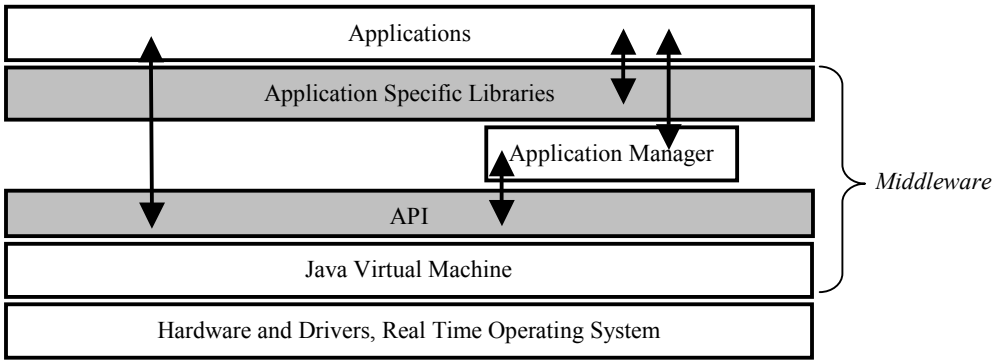


**Figure 17. System architecture for applications.**

## 3.1 MIDDLEWARE

Middleware includes a Java virtual machine; APIs; an application manager (and application specific libraries) and/or resident television-specific applications, for example, the Navigator and digital Teletext. The real time operating system (RTOS) and related device-specific libraries control the hardware via a collection of device drivers. The operating system provides the system-level support needed to implement the Java virtual machine and class libraries that comprise the DVB-Java platform [46].

The Java virtual machine (JVM) is used to interpret an applications Java bytecodes and is responsible for a systems hardware and operating system independence. It will have a relatively small size and the ability to execute code securely. A Java virtual machine knows nothing of the Java programming language, only a particular binary format known as the class file format [49]. A class file contains the Java virtual machine instructions (or bytecodes) and a symbol table as well as other ancillary information. Some important mechanisms provided by the Java virtual machine are vital to digital television applications. For example, bytecode verification provides guarantees about the validity of instructions being executed; class loading mechanisms enforce how code is loaded into the machine and can provide guarantees about the code's source while strong name-space management decreases the chance of code-spoofing, etc.

API's operate within the hardware context of the set-top box and encapsulate the functionality exposed by the system libraries that control the television-specific hardware on the device [46]. APIs provide an abstraction that allows the application programmer to

remain unaware of the details underlying the digital televisions hardware environment. The APIs used in the system architecture include basic Java APIs; JMF APIs; a Java API for XML parsing and a Java API for XML Messaging. The application libraries used by the resident applications are also stored on set-top box. One of the basic requirements for middleware is a small footprint and, in addition, the middleware can be stored in Flash ROM to improve performance (i.e., fast response) in set-top box. Publication [P7] gives the results relating to storage and memory consumption of the middleware used for this thesis which include an application manager to control the applications running on it and application libraries. The implementations can support low-end set-top boxes with small amounts of RAM and no additional storage capacity.

### 3.2 APPLICATION MANAGER

Using an application manager, a set-top box is capable of running multiple applications simultaneously. Therefore, the application manager must complete a series of tasks, such as managing the lifecycle of applications (including both resident and signaling applications); accessing system resources; system adaptation and managing remote control keys. Figure 18 shows the abstract functions performed by the application manager created for this thesis.

The application manager was designed as the main entry point for the execution of applications and consequently only the application manager can activate applications. The application manager can be started when the viewer activates one of the interactive services. When the Java virtual machine starts, there is usually a single non-daemon thread (which calls the method named main of some designated class). The Java virtual machine then continues to execute threads until the exit method of class Runtime has been called.

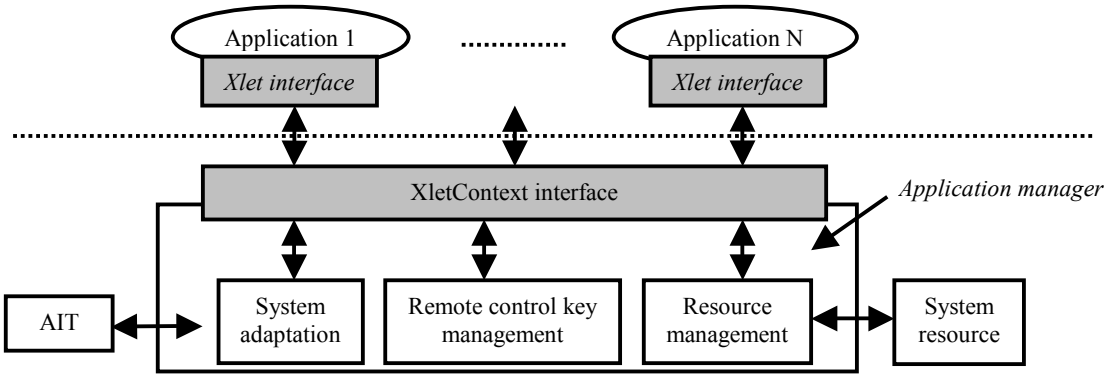


**Figure 18. The Function of the application manager.**

Applications (called Xlets) may be provided from different service providers and in order to run services on a common platform, applications must exhibit a common behavior (i.e., an interface). To run multiple applications simultaneously each application must be designed as a system level thread. The application manager communicates with applications via an interface signaling mechanism and each application communicates with the application manager via an interface passed down to it. An application is allowed to exists in one of four lifecycle states (i.e., initialized, running, paused and terminated). Both the application manager and an application manage error signaling and exceptions. Publications [P7] and

[P8] present a detailed description of the principle and implementation of the application manager.

In addition to setting up the system the application manager must execute using characteristics specified by the XletContext interface. They include four functional methods: starting, resuming, pausing and destroying an Xlet. The mechanism implemented to achieve this makes use of a properties file; cached environment variables and cached Xlet data. They are a shared resource used by both the application manager and Xlets. The application manager was implemented purely in Java and its storage size, memory consumption, time delay (start up, switch) are referenced in publications [P7] and [P8].

An application must execute using the characteristics specified in Xlet interface which consists of four methods: start, resume, pause, and stop. The Xlet employs a thread monitor variable, which allows multiple Xlets to run simultaneously in the set-top box as long as sufficient memory resources exist. The principle behind the diagram is that the monitor monitors the lifecycle state, for example, when the lifecycle state is changed from paused to live the current Xlet thread will waken and resume execution immediately. Each Xlet also has the possibility to launch other Xlets, however, only the live Xlet has the key focus.

The application manager is also responsible for system adaptation (e.g., adaptation of screen aspect ratio and size). When an application is started, the application manager will pass the current screen aspect ratio and size using the environment variables contained in the interface. This allows the application to calculate the correct resolution and resize all the graphical user interface components.

In a multitasking PC based operating system the problem of key interference does not exist since the operating system takes care of which application (i.e., window) has focus and knows which application to forward user inputs. In the digital television environment several concurrent applications can potentially interact with the viewer and therefore a mechanism must be defined that avoids key interference. The key managers avoids key grabbing problems that may exist when coexistent Xlets are spurned ensuring that the response of each Xlet to a particular user input will not result in cross interference.

## 3.3 SUMMARY

This chapter discussed the system architecture designed to run interactive services and ensure interoperability. The design mechanism and methods of dealing with the difficulties of a multi-Xlet system are described. Serious problems associated with a multi-Xlet system include resource management and namespace issues. Solutions to these issues are well documented in the papers, although some performance issues were not considered. Multiple Xlets can run on the set-top box concurrently, however the number of running Xlets will ultimately affect system performance. Therefore, the overall number should be considered carefully. When an application is deadlocked, no key can be pressed and therefore the application manager should have ability to handle applications that (i.e., emergency an exit and system timeout).

# 4 JAVA USER INTERFACE

Watching TV and using a computer are fundamentally different tasks as a television screen is significantly different from a computer's monitor and a remote control becomes the main input device (a computer style keyboard is normally an option and a mouse not practical). The main components of a digital television graphical user interface include video and audio; subtitles; interactive service graphics; animation and large amounts of text. Consequently, digital television applications have more critical user interface requirements and constraints resulting in increased complexity of the user interface design.

## 4.1 CONSTRAINTS AND CRITERIA

With the increased functionality of interactive digital television, the availability of screen and input devices is always an important issue in user interface design. In addition to the general constraints and requirements introduced in Chapter 1, the following points are also pertinent:

- Screen space is small and the viewing distance large, therefore user interface graphics must share screen space with the TV program. This forces presentations to use correspondingly large font sizes and requires the segmentation of content into screen-sized pages.
- The users will not tolerate vast amounts of navigation with a remote control and thus the complexity of the input device arises.
- Digital television interactive services are not the same as Internet tasks. For example, scrolling through content and navigating between hyperlinks can be difficult using a remote control because it only has up, down, left, right and select keys for this purpose.
- The fault intolerance and passivity of the user must be considered.
- The TV context must be maintained. The user interface appears as a graphic on the screen and must interfere with the primary TV program as little as possible.

The most important criteria when developing a user interface includes: code size; memory consumption; latency and bandwidth, as well as look and feel. Almost two thirds of the applications size and memory usage result from coding and executing user interfaces. Small code size is especially important in the case of a low-end set-top box. The size of the application code has a direct consequence on the size of the broadcast stream which, in turn, is a trade off between the cost of bandwidth and speed of downloading the application. The bigger the application code, the larger the bandwidth required to download it in a given time. Latency is a key factor in user interface development and delays should be as small as possible. Therefore, the methods employed for development should consider these criteria and follow three core rules, namely: extensibility and flexibility; robustness and reusability.

The development tools studied and used in the thesis are Java and JMF APIs which we call the Java user interface [P5]. In the following subsections, the three main issues of a Java user interface for digital television applications are discussed and the methods studied are outlined. Results are also presented relating to code size; memory consumption and ease of navigation.

