PUBLICATION P2

C. Peng, P. Cesar, and P. Vuorimaa. Integration of applications into digital television environment. In *Proceedings of the 7th International Conference on Distributed Multimedia Systems*, Taipei, Taiwan, September 26–28, 2001, pages 266–272. Knowledge Systems Institute.

INTEGRATION OF APPLICATIONS INTO DIGITAL TELEVISION ENVIRONMENT

Chengyuan Peng, Pablo Cesar, and Petri Vuorimaa
Telecommunications Software and Multimedia Laboratory,
Department of Computer Science and Engineering,
Helsinki University of Technology,
P.O. Box 5400, FI-02015 HUT, Finland.
Email: pcy@tml.hut.fi, pcesar@tml.hut.fi, and petri.vuorimaa@hut.fi

ABSTRACT

The overall software system integration to run interactive services in set-top box is hard and essential since there is no ready system to ensure the cross-platform interoperability of the applications. This paper presents a layered system integration model, which consists of interoperable applications, Application Programming Interface (API), system software, and software and hardware resources. The digital television applications are classified into three categories according to the service architecture. All applications are managed and controlled by an application manager, which plays a central role in the model. The set-top box hardware prototype used in the system is a single board computer. Linux with selected modules was chosen as the set-top box operating system. Kaffe Java virtual machine was ported to Linux platform to interpret Java bytecodes of applications. A working prototype system is introduced in the paper. Finally, we give the system running results and further improvement.

Keywords: software integration, middleware, application manager, Kaffe Java virtual machine, Linux, DVB-Java applications, set-top box.

1. INTRODUCTION

The digitalization of television is inevitable [1]. The main reason is that digital technology allows more efficient usage of radio frequencies. With digital broadcasting technology it is possible to transmit 20-30 Mbps within the capacity of one analogue television channel [2]. This means four or five television channels instead of one. At the same time, digital technology removes some defects, e.g., ghost images. The additional capacity can also be used for multi-channel sound system and additional speech channels.

The digital television allows also broadcasting of data. All kinds of interactive services can be introduced in digital television environment. Thus, the main task of system integration is to execute these digital television services to ensure cross-platform interoperability.

Our work follows the Digital Video Broadcasting (DVB)-Multimedia Home Platform (MHP) standard. The DVB-MHP encompasses the peripherals and interconnection of multimedia equipment via the in-home digital network. The MHP solution covers the whole set of technologies that are necessary to implement digital interactive services in the home - including protocols, common API languages, interfaces, and recommendations [3].

The DVB-MHP adds a technical solution for set-top box that enables the reception and presentation of applications in a vendor, author, and broadcaster neutral framework [4]. Applications from various service providers are cross-platform interoperable with different DVB-MHP implementations in a horizontal market, where applications, networks, and MHP terminals can be made available by independent providers.

In order to realize this goal, the DVB-MHP has proposed a reference model, which allows the development of high-level APIs and applications, independent of the DVB-MHP system infrastructure. The DVB-MHP reference model mainly consists of four layers, i.e., hardware and software resources, system software or middleware, APIs, and interoperable applications.

The hardware and software resources include tuner/demodulator, demultiplexer, decoders (A/V, DVB-SI (Service Information), subtitles, etc.), graphics processor, a common interface, a communication interface/modem, a Conditional Access (CA) module, memory, main processor, bootloader, remote control receiver module, Real-Time Operating System (RTOS), and associated drivers, etc. [5]. The minimum memory configuration is 16 Mbytes RAM, 4 Mbytes video RAM, and 8 Mbytes Flash ROM. The processor speed should be from 150 mips to more than 200 mips [5]. It is specified that 70 percent of CPU time should be devoted to run the applications, remaining 30 percent being used for system management [3].

ISBN: 1-8901706-09-8 © 2001 KSI

The system software layer isolates the interoperable applications from the hardware and software resources layer. The applications do not directly access resources. They use the APIs to access the resources of set-top box. The APIs provide the associated services to the applications. An API by definition is a set of high-level functions, data structures, and protocols, which standard interface for platform-independent application software [6]. The system software supports APIs, Java virtual machine, resident software (i.e., application manager, which controls the lifecycle and signaling of DVB-Java applications, called Xlets), databases (DVB-SI), etc.

The operating system is one of the most important software resources in set-top box. A set-top box operating system requires handling sophisticated tasks in a small memory space. It must be fast, reliable, and capable of functioning in real-time environment. A number of real-time operating systems have emerged to drive the next generation of advanced set-top box. Linux is one of the candidate operating systems. In addition to being free and having a large developer community, many of its features are suitable for set-top box (cf. Section 3.1).