## 4.2 SCREEN DISPLAY LAYOUT

Figure 19 shows a typical TV screen layout design for interactive television. The whole screen display consists of three display planes, namely subtitle OSD, JMF video container and application root container. These three planes are resources shared by all the applications which are able to perform basic control of video (e.g., scaling) and audio (e.g., turn on or off) using the JMF APIs. Applications can obtain running JMF players needed by interactive services via the application manager and environment variables (cf. chapter 3). The TV context is maintained as a high priority in this design.

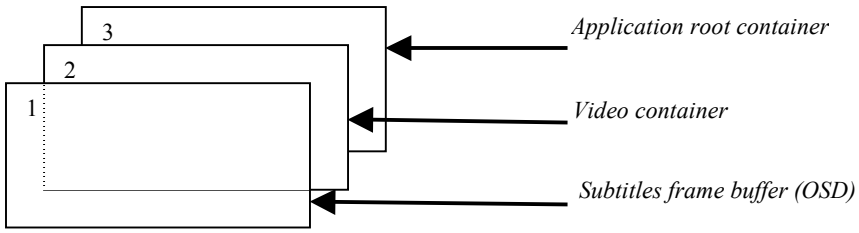


**Figure 19. TV screen display layout.**

The subtitle frame buffer is like a glass placed in front of everything. It is a separate display frame buffer which has nothing to do with the video container. The subtitle decoder is responsible for placing the decoded subtitle data on the frame buffer. The video container is used for rendering program video and is a heavyweight Java AWT component having no transparency to objects behind it. The application root container is used for drawing the applications graphical user interface. It can control the video container so that video is scaled to a specific size as a shared portion of the TV screen (with the graphical user interface) . The application root container can host multi-application user interfaces simultaneously.

## 4.3 PRESENTATION OF THE GRAPHICAL USER INTERFACE

One problem in developing a Java user interface is presentation. Two options can be employed to develop a graphical user interface for digital television. The first is the standard Java AWT widget set and JDK 1.1.X event model with various TV extensions for streamed media (video/audio). The second option is the Home Audio/Video Interoperability (HAVi) widget set and its event model. The thesis does not provide a detailed study of the HAVi user interface components, only a brief introduction.

HAVi is a standard that defines interoperable middleware system architecture for developing applications for digital products, such as cable modems; set-top boxes; integrated TVs; Internet TVs and intelligent storage devices for A/V content [50]. HAVi is essentially a distributed programming environment that provides mechanisms allowing inter-application communication over IEEE 1394. HAVi specifies a set of system services that form a foundation for building distributed applications including: messaging; events; device discovery; lookup functionality and configuration of streaming connections.

HAVi graphical user interfaces can be rendered on a range of displays varying from text-only to high-level graphical displays. The graphical user interface need not appear on the device itself and may be displayed on another display device from another manufacturer

[50]. To support this feature, two mechanisms are provided (i.e., the Level 1 UI and Level 2 UI). The Level 1 UI, called Data Driven Interaction (DDI), was intended for Intermediate A/V devices (IAV) and is the encoding of user interface elements. DDI elements can be loaded and displayed by a DDI controller which is used to generate messages in response to user input [50]. The Level 2 UI is a set of APIs based on Java and is used in the MHP [50] [51] [52].

### 4.3.1 Java AWT Widget Set vs Drawing Objects

In this thesis the first option is studied, i.e., using the standard Java AWT widget set and JDK event model. However, the look and feel of standard Java AWT components is not suitable for digital television applications. Using the standard Java AWT widget set increases rendering time and memory consumption, therefore a set of television-specific user interface components were developed including: TVButton, TVList, form objects and formatted text components, etc. These components are graphic-based and written either as lightweight modules by extending java.awt.Component or as drawing objects using java.awt.Graphics primitives.

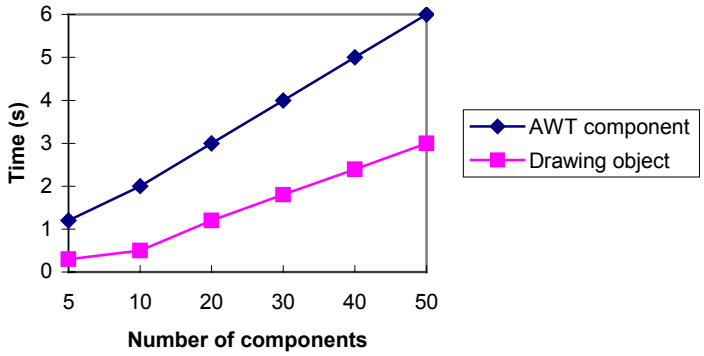


**Figure 20. Comparison of time delay.**

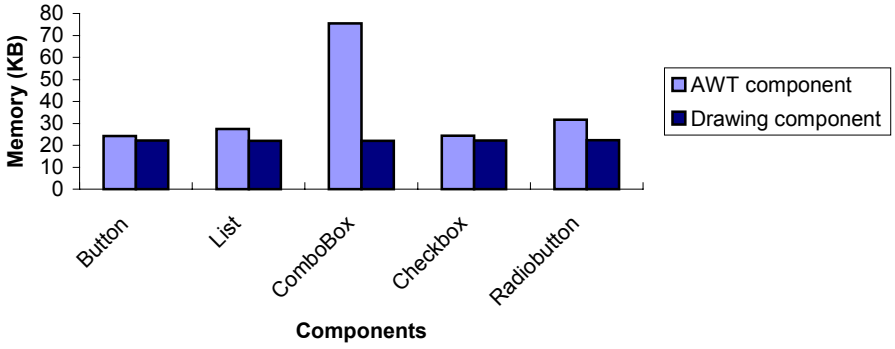


**Figure 21. Comparison of memory consumption.**

Increasing the number of components derived from Java AWT objects will increase the start-up time and memory consumption. Figure 20 shows the time delays of Java AWT components and drawing objects. The latency is increased more when adding additional standard AWT components, rather than adding drawing objects. Figure 21 shows that memory consumption can be significantly decreased when using drawing objects (e.g.,

ComboBox). Forms were designed in the main using drawing objects (cf. [P4]). Another advantage using drawing objects is that they make it easy to draw focus and navigate. The components designed avoided using image files, since they increase rendering and accessing time and are able to resize to adapt different screen aspect ratios set in set-top box.

### 4.3.2 UI Components Layout and Representation

Applications can obtain a handle to their application root container (cf. figure 19) via the application manager using environment variables (cf. chapter 3.2). An application can decide the size and position of the root container, which usually has a fixed and full screen size with an origin located in the upper-left corner on TV screen.

Layout of the user interface components was designed as a hierarchical structure with the application root container as its root node. These components should be placed manually (e.g., UI authoring tools) rather than using Java AWT layout managers, as a digital television UI needs a more flexible layout. Java AWT layouts are quite limited (e.g., border, bag, etc.) [53]. The rules applied are: Firstly, the first added components are displayed in front of those added later. Secondly, the last component added is displayed at the back. Thirdly, each component can be switched on or off. Finally, a heavyweight component is always displayed in front of lightweight one. Heavyweight components are discussed in [P5] and Figure 22 shows an example UI layout of the Navigator.


**Figure 22. An example of screen layout.**

UI components and their layout structure were described using the XML format via corresponding DTD definitions. Applications extract XML UI data information during initialization and draw them at run time. One advantage of this representation is that an application code does not require recompilation when the layout changes. Another advantage is that the user interface layout design is only focusing on the authoring stage, when all the information (e.g., size, position, background image, caption, color, etc.) can be stored in the XML files. The following code sample shows the DTD format of a typical layout used in Figure 22.

```
<!ELEMENT    Rootwindow (Background, ButtonBar, Menu+, Page)>
<!ELEMENT    Background       (Timer)>
<!ELEMENT    Background       (#PCDATA)>

<!ELEMENT    Timer    (#PCDATA) >
```

```
<!ELEMENT    ButtonBar ( ColorButton+, Menu+ )>
<!ATTLIST      ButtonBar
        W CDATA #REQUIRED
        H CDATA #REQUIRED>

<!ELEMENT    Menu    (#PCDATA)>
<!ATTLIST      Menu
        W  CDATA #REQUIRED
        H  CDATA #REQUIRED
        SW CDATA #REQUIRED
        SH CDATA #REQUIRED
        OX CDATA #REQUIRED
        OY CDATA #REQUIRED
        SOX CDATA #REQUIRED
        SOY CDATA #REQUIRED
        NUMBER CDATA #REQUIRED
        INTERVAL CDATA #REQUIRED>

<!ELEMENT    Page    (#PCDATA)>
<!ATTLIST      Page
         W CDATA #REQUIRED
        H CDATA #REQUIRED
        X CDATA #REQUIRED
        Y CDATA #REQUIRED>
```

This hierarchical layout has many advantages: it is suitable for a television UI; the UI can be well defined using the XML format and UI layouts together with the event model (cf. chapter 4.4) can provide ideal navigation among UI components.

### 4.3.3 Video/Audio Rendering and Synchronization

The JMF provides a framework for displaying time-based media that are independent of transport mechanism, transport protocol, and media content type [54]. The JMF defines a javax.media.Player interface for time-based media data. A Player object encapsulates the state machine required to acquire resources and manage the rendering of time-based media streams. A player of the JMF did the video-rendering task via Java AWT Component. A player object also provides various controls for the rendering facilities (e.g., audio volume and video picture controls). Publication [P5] has more description about the player.

JMF allows the specification of synchronization relationships between media and the clock that serves as the synchronization master for presenting media (e.g., video synchronization with subtitles, etc.). Decoders synchronize their operation with a set-top box reference clock. The synchronization between A/V and data is achieved by assigning time-stamps to A/V access units during encoding. These time-stamps specify the time when a video or audio access units should be decoded and presented. The video container (cf. figure 19) can be passed to applications so that its size can be controlled by applications.

## 4.4 NAVIGATION

A remote control is the main input device of digital television although, a keyboard is optional. A remote control driver was installed into the software resource and handled by the

RTOS. Navigation is completed using only limited buttons on the remote control with no cursor display on the TV screen.

### 4.4.1 A Remote Control Model

Figure 23 shows a conceptual model of digital television remote control [25]. The right part is used specifically by interactive services. Usually, there are four direction buttons and a 'select' key; two resident service keys (i.e., Teletext and Navigator); an interactive program key; a back key and four color keys (i.e., red, green yellow, and blue). Pressing the “Navi” button brings up the listing menu with different functions. Pressing the Teletext button brings up the digital Teletext service main page. Hit the “app” key, when an interactive program logo appears. The back and four color keys are used by interactive services.

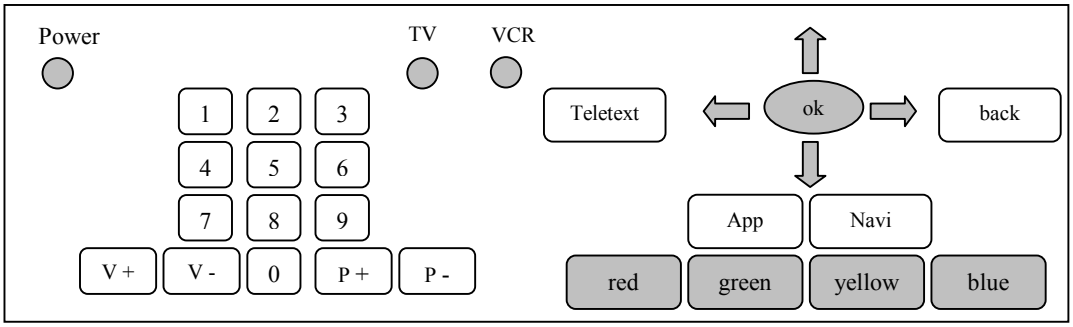


**Figure 23. A conceptual model of a remote control.**

Navigation between applications creates difficulties for user interface design and implementation as navigation should be obvious and consistent [55]. In addition to considering the usability issues, user interfaces must complete complicated navigation functions (e.g., calculation, reasoning, etc.). Some UI components may have text input functions, or have their own navigation functions (e.g., Radio buttons, Combo box, check box, etc.). Text information may appear as a collection of hyperlinks, etc., meaning that the number of UI components on the screen should be reduced and long forms that ask viewers to enter large amounts of text should be avoided. The complexity of the navigation function lies in implementing a simple navigation map within the user interface tree (hierarchical layout) using limited buttons (up, down, left, right, etc.).

### 4.4.2 Navigation Event Model

A delegation key event model was designed to solve navigation problems (cf. figure 24). Each event (e.g., hyperlink, button press, checkbox press, etc.) is a customized Java AWT event having a unique event ID. Each Event ID is associated with a customized key listener, which is registered with a parent container and is responsible for receiving events. The events are then multicast to the drawing sub-components. Each key listener can be added or removed from the container by its parent. This approach solves most navigation problems (especially hyperlink events) and has a fast response. Figure 24 illustrates the principle of an event model. After the container has been created it installs its key listeners and enables the key events. The container itself will take care of events generated by it’s child processes.

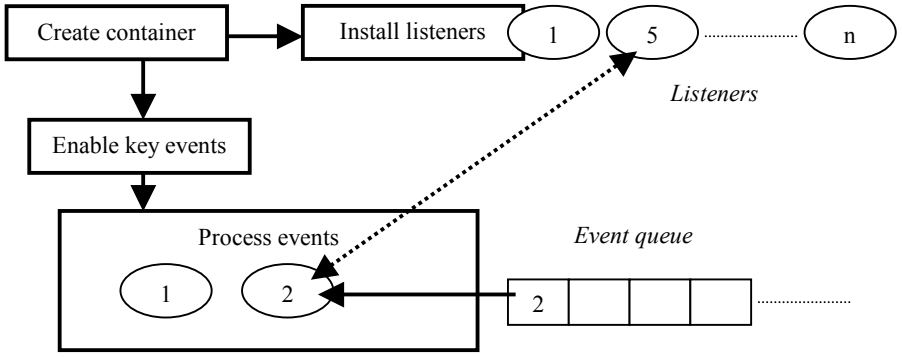


**Figure 24. Event model of a remote control.**

# 5 APPLICATION CONTENT

The application content in this thesis is defined as the information required to describe an application. It consists of the data to be presented by an application (e.g., text, graphics, image, animation); a structure to describe the data and a script to describe the interactivity (e.g., the workings of a quiz show). The characteristics of application content vary:

- Some data updates frequently and the continuous streams of information must be delivered directly from the service provider to set-top box. For example, news or stock tickers, live sports events, etc.
- Some data does not update frequently, but is relatively large for example, digital Teletext content.
- Most of an applications data content (e.g., DVB-SI tables, digital Teletext pages, subtitles, etc.) is stored in broadcast data and object carousels, and transmitted via the broadcast network embedded in MPEG streams.
- Content can be multi-lingual.

Basic requirements for application content is listed below:

- It should be rich in content and interesting to viewers.
- It should be able to adapt to a variety of different set-top box platforms and devices to ensure interoperability and portability [19].
- It should be up-to-date.
- It should have a small footprints.
- It should not increase the overall average latency of the application. For example, the access time and information update rates are important considerations in the design of a broadcast Teletext system [56].
- It should consume minimal network bandwidth.

Application content therefore, should be defined, authored, stored, transmitted and processed well in order to solve the above issues.

## 5.1 XML AND JAVA

Chapter 1.3.1 introduced the inefficiency of HTML as a data structure. The Extensible Markup Language (XML) is the universal format for structured documents and data on the Web [57] and is recommended by the World Wide Web Consortium (W3C). Like HTML, XML encloses data in tags but there are significant differences between the two markup languages. Firstly, XML tags relate directly to the meaning of the enclosed text, whereas HTML tags specify how to display the enclosed text. Secondly, XML is extensible, that is, they allow authors to define their own tags to describe content. HTML is limited to those tags that have been predefined in the HTML specification.

With the extensibility that XML provides, tags can be created using an XML schema language for a particular type of document. A schema describes the structure of a set of XML documents and can be used to constrain the content of the XML documents. The most widely used schema language is the Document Type Definition (DTD) schema language

[58] which also defines the hierarchical structure of an XML document, including the order in which the tags must occur [57].

Java platforms make application code portable and XML makes data portable [59]. XML and its scripting facilities provide many features suitable for representing application content in a digital television environment. The XML standard allows the application to define its own markup languages with emphasis on specific tasks. More importantly, XML delivers the interoperability of data across applications and hardware. The properties of XML markup make it suitable for representing data, concepts, and contexts in an open, platform, vendor and language neutral manner [59]. Also, there is a greater opportunity to reuse this data outside of the applications.

XML data and documents cannot display themselves, they must be parsed and processed by declarative applications. The Java platform can become a ubiquitous runtime environment for processing XML documents offering a hierarchical representation of data. XML simplifies the software development process and content organization and can also be used to "mark up" images, video streams, audio streams and other assorted binary data objects. This provides a mechanism to index, search and manipulate streams within applications.

In addition, both XML and the Java platform intrinsically support Unicode character sets [57]. Using the Unicode standard, applications can represent characters in multiple national languages and with XML markup as the format for data exchange and an internationalized application written in Java for processing, XML documents can be exchanged globally.

## 5.2 Data Structure of Application Content

Essentially, there are three types of data structures used in defining application content. Firstly, DVB-SI tables (cf. chapter 1.2.1) define a standard data structure to carry TV program schedules and interactivity script information, etc. Secondly, the DVB subtitle system has defined a structure for program subtitle data. Lastly, the data structure for information service pages (i.e. digital Teletext) is defined in this thesis and described in the following paragraphs.

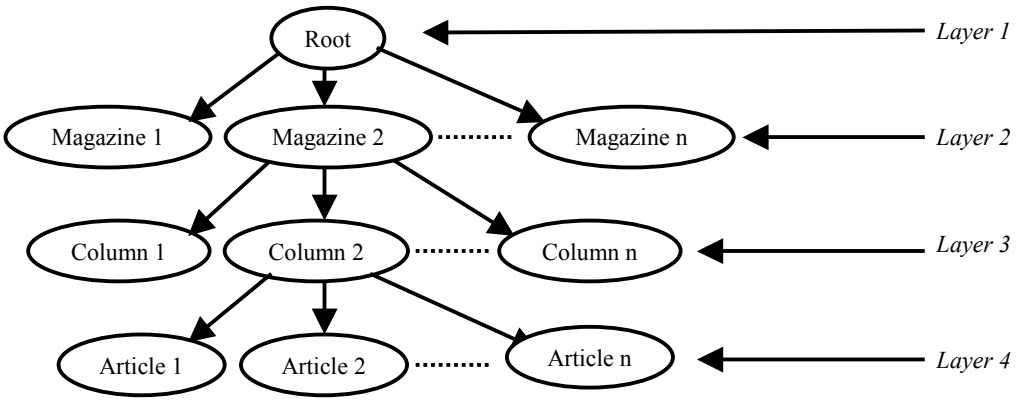


**Figure 25. Data structure of application content.**

The data information of digital television is a hierarchical structure in nature. Figure 25 shows the data structure of application content (excluding A/V content) defined in this thesis. It can be divided into four layers (deeper layers are not preferred for navigation and presentation, and therefore should be avoided): the root layer represents overall content; the second layer represents the category of information (i.e. news, sports, transportation, etc.); the third layer is a sub-category of nodes from layer 2 and the fourth layer contains leave nodes, which include concrete application content information. The data structure describes document-centric metadata information.

Figure 26 shows the metadata architecture of information content. It includes a start-up document, which represents the first three layers of the content outlined in Figure 25 and separate XML page and/or non-XML documents (which includes the data). This structure does not include a user interface document. The start-up page has pointers to separate pages, which can be organized in separate directories in an object carousel (e.g. page documents from each magazine are stored in a directory). The design described in this thesis utilizes DVB data carousel features. In addition, updating a page does not affect service browsing therefore, this structure is suitable for frequently updated content.