The Java virtual machine is the key technology to ensure the cross-platform interoperability of the applications. The Java virtual machine is an abstract computing machine [7]. It is hardware- and operating system- independent. Like a real computing machine, it has an instruction set and manipulates various memory areas at run time. The Java virtual machine knows nothing of the Java programming language, only of a particular binary format.

2. APPLICATIONS

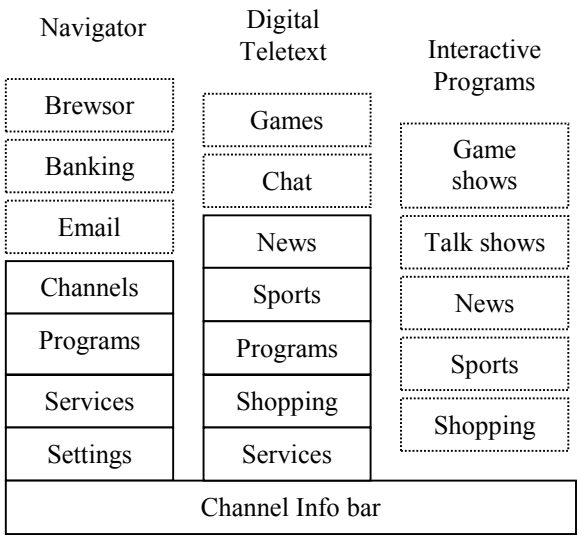

Figure 1. Digital television service architecture.

Figure 1 shows a digital television service architecture, which describes the roles of the different services [8]. The services are divided into three categories: Navigator, Digital Teletext, and Interactive Programs. The Navigator is the main index of digital television services. Through it, the viewer can access all the other services of the digital television. The Digital Teletext is an enhanced version of current Teletext service [9]. Finally, the Interactive Programs are enhanced versions of current television programs.

2.1 Navigator

The main task of the Navigator is to display program information to the viewer (cf. Figure 2). For example, in the DVB standard [10] the program information is transmitted according to the Service Information (SI) specification [11]. The DVB-SI fields contain information about the available networks, channels, programs, and also data services. The Navigator displays this information to the viewer and allows the viewer to access the different services.

The DVB-SI fields can also carry information about the forthcoming programs (e.g., channel, date, starting time, ending time, description, and type). This information is displayed by the Electronic Program Guide (EPG). In the above service architecture, the EPG is integrated within the Navigator. Thus, the Navigator is a place where the viewer can access information about services available now and in the future. The Navigator can contain information provided by different operators at the same time. Thus, it is a neutral information source, and should contain only generic information.

Figure 2. EPG from Navigator.

2.2 Digital Teletext Service

The current analogue Teletext [9] is a popular service, e.g., in Europe. The service contains textual information with simple graphics, which is displayed as pages. The user can access different pages by giving three digit codes with a remote control.

ISBN: 1-8901706-09-8

The main advantage of the Digital Teletext is that it can contain formatted text, tables, high-quality graphics, images, and even animations (cf. Figure 3). The viewer can access pages by menus and scrolling page lists on screen or even uses bookmarks and color coded hyperlinks, which are activated by the four color (i.e., red, green, yellow, and blue) buttons of the remote control.

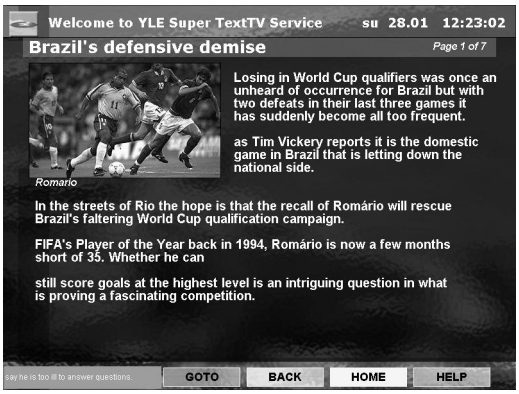

Figure 3. Sports page from Digital Teletext service.

The Digital Teletext service is in fact a simple browser. The main difference is that the interactivity is more limited. Thus, services that contain more interactivity (e.g., games) should be separate applications. The Digital Teletext content should be coded with some markup language. Extensible Markup Language (XML) is a natural choice for this purpose. In [12] and [13], we have described an XML based implementation of the Digital Teletext service.

Figure 4. Interactive program from Ice Hockey program.

2.3 Interactive Programs

Finally, the interactive programs form the third service category. They are usually based on some existing television program format. For example, infomercials, quiz shows, and talk shows are already interactive. The viewers have to use telephone, fax, or Internet to contact the service provider, though.

The main advantage of the digital television is that the viewers can use the interactive programs directly on their television screen with the remote control. The viewers can, for example, access background information, chat with other viewers, or place orders. The example application was an interactive service for ice hockey TV program (cf. Figure 4).