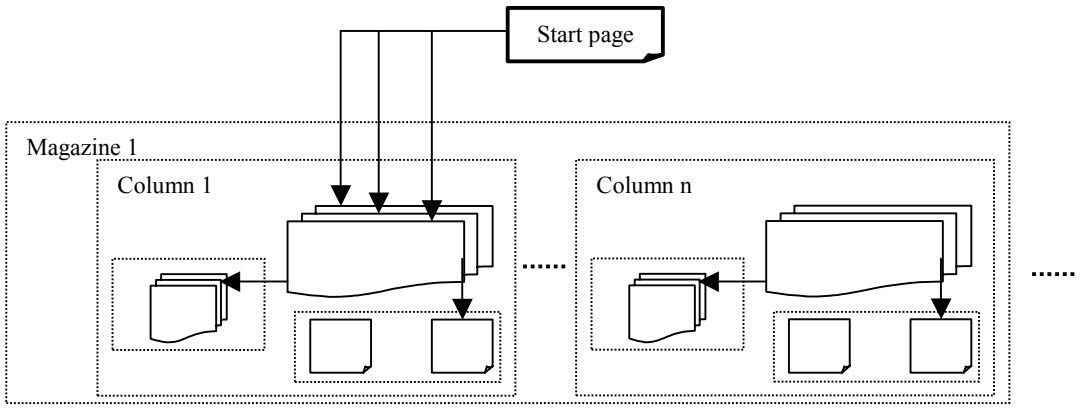


**Figure 26. Document architecture in XML.**

Part of creating XML markup language objects includes defining the elements, attributes and rules for their use. This information is stored in a DTD file which may be included within XML documents or the DTD's can be external. However, if the DTD is stored externally then the XML document must provide a reference to it. If a document does provide a DTD and the document adheres to the rules specified in the DTD then it is considered valid. Most XML page documents have the same DTD structure. Some special pages may have their own DTD structure, e.g., an order info page, news and stock ticker pages, etc. Each page can have text, images, graphics and interactive content (e.g., text with hyperlinks, forms, etc.). Publication [P3] and [P4] present more information. More importantly, hyperlinked pages can also be defined and retrieved by using a three digit number as its index (i.e., magazine number, column number, and article number).

All the names in a DTD are unique to avoid ambiguity however, if a particular XML document references more than one DTD there is a possibility that two or more DTDs

contain the same name. The document needs to specify a namespace for each DTD so that the parser knows which definition to use when it is parsing an instance of a particular DTD.

Table 2 lists figures describing the average size of each XML page defined for this thesis, according to the above data structure. The XML and DTD files for start page and user interface pages were downloaded together with application code and only need about 12 KB. The XML and DTD files for the article page needs about 50 KB (four images used on average). The creation of long XML pages must be avoided as the horizontal or vertical scrolling of a digital television user interface is not ideal. Images on the pages can take up to 60% of the available bandwidth [60]. Images should be as small size as possible, or they can be delivered in compressed binary code that is possible to embed in XML pages. A page with a small size can decrease retrieval time and consume limited cache memory in the set-top box.

| Start page (KB) | | Article page (KB) | | | UI page (KB) | |
|---|---|---|---|---|---|---|
| XML | DTD | XML | DTD | 4 Images | XML | DTD |
| 7 | 2 | 3 - 4 | 1 | 46 | 2 | 1 |

**Table 2. Size of application content pages.**

## 5.3 XML PAGES IN DATA CAROUSEL

XML pages are stored in a broadcast data carousel where the main structuring mechanisms are data carousel modules [27] [60] (cf. chapter 1.2.3). The metadata content may be stored within a single group, which is defined in the DVB data carousel specification. A group includes modules and one module contains the document files or directories that belong to one magazine (cf. figure 26).

A data carousel module contains one or more blocks and the blocks in a data carousel module exist in a sequential order. Each block is preceded by a header containing a unique identifier for the block and it's length. To retrieve an XML file from the data carousel the set-top box needs to parse the data carousel module until the requested block is reached. The size of a data carousel module is not limited in the DSM-CC standard but specific profiles often define a maximum size for practical reasons. From an application point of view receivers cache at least one object carousel module so that all files belonging to it are directly available to the client [60]. However, it is impossible for low-end set-top box to cache all pages within one magazine, when several blocks are possible.

During broadcast, the data carousel usually sends data carousel modules in sequential order, although this is not essential. It is possible to interleave data blocks to keep large files from blocking the broadcast stream exclusively for a long time [60]. Interleaving is also used in some application domains where set-top boxes can receive a data stream at full bandwidth for a short period of time only and need a pause to process the data before being able to receive the next block. The priority of modules in the data carousel is handled by giving each differing latency times which determines how often a module gets repeated during one 'rotation' of the carousel. Modules with short latency times will have a higher priority and be repeated more often than others [60].

There are a number of additional characteristics that influence the behavior of an application from content point of view, for example, the grouping of files and directories in modules, the order of these modules, and their priorities [60]. Methods to improve the delivery of applications exist and can increase the overall speed of the applications. However, experiments are needed.

## 5.4 CONTENT PARSING IN SET-TOP BOX

After receiving a viewer's request for an XML page, the application is responsible for transferring the XML page from the network, parsing it and presenting the content on the TV screen. Two approaches can be employed to parse application based XML pages: a simple API for the XML (SAX) model and the Document Object Model (DOM) model. The condition is that the JAXP APIs must be embedded in the set-top box as a part of middleware (cf. chapter 3.1).

Figure 27 shows the SAX model. To complete a parsing task a SAXParser object has to be created from a SAXParserFactory object. Then, an XMLReader can be obtained from the SAX Parser object, after which the XMLReader sets the ContentHandler, ErrorHandler, etc. Finally, the XMLReader parses and validates the XML document.

The SAX model is an event-based parser [61] which reads an XML document from beginning to end in a linear manner and, each time it recognizes a syntax construction, it notifies the application that is running it [62]. Data access is read-only. The SAX parser notifies the application via the ContentHandler interface through its methods (e.g., Start document, Start Element, etc.). The SAX parser can perform validation while it is parsing XML data, namely, it checks that the data follows the rules specified in the XML document's DTD.

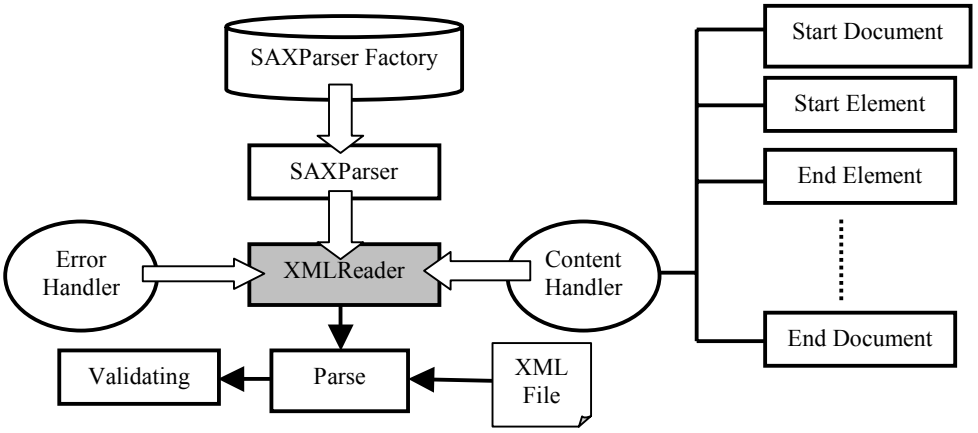


**Figure 27. The SAX model for content parsing.**

The SAX model is used only when a piece of data from a large document is needed, as the SAX parser parses data as a stream. It is very fast and tends to be preferred for server-side applications and data filters that do not require an in-memory representation of the data. Another advantage of the SAX model is its low memory usage (only events are generated). The disadvantage of the SAX model is that when XML data becomes more complicated, the application code will become difficult to design and maintain.

Figure 28 shows the DOM model. A DocumentBuilderFactory object is created, which is then used to create the DocumentBuilder object. The DocumentBuilder object carries out validation and error handling tasks, after which the DocumentBuilder object calls the parse method to parse an XML file and produces a Document object in memory.

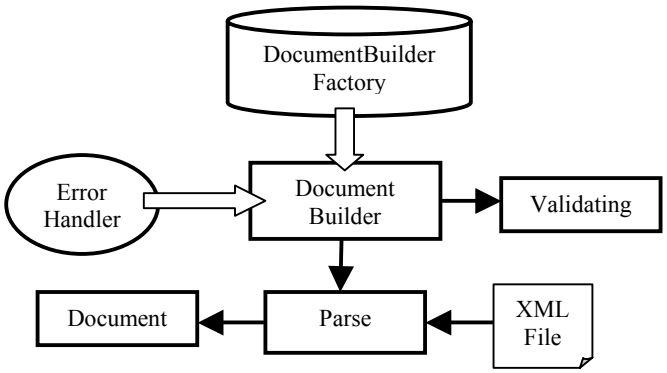


**Figure 28. The DOM model for content parsing.**

The DOM model reads an entire XML file and then saves it as a document tree in memory [61]. It allows the application to parse an XML document and obtain a Document object from a DocumentBuilder. The DocumentBuilder can be obtained from the DocumentBuilder Factory but, unlike the SAX parser, the DOM parser allows random access to particular piece of data in an XML document [62]. With the DOM model, one can build an object representation of the document; manipulate it in memory and add a new element or delete an existing one.

The DOM model is ideal for interactive applications because the entire object is present in memory, where it can be accessed and manipulated by the user. The disadvantage of the DOM model for content parsing is its high memory usage (the document is loaded in memory) however, this can be overcome by an appropriate document architecture (cf. figure 26).