3. SYSTEM INTEGRATION MODEL

Figure 5 shows the system integration model with the necessary system components. It complies with the DVB-MHP reference model. The Linux and Kaffe OpenVM [16] were selected as the operating system and Java virtual machine of the set-top box, respectively. The XML parser and Java Media Frame Work (JMF) API are as part of the APIs. The resident applications (i.e., Navigator, Digital Teletext, and application manager), which belongs to the system software, are on the same level of the APIs since they uses both low-level and high-level APIs. Kaffe Java virtual machine is used to interpret the Java bytecodes of applications. The drivers include remote control, network, and display console driver, etc.

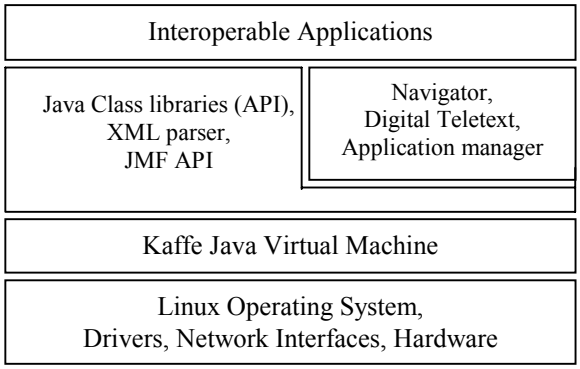

Figure 5. Digital television system integration model.

3.1 Set-top Box Operating System

Linux is truly global standard and most of its features are suitable for set-top box. For example, it is very easy to port to new platforms (i.e., a hardware change does not mean a re-selection or re-writing of the operating system). Linux is made for networking. Virtually all networking protocols in use in the Internet are native to Linux [17]. There are no runtime royalties. Linux is sophisticated, efficient, robust, reliable, modular, and highly configurable.

Although Linux is not a real-time operating system (i.e., the Linux kernel does not provide the required event prioritization and preemption functions), several add-on options are available that can bring real-time capabilities to Linux-based systems [17].

The Linux was selected as our set-top box operating system. All unnecessary components, like virtual memory

ISBN: 1-8901706-09-8

and networked file system, were removed from the operating system.

Set-top box is different from traditional desktop PCs. For example, X-windows on the digital television screen is not needed. Instead of X-windows, we use a text-based frame buffer console. This important feature is supported by Linux Kernel 2.2.x.

The main idea behind the frame buffer console is to provide a hardware-independent console device. The frame buffer device provides an abstraction for the graphics hardware. It represents the frame buffer of some video hardware and allows application software to access the graphics hardware through a well-defined interface, so the software doesn't need to know anything about the low-level (e.g., hardware registers) system [15].

3.2 Set-top Box Java Virtual Machine

Transvirtual Technologies, Inc. offers the first truly cross-platform Java implementation - Kaffe Java virtual machine. It is free and available as open source code. It runs on virtually any Internet appliance or embedded system [16]. Kaffe is compact, extensible, and easy to install and configure. Kaffe was developed on Unix-like systems, specifically Linux, so that it is easy to port it to Linux.

Kaffe is a full implementation of the PersonalJava 3.0 specification, including the PersonalJava Class Libraries, as well as integrated classes for handling graphics, file management, and networking. Specifically, its Abstract Window Toolkit (AWT) 1.1 implementation supports for non-windows platforms (e.g., frame buffer console device) using its own Java based windowing system. Customizable “look and feel” has no memory and execution costs of Sun’s swing.

Kaffe’s Java virtual machine includes core virtual machine, native operating system thread, garbage collection, class loaders, interpreter, verifier, Just-In-Time (JIT) compiler, standard Java Native Interface (JNI) interface to native code, network, etc. Kaffe also has the minimum footprint for Java virtual machine, which can be as little as 54 KBytes. The minimum core libraries are 138 KBytes. Diskless operations do not need native file systems. Classes can be loaded from RAM, ROM, or flash memory [16].

3.3 Application Manager

All the applications (i.e., resident and downloadable Xlets) need an application manager to launch the code, control its execution, and collect garbage (i.e., manage the lifecycle of the applications and signaling). Each application cannot run by itself without the application manager.

The application manager is by definition a part of the system software and resident in the set-top box. It is responsible for checking the code and data integrity, synchronizing the commands and information, adapting the presentation graphic format to suit the platform display, obtaining and disposing of the system resources, managing the error signaling and exceptions, initiating and terminating any new sessions, and allowing the sharing of variables and contents [4]. In addition, the application manager must maintain a remote control key event model to manage the button events [18].