## 5.5 CONTENT AUTHORING

An important part is to separate content authoring and application development. Digital television application content may include several hundred pages and some of these are updated on a regular basis, sometimes automatically with data from an external source (e.g., the WWW). It is impractical to update application code therefore, the content authoring process is very important for application layout and behavior.

Efficient authoring tools and content management systems are needed which may draw and edit application content (e.g., text, images, graphics, animation, etc.) based on templates. This authoring software may also review pages prior to transmission so that content can be validated and content correctness ensured. Its output will be an XML file.

XML parsers can immediately provide some content validation by ensuring that all the required fields are provided and are in the right order using a DTD. Although the SAX and

DOM models can validate application content it is preferable that content format validation and exception handling be completed outside of the application (i.e., during content authoring stage). In this way memory resources consumed and average latency can be reduced, making the whole system more reliable.

## 5.6 DISCUSSION

Application content itself must be interesting and be kept up-to-date. One possible approach that the service provider uses in content authoring is network information access from the Internet. It may automatically retrieve and filter dynamic content from multiple heterogeneous pre-defined sources (the WWW) for updates or new information.

# 6 RETURN CHANNEL COMMUNICATION MODELS

Interactive television is a new paradigm that will merge appeal and mass audience of traditional television viewing with the interactivity of the World Wide Web (WWW). It is an integral part of the new television experience. The interactive capability of a set-top box is a function of the network connectivity and the DVB broadcast system has the ability to utilize a return channel between a set-top box and the broadcaster, or service provider, to deliver data. This currently requires a modem and telephone or cable TV return path, although plans are afoot to provide satellite up-link capability (cf. chapter 1.2.4). MHP has proposed a return channel protocol stack (shown in figure 29) that can be used as a reference communication protocol [42].

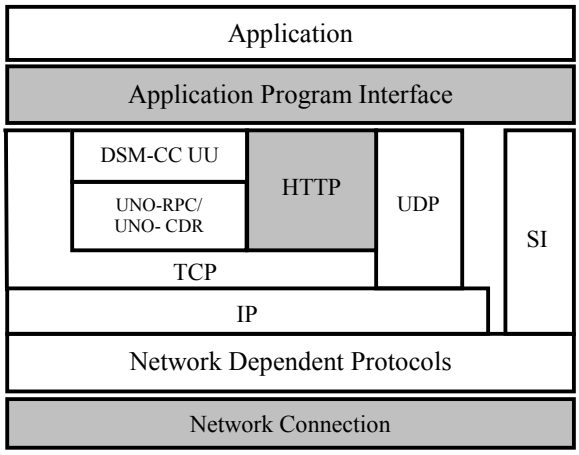


**Figure 29. Return channel protocol stack.**

Interactive services have various levels of interactivity and characteristics and each application will have it's own requirements, for example:

- Safely and reliably transfer data through a return channel.
- Reduce long latency connection time [63].

Reduction of interaction delay depends on many factors (e.g., increased physical network bandwidth). It is necessary to establish some communication methods (independent of the transmission medium (PSTN, Cable, etc.)) for digital television applications and this chapter will describe these with reference to the applications. The methods studied may provide guidelines for Java solutions in order to meet the interactivity requirements particular to different services. The basic communication modes can be divided into two types: synchronous and asynchronous based on a client-server structure.

## 6.1 SYNCHRONOUS COMMUNICATION MODE

The synchronous mode communicates in a point-to-point manner, such that when an application wants to communicate with the server reliably, a connection can be established and data can be sent back/forth. This model guarantees that data sent from one end of the connection actually gets to the other end and in the same order that it was sent, otherwise, an error is reported. Three approaches in Java can be adopted, and these are described below.

**URL Connection Model**

This approach uses the Uniform Resource Locator (URL) connection to access information on the Internet (HTTP). A URL can be the address of a resource on the server [64] and a Java application can use the URL to read from and write to that connection. Figure 30 shows the URL connection model. On the client side, once the URL connection has been established reading and writing can be done through output and input streams. On the server side, a CGI-bin script usually receives the data; processes it and then sends a response in the form of a new HTML page.

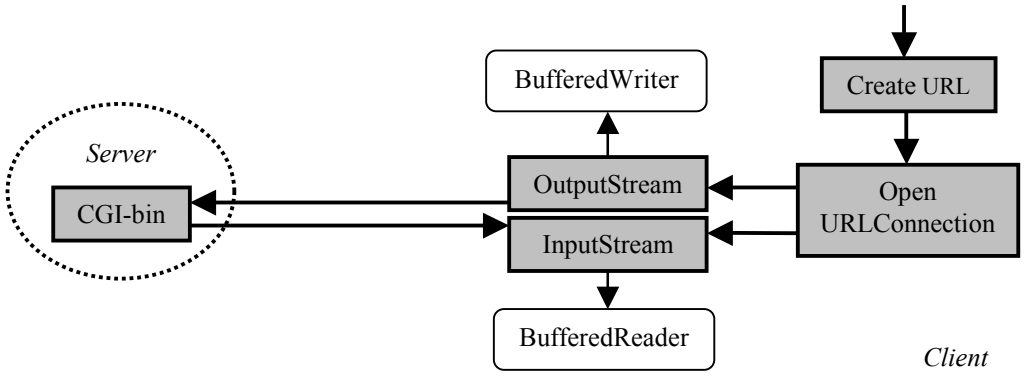


**Figure 30. URL connection model.**

**Socket Connection Model**

Figure 31 shows the Socket connection model where the server has a server socket which is bound to a specific port number [64]. On the client-side, the client knows the correct server name and port number. The server may support multiple clients who's requests can come into the same server socket. The server uses a thread for each client connection and once the server accepts a connection request from the client, a new socket is bound to a different port and a new thread spawned which reads from and writes to the client. A new socket is required in order that the server can continue listening to other clients. The client and server can now communicate by writing to or reading from their sockets.

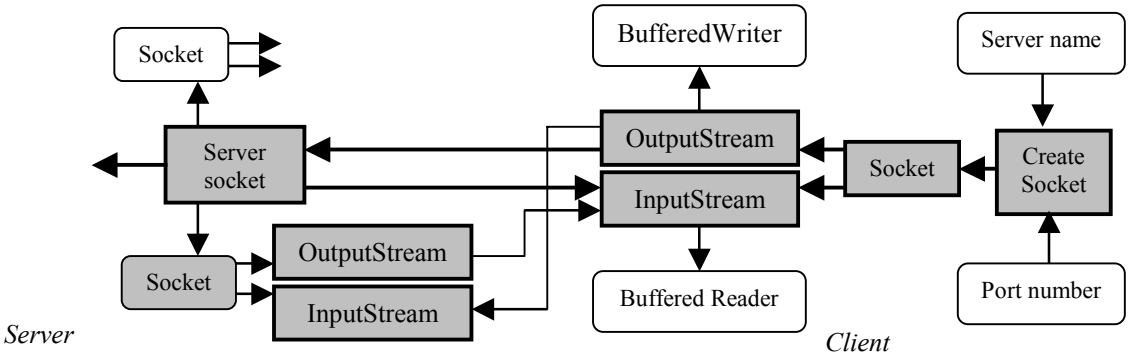


**Figure 31. Socket connection model.**

**SOAP Connection Model**

Figure 32 shows the SOAP connection model [65] in which a client can send SOAP messages directly to the server using a SOAPConnection object. The server runs a Servlet to receive a SOAP message and sends a SOAPMessage back to the client [66]. This communication model exchanges SOAP messages with or without XML format messages.

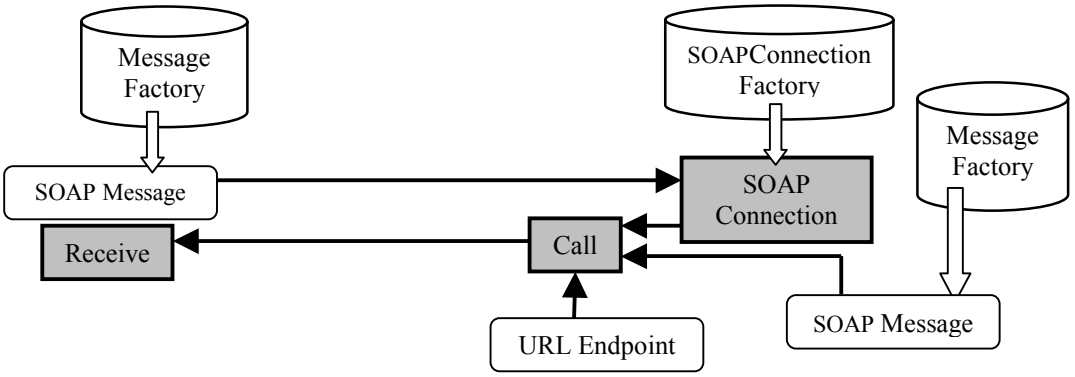


**Figure 32. SOAP connection model.**

### 6.2 ASYNCHRONOUS COMMUNICATION MODE

Point-to-point communication may cause performance degradation although its reliability can be guaranteed [64]. Asynchronous communication is a type of one-way communication between clients and servers where clients and servers do not have to, and do not need a dedicated point-to-point channel.

**UDP Connection Model A**

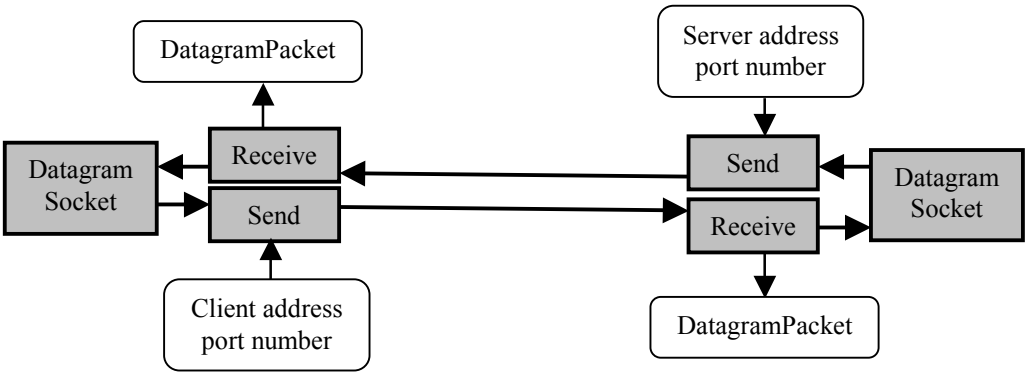


**Figure 33. UDP connection model A.**

The UDP connection model A is shown in Figure 33. The client communicates with the server via datagram packets, which are independent, self-contained messages sent over the network, whose arrival, time of arrival, and content are not guaranteed [67]. These clients and servers do not have and do not need a dedicated point-to-point channel. The server continuously receives datagram packets over a datagram socket and also sends responses to

the client. Therefore, the client can send and receive datagram packets to and from the server via a datagram socket.

**UDP Connection model B**

The UDP Connection model B is shown in Figure 34. On the client side a multi-cast socket is created which listens for datagram packets passively. The server can then broadcast datagram packets to multiple clients with the same group ID.

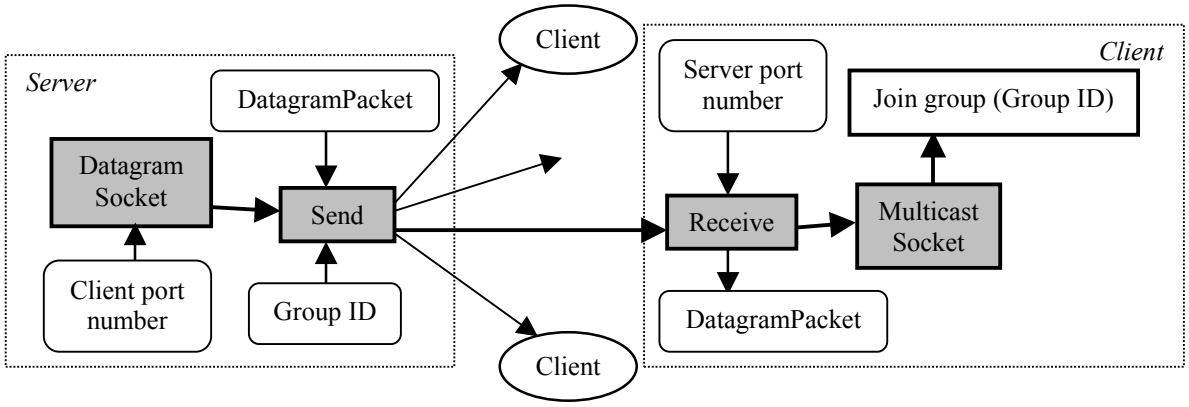


**Figure 34. UDP connection model B.**

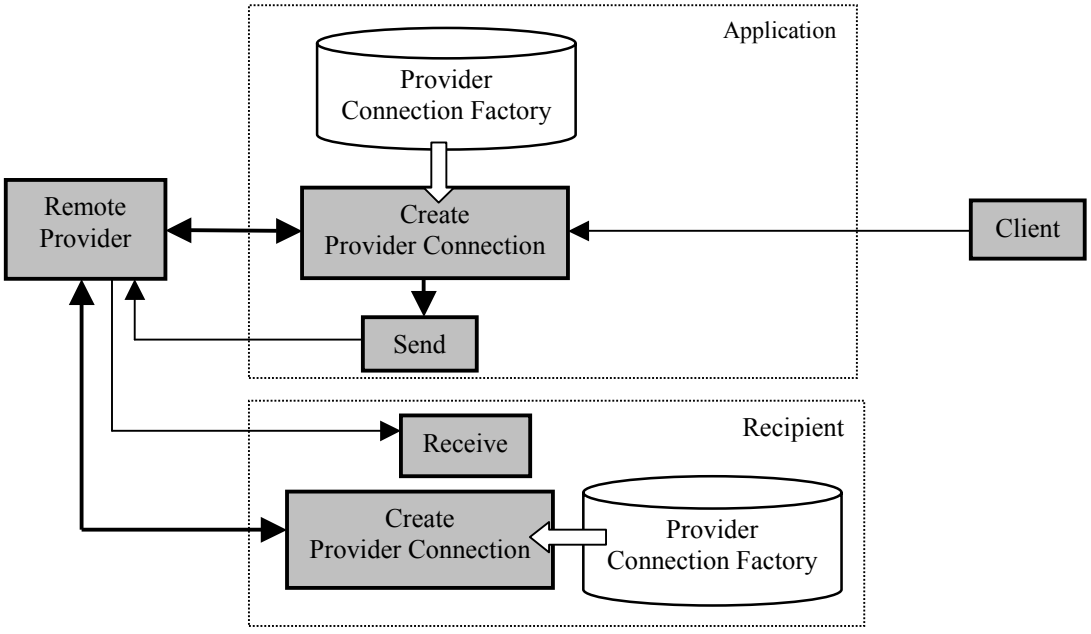


**Figure 35. Provider connection model.**

**Provider Connection Model**

The Provider connection model is shown in Figure 35, where all SOAP messages go through a remote provider [65]. The client or sender transmits a message to a recipient and does not

wait for a response, but continues processing other tasks. The remote provider takes a message sent by an application and caches it until the message has been successfully delivered to the receiver. Once the receiver gets the message, it may read and process or send a response message to the sender.

## 6.3 Comparison of Communication Models

The URL connection model is suitable for one-to-one communication from broadcaster to set-top box. It provides a relatively high-level mechanism for accessing resources on the Internet. This model is suitable for one-to-one and two-way communications and applications requiring reliable communications are suitable candidates. Typical examples are: advertisements, games, quiz shows, polls, etc. The URL connection model is appropriate for high-level connection to web resources.

The socket connection model is also suitable for two-way and one-to-many communication. Example interactive services can be chat clients, games, TV shopping, quiz shows, sports, etc. The socket connection model is more suitable for lower-level network communication and provides a reliable connection between client and server. That is, no data can be dropped and it arrives at the client side in the same order that is was sent.

The SOAP connection model is simple to initiate and is suitable for one-to-one and two-way communication. However, it has limited reliability and message delivery guarantees since it relies largely on the dependability of the underlying transport for delivering a message. This model is suitable for XML messaging services that require document-centric messages. Typical examples are: interactive digital Teletext [P4], messaging services, etc.

The Datagram (UDP) connection models A and B do not provide a mode of communication that delivers independent information packages where the arrival and order of arrival is guaranteed. These clients and servers do not have and do not need a dedicated point-to-point channel however, some applications may benefit from this type of communication. They are suitable for one-way and one-to-many communications and model B can be used on the client-side to listen for packets that the server broadcasts to multiple clients.

The provider connection model is suitable for one-way and one-to-many communication. This model also uses messaging mechanisms similar to the SOAP model. Example applications are: chat, messaging services, etc. and objects must be deployed in a Java Servlet container. This approach is reliable, although it is more complicated than the SOAP connection model.

## 6.4 Summary

According to the return channel protocol stack in Figure 29, there is the option to use DSM-CC User to User connections to deliver multimedia broadband services [68]. These interfaces or libraries (defined in the DSM-CC U-U protocol) provide modular building blocks that can be used for a wide range of multimedia applications, from video-on-demand to video conferencing to games and distance learning.

Finally, system operators must integrate Java technology into their servers to support the proposed communication models in order to deliver compelling interactive services to set-top boxes. The most popular servers are web servers, application servers, broadcast servers, transaction servers, etc. [19]. For example, in order for interactive television applications to receive new data to enable media synchronization or content updates for applications (i.e. stock-tickers or sports scoreboards) Java Servlets must be utilized on the server to receive, parse, package and send resource updates to applications. To support this, Java applications must have access to an underlying event signaling mechanism, which notifies the active application upon arrival of new resource updates. On the broadcaster's backend, a response collection and aggregation system must be set-up to deliver viewer feedback and transaction data, etc.

# 7 CONCLUSIONS

Set-top boxes will become the center of networked homes of the future and, as a result, digital television applications will be an important part of this. The research topics studied in this thesis will be useful and valuable for future application developers, service providers, set-top box manufacturers, and broadcasters. The main contributions in the thesis are summarized as follows:

- System architecture and application manager designs for applications. The design is efficient for the MHP applications and the Middleware structure is novel, especially embedding only an XML parser without an XSL parser. Some functionality, interfaces and protocols of the application manager are new ideas, for example allowing Xlets to access resources is novel and efficient.
- Java user interface issues. The idea of using Java AWT drawing objects for rendering application user interfaces is new and the methods for display layout using XML are noteworthy. The event model is a new design for input limited set-top boxes and the idea can be used in many other portable devices with limited inputs.
- Application content processes. The structure and representation of application content and the architecture of content documents for data carousels are new ideas. The concept of using either a SAX or DOM parser and suggestions of content authoring are novel and efficient.
- The synchronous and asynchronous communication models using Java technology are suggestions that can be used in different applications via a return channel.
- The design and implementation of the three applications and a subtitle system using the concepts and methods studied in the thesis.

The challenges in future digital television application areas come from both the evolution of the set-top box [64] and convergence of computer, communications, and consumer products. To enable evolution, the existing methods will not be appropriate, for example, there will be larger storage requirements, more powerful processors, expanding information, etc. Together, these developments will change the way television presents itself and the systems developed will require scaleable and more flexible solutions to the evolving set-top box applications in response to rapidly evolving demands.

The convergence of consumer products and communications presents a set of challenges to applications. Mobile digital television enables the same information to be broadcast to the mobile (e.g., WAP, GPRS) and portable (e.g., PDA, laptop) devices (while on the move). Re-implementing the same interactive application for dozens of different devices is unthinkable and authoring the same content three (or more) times for each target platform is not a sustainable proposition.

One possible common solution to the above challenges is to build entire service delivery system (both server and client) using Java technology, since the Java platform can offer the development speed, architectural flexibility and platform portability that will be required. However, targeted solutions to these problems must be studied and one option would be the use of intelligent agent technology as it can provide many routes toward solving the above problems. I am working on this.

# BIBLIOGRAPHY


1. M. Robin and M. Poulin, "*Digital Television Fundamentals - Design and Installation of Video and Audio Systems*," McGraw-Hill, 1997.
2. U. Reimers, "Digital Video Broadcasting," *IEEE Communications Magazine*, Vol. 36, No. 6, June 1998, pp. 104-110.
3. C. P. Sandbank, "Digital TV in the Convergence Environment," *IEEE Computer Graphics and Applications*, Vol. 21, No. 1, Jan.-Feb. 2001, pp. 32-36.
4. DVB, "Digital television global forecast report," Ovum Consultancy Ltd. in association with DVB project office, April 1999.
5. ETR 154, "Digital Video Broadcasting (DVB); Implementation guidelines for the use of MPEG-2 systems, video and audio in satellite, cable and terrestrial broadcasting applications," European Telecommunications Standards Institute, October 1996.
6. B. Fox, "Digital TV comes down to earth," *IEEE Spectrum*, October 1998, pp. 23-29.
7. TR 101 154, "Digital Video Broadcasting (DVB); Implementation guidelines for the use of MPEG-2 Systems, Video and Audio in satellite, cable and terrestrial broadcasting applications," European Telecommunications Standards Institute, July 2000.
8. B. Fox, "Digital TV Rollout," *IEEE Spectrum*, Vol. 38, No. 2, Feb. 2002, pp. 65-67.
9. Tektronix, "A Guide to MPEG Fundamentals and Protocol Analysis (Including DVB and ATSC),"1997.
10. Doc. A/54, "Guide to the use of the ATSC digital television standard," Advanced Television Systems Committee, October 1995.
11. Doc. A/53B, "ATSC Digital Television Standard (Revision B)," Advanced Television Systems Committee, August 2001.
12. T. Saito, "ISDB-S – Satellite Transmission System for Advanced Multimedia Services Provided by Integrated Services Digital Broadcasting," *ABU Technical Review*, No. 189.
13. T. Saito, "ISDB-T – Satellite Transmission System for Advanced Multimedia Services Provided by Integrated Services Digital Broadcasting," *ABU Technical Review*, No. 189.
14. T. Saito, "ISDB-C – Satellite Transmission System for Advanced Multimedia Services Provided by Integrated Services Digital Broadcasting," *ABU Technical Review*, No. 189.
15. M. Azimi, R. K. Ward and P. Nasiopoulos, "New Interactive Services for Digital TV," *Proceedings of International Conference on Image Processing*, 2001, Vol. 1, 2001, pp. 814-817.
16. EN 300 421, "Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for 11/12 GHz satellite services," European Telecommunications Standards Institute, August 1997.
17. EN 300 429, "Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for cable systems," European Telecommunications Standards Institute, April 1998.
18. EN 300 744, "Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television," European Telecommunications Standards Institute, January 2001.

19. G. O'driscoll, "*The Essential Guide to Digital Set-top Boxes and Interactive TV*," Prentice Hall PTR, 2000.
20. P. A. Sarginson, "MPEG-2: Overview of the systems layer," BBC Research and Development Report, No. 1996/2.
21. ETS 300 468, "Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems," European Telecommunications Standards Institute, January 1997.
22. ETR 211, "Digital broadcasting systems for television; Implementation guidelines for the use of MPEG-2 systems; Guidelines on implementation and usage of service information," European Telecommunications Standards Institute, April 1996.
23. J. G. Kim, H. Lee, J. Kim and J. H. Jeong, "Design and implementation of an MPEG-2 Transport stream Multiplexer for HDTV Satellite Broadcasting," *IEEE Transactions on Consumer Electronics*, Vol. 44, No. 3, August 1998.
24. M.C.D. Maddocks, "An Introduction to digital Modulation and OFDM Techniques," BBC Research and Development Report, No. 1993/10.
25. NorDig II, "Digital Integrated Receiver Decoder Specification, for use in cable, satellite and terrestrial networks," Version 1.0, June 2001.
26. M. Harigal, T. Nakazawa, T. Yoshida, Y. Hirakoso, K. Suto, and H. Neishi, "LSI Chip Set for Closed Caption Decoder System," *IEEE Transactions on Consumer Electronics*, Vol. 37, No. 3, August 1991.
27. EN 301 192, "Digital Video Broadcasting (DVB); DVB specification for data broadcasting," European Telecommunications Standards Institute, December 1997.
28. ETS 300 800, "Digital Video Broadcasting (DVB); Interaction channel for Cable TV distribution systems (CATV)," European Telecommunications Standards Institute, January 1998.
29. EN 300 473, "Digital Video Broadcasting (DVB); Satellite Master Antenna Television (SMATV) distribution systems," European Telecommunications Standards Institute, August 1997.
30. EN 301 193, "Digital Video Broadcasting (DVB); Interaction channel through the Digital Enhanced Cordless Telecommunications (DECT)," European Telecommunications Standards Institute, July 1998.
31. EN 301 195, "Digital Video Broadcasting (DVB); Interaction channel through the Global System for Mobile communications (GSM)," European Telecommunications Standards Institute, February 1999.
32. EN 301 199, "Digital Video Broadcasting (DVB); Interaction channel for Local Multi-point Distribution Systems (LMDS)," European Telecommunications Standards Institute, June 1999.
33. ETS 300 801, "Digital Video Broadcasting (DVB); Interaction channel through Public Switched Telecommunications Network (PSTN)/Integrated Services Digital Networks (ISDN)," European Telecommunications Standards Institute, August 1997.
34. EN 301 790, "Digital Video Broadcasting (DVB); Interaction channel for satellite distribution systems," European Telecommunications Standards Institute, December 2000.
35. ETS 300 802, "Digital Video Broadcasting (DVB); Network-independent protocols for DVB interactive services," European Telecommunications Standards Institute, November 1997.

36. TR 101 194, "Digital Video Broadcasting (DVB); Guidelines for implementation and usage of the specification of network independent protocols for DVB interactive services," European Telecommunications Standards Institute, June 1997.
37. G.S. Mills, W. H. Dobbie and G. Bensbeg, "DVB specifications for broadcast-related interactive TV services," *Electronics & Communication Engineering Journal*, February 1997, pp. 38-42.
38. ETR 289, "Digital Video Broadcasting (DVB); Support for use of scrambling and Conditional Access (CA) within digital broadcasting systems," European Telecommunications Standards Institute, October 1996.
39. TS 103 197, "Digital Video Broadcasting (DVB); Head-end implementation of DVB SimulCrypt," European Telecommunications Standards Institute, June 2000.
40. TS 101 197, "Digital Video Broadcasting (DVB); DVB SimulCrypt; Part 1: Head-end architecture and synchronization," European Telecommunications Standards Institute, June 1997.
41. J. - P. Evain, "The Multimedia Home Platform - an overview," *EBU Technical Review*, spring 1998, pp.4-10.
42. TS 101 812, "Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.0.1," European Telecommunications Standards Institute, October 2001.
43. Sun Microsystems, Inc., "Java Technologies for Interactive Television," Technical White Paper, 2001.
44. M. Milenkovic, "Delivering Interactive services via a Digital TV Infrastructure," *IEEE Multimedia*, Vol. 54, October - December 1998, pp. 34-43.
45. C. Fuhrhop, A. Kraft, and R. Kubis, "Object Carousel Simulator for Broadcast Applications," *Eurographics Multimedia'99 Workshop*, 7-8 September 1999 Milan, Italy.
46. B. Calder, J. Courtney, B. Foote, L. Kyrnitszke, D. Rivas, C. Saito, J. VanLoo, and T. Ye, "Java TV API Technical Overview," the Java TV API White paper, Version1.0, November14, 2000.
47. ETS 300 743, "Digital Video Broadcasting (DVB); Subtitling systems," European Telecommunications Standards Institute, September 1997.
48. G. Forbes, "Closed Captioning Transmission and Display in Digital Television," *Proceedings of Second International Workshop on Digital and Computational Video*, 2001, pp. 126 –131.
49. T. Lindholm and F. Yellin, "The Java Virtual Machine Specification," Sun Microsystems, Inc., 1999.
50. G. O'driscoll, "*The Essential Guide to Home Networking Techn*ologies," Prentice Hall PTR, 2001.
51. R. Baier, C. Gran, A. Scheller, and A. Zisowsky, "Multimedia Middleware for the Future Home," *the International Workshop on Multimedia Middleware*, Ottawa, Canada, October 5th, 2001.
52. P. Cesar and P. Vuorimaa, "A Graphical User Interface Framework For Digital Television," *Proceedings of the 10th International Conference on Computer Graphics, Visualization and Computer Vision*, *WSCG'2002*, Czech Republic, February 4 – 8, 2002**,** pp. 1-4.
53. G. Cornell and C. S. Horstmann, "*Core Java™,*" SunSoft press, 1996.
54. Sun Microsystems, Inc., "Java Media Framework API Guide," November 19, 1999.

55. M. - L. Tomsen, "*Killer content, Strategies for Web Content and E-Commerce*," Addison Wesley, April 2000.
56. Y.M. Siu, C. K. Chan, and K. L. Ho, "Real Time Teletext Broadcast System Performance Enhancement Using Ghost Rows," *IEEE Transactions on Broadcasting*, Vol. 43, No. 1, March 1997.
57. W3C Recommendation, "Extensible Markup Language (XML) 1.0 (Second Edition)," October 2000.
58. S. - St. Laurent and R. Biggar, "*Inside XML DTDs*," McGraw-Hill, New York 1999.
59. Sun Microsystems, Inc., "Portable Data / Portable Code: XML & Java Technologies," Available at http://java.sun.com/xml/ncfocus.html.
60. C. Fuhrhop, K. Hofrichter and Andreas Kraft, "Authoring of Teletext Applications for Digital Televison Boradcast," *the 6th International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services*, 12-15 October 1999, Toulouse, France.
61. T. Violleau, "Java Technology and XML Part 1 - An Introduction to APIs for XML Processing," November 2001. Available at http://developer.java.sun.com/developer/technicalArticles/xml/JavaTechandXML/.
62. R. Mordani, J. - D. Davidson and S. Boag, "Java API for XML Processing," Version 1.1 Final Release, February 6, 2001.
63. S. O'Leary, F. Ryan, B. Wynne and C. Gilliam, "Interactive Digital Terrestrial Television --- The Wireless Return Channel and the EU Sponsored WITNESS Project," *IEEE Transactions on Broadcasting*, Vol. 47, No. 2, June 2001, pp. 160-163.
64. J. Couch, "*Java™ 2 Networking*," McGraw-Hill, 1999.
65. N. Kassem, A. Vijendran and R. Mordani, "JAXM specifications, Java API for XML Messaging (JAXM) v0.93," Public Review Draft 2, August 2001.
66. D. Coward, "Java Servlet Specification," Final Release, August 8, 2001.
67. F. Halsall, "*Data Communications, Computer Networks and Open Systems*," Fourth Edition, Addison-Wesley, 1998.
68. ISO/IEC 13818-6, "Information technology -Generic coding of moving pictures and associated audio information: Extensions for Digital Storage Media Command and Control," 1998.

## APPENDIX A: ABBREVIATIONS OF THE DVB RETURN CHANNELS

| Standard | Description | Return channel | Document |
|---|---|---|---|
| DVB-RCC | DVB Interaction Channel for Cable TV Distribution Systems | CATV | ETS 300 800 |
| DVB-RCCS | DVB Interaction Channel for Satellite Master Antenna TV (SMATV) Distribution Systems | SMATV | EN 300 473 |
| DVB-RCD | DVB Interaction Channel through the Digital Enhanced Cordless Telecommunications (DECT) | DECT | EN 301 193 |
| DVB-RCG | DVB Interaction Channel through the Global System for Mobile Communications (GSM) | GSM | EN 301 195 |
| DVB-RCL | DVB Interaction Channel for Local Multipoint Distribution System (LMDS) | LMDS | EN 301 199 |
| DVB-RCP | DVB Interaction Channel through Public Switched Telecommunications Network (PSTN)/ Integrated Services Digital Networks (ISDN) | PSTN/ISDN | ETS 300 801 |
| DVB-RCS | Interaction for Satellite Distribution Systems | Satellite | EN 301 790 |
| DVB-NIP | DVB Network-Independent Protocols for DVB Interactive Services | | ETS 300 802 |

## APPENDIX B: SUMMARY OF DVB-JAVA PLATFORM APIS

| API | |
|---|---|
| Fundamental DVB-J APIs | Java Platform API |
| | MHP Platform API |
| Presentation APIs | Graphical User Interface API |
| | Streamed Media API |
| Data Access APIs | Broadcast Transport Protocol Access API |
| | Multicast IP over the Broadcast Channel API |
| | IP over the Return Channel API |
| | MPEG-2 Section Filter API |
| | Mid-Level Communications API |
| | Persistent Storage API |
| Service Information and Selection APIs | DVB Service Information API |
| | Service Selection API |
| | Tuning API |
| | Conditional Access API |
| | Protocol Independent SI API |
| Common Infrastructure APIs | DVB-J application lifecycle API |
| | Application Discovery and Launching API |
| | Inter-Application Communication API |
| | Basic MPEG Concepts API |
| | Resource Notification API |
| | Content Referencing API |
| | Common Error Reporting API |
| Security APIs | Basic Security API |
| | TLS / SSL Over the Return Channel API |
| | Additional Permissions API |
| Other APIs | Timer Support |
| | User Settings and Preferences API |
| | Profile and Version Properties |
| Java Permissions APIs | Permissions for Unsigned Applications API |
| | Additional Permissions for Signed Applications API |
| Content Referencing APIs | Transport Stream API |
| | Network API |
| | Bouquet API |
| | Service API |
| | DVB Event API |
| | MPEG Elementary Stream API |
| | File API |
| | Directory API |
| | Drip Feed Decoder API |
| | Irrelevant API |

# PART TWO: PUBLICATIONS

## List of Publications

[P1] Chengyuan Peng and Petri Vuorimaa, "A Digital Television Navigator," *Proceedings of IASTED International Conference on Internet and Multimedia Systems and Applications*, November 19-23, 2000, Las Vegas, Nevada, USA, pp. 69-74.

[P2] Chengyuan Peng, Artur Lugmayr, and Petri Vuorimaa. "A Digital Television Navigator," *the International Journal of Multimedia Tools and Applications*, Vol. 17, Issue 1, May of 2002, pp. 129-149.

[P3] Chengyuan Peng and Petri Vuorimaa. "A Digital Teletext Service," *Proceedings of the 9th WSCG International Conference on Computer Graphics, Visualization and Computer Vision*, Czech Republic, February 5 - 9, 2001, pp. 120-125.

[P4] Chengyuan Peng and Petri Vuorimaa, "Interactive Digital Teletext Service," *Proceedings of the 6th world Multiconference on Systemics, Cybernetics and Informatics*, July 14-18, 2002, Orlando, Florida, USA, Vol. X, pp. 181-186.

[P5] Chengyuan Peng and Petri Vuorimaa. "Development of Java User Interface for Digital Television," *Proceedings of the 8th WSCG International Conference on Computer Graphics, Visualization and Interactive Digital Media*, February 2000, Czech Republic, pp. 120-125.

[P6] Chengyuan Peng and Petri Vuorimaa, "Decoding DVB Digital Television Subtitles," *Proceedings of the 20th IASTED International Multi-conference on Applied Informatics*, February 18-21, 2002, Innsbruck, Austria, pp. 143-148.

[P7] Chengyuan Peng, Pablo Cesar, and Petri Vuorimaa, "Integration of Applications into Digital Television Environment," *Proceedings of the 7th International Conference on Distributed Multimedia Systems*, September 26-28, 2001, Taipei, Taiwan, pp. 266-272.

[P8] Chengyuan Peng and Petri Vuorimaa, "Digital Television Application Manager," Proceedings of the *IEEE International Conference on Multimedia and Expo 2001*, Tokyo, Japan, August 22-25, 2001, pp. 685-688.

# Summary Of Publications

This section of the thesis gives a brief introduction to the articles and summarizes my contributions referenced in the articles. The order of the publications is the cite order in part one of this thesis.

[P1] This paper described the functionality and theory of a digital television Navigator, which is the most important application in future set-top boxes. The paper systematically discussed the DVB-SI used by the navigator, the simulation of data carousels, Navigator user interface design, data organization, the remote control model and the Navigation process model. The demultiplexing of DVB-SI was implemented in a local server and the Navigator was designed with local interactivity and implemented in Java by the author. The author of this thesis is the main author of the article.

[P2] This paper represents the continuation of research presented in [P1] and includes new results. It describes a subtitle control function via the Navigator where the viewer can switch on or off subtitles and select appropriate languages. The remote server was designed and responsible for receiving transport streams, demultiplexing DVB-SI and sending demultiplexed tables to the receiver. A software reference model for DVB-SI multiplexing was also presented along with some performance results. The server side design and implementation (in C++ and Java) was done by the second author apart for the design of DVB-SI table encapsulation. The author of this thesis is the main author of the article.

[P3] This paper discussed the design of a lightweight digital Teletext browser with local interactivity for digital television and presented the application content model and data structure; page parsing processes; caching; graphical user interface; navigation and server design and implementation. The application content was defined using the XML language and implemented in Java by the author. The author of this thesis is the main author of the paper.

[P4] This paper is a continuation of research presented in [P3]. The design and implementation of two-way interactivity is presented along with a two-way communication mechanism based on the client-server model, using the SOAP Messaging mechanism. The server was designed and implemented using Java Servlet technology. The application can deal with forms (including interactivity) via a return channel; presentation and Navigation, etc. The author of this thesis is the main author of the paper.

[P5] This paper discussed the user interface issues of digital television applications using Java. It described the architecture of digital television application user interfaces and the APIs for presentation and streamed media as well as discussing the possibilities and merits of developing digital television user interfaces using Java AWT widgets and JMF tools. Some important issues, e.g., transparency and look & feel were described. A demonstration application (an ice hockey interactive program) was

developed in Java by the author. The author of this thesis is the main author of the paper.

[P6] This paper described the new DVB subtitling system and presented a software based decoding model for set-top boxes. The system uses MPEG-2 multiplexing and region-based bitmap graphics technologies to transmit subtitle data to set-top boxes. The decoding model was designed so that interactivity can be achieved via a Navigator user interface as well as allocation and organization in memory. Two implementation approaches were proposed using the API presented in this paper. One approach used OSD resources via the Java Native Interface (JNI) while the second approach used pure Java APIs and no OSD hardware. The author of this thesis is the main author of the paper.

[P7] This paper presented a system integration model to run different digital television applications in a prototype set-top box. The integration model included hardware, middleware (including operating system, Java virtual machine, APIs, and application manager) and applications. Some important results were presented and the author of this thesis was the main author of the paper. The applications and application manager used were developed in Java by the author while the second author migrated to the Linux and Kaffe platforms.

[P8] This paper presented the design and implementation of a middleware application manager. The application manager plays an important role in set-top boxes and this paper presented a communication model that allows an application manager to run multiple applications simultaneously. The application lifecycle model was designed in order that the application manager can signal and manage application states. Applications can be resident or signaled via the broadcast channel using an Application Information Table (AIT) transmitted within the program transport stream. The application manager was implemented in Java by the author and the signaling application was stored on a server and transmitted using HTTP via the Internet. The author of this thesis is the main author of the paper.