Figure 6 shows the interface between Xlets and the set-top box resources [18]. The set-top box resources include shared memory, video plane, On Screen Display (OSD) plane, properties, etc. Non-resident Xlet code can be loaded into set-top box memory from data carousel at run time. The signaling information of an Xlet is carried in the Application Information Table (AIT), which can be identified from the broadcasting transport streams or from the network (IP).

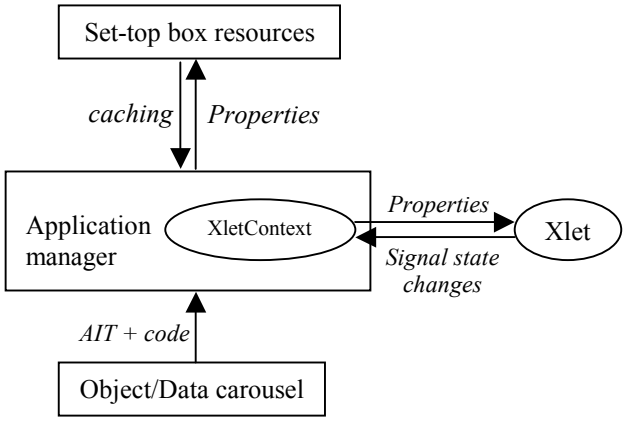

Figure 6. The interface between Xlets and set-top box environment.

The application manager and an Xlet implement XletContext and Xlet interfaces, respectively. The interfaces define the protocols of behavior that can be implemented by any applications, anywhere. The interfaces define a set of methods, but do not implement them. The application manager and Xlet application that implement the interfaces agree to certain behavior. In addition, the XletContext acts as a bridge between Xlet and the application manager so that an Xlet can access system resources and notify the state changes to the application manager during its execution. The application manager is also able to control the state changes of Xlet and terminate an Xlet at any time.

3.4 Service Delivery and Control

The applications include resident and downloadable software. They were implemented using Java language.

The Navigator and Digital Teletext service are the resident applications. Thus, they belong to the system software and were stored in the flash memory of set-top box. They use the Java APIs, and thus, they are in the same level as the APIs (cf. Figure 5). The resident applications are updated via a hardware module-bootloader of the set-top box by means of the Navigator.

A downloadable application is multiplexed with the television program and transmitted using so called Digital Storage Media -Command and Control (DSM-CC) object carousel, which send the code and data as a repeating cycle. The objects are automatically downloaded into the digital television receiver (i.e., set-top box), which then runs them. DSM-CC UU (User to User) is the interface that is used to extract DSM-CC carousel objects from the broadcast streams, or via an interactive access to a remote server if there is a return channel.

The Navigator and Digital Teletext service can be initiated via remote control. Usually, there are buttons in a remote control, which corresponds to the two functions. Also, there is a button on the remote control to be used to launch the downloadable applications (i.e., interactive programs).

4. IMPLEMENTATION

Figure 7. Testing system.

4.1 Hardware and Software Environment

The set-top box hardware for system testing is an all-in-one multimedia Pentium processor-based single board computer (SBC) - PCM-5864. It uses K6-2 processor. The CPU speed is 266 MHz. It supports a 36-bits LCD flat panel, 32-bit PCI-bus 100/10 Mbps Ethernet interface, video in and TV-out (supports NTSC and PAL formats), and integrated 3D surround audio.

A programmable watchdog timer is used as an internal timer of the set-top box. The configurations also include 32 Mbytes RAM, 2 Mbytes standard display RAM, and a Compact Flash card with 16 Mbytes memory. The display resolution was set to 640 x 480 mode [19]. Figure 7 shows the screenshot of the testing system.

The Debian Linux 2.2 with Linux kernel 2.2.4, Kaffe Java virtual machine 1.05, and XML parser were installed in the hard disk. Since Kaffe does not support Xerces parser (by Apache) that implements Sun's Java API for XML Parsing (JAXP) interface, xp (by James Clark) parser with Docuverse Document Object Model (DOM) was used. The VESA 2.0 was used as the frame buffer console driver. The Java bytecodes and data of Navigator, Digital Teletext, application manager, and drivers were also stored in the hard disk.

The applications (i.e., Navigator, Digital Teletext, and Ice Hockey) were originally developed in Windows 98 and JDK1.2 environment. In a later stage, the code was immigrated to Debian Linux 2.2 and Kaffe 1.05 environment. Finally, the Java bytecodes of Navigator and Digital Teletext were ported to the set-top box environment. The Ice Hockey Java bytecodes were placed on the server and loaded via HTTP protocol. The Java Media Frame Work (JMF 1.0) player was used to playback video.

4.2 Run-time System

The application manager is started up automatically after the set-top box was powered up and kept running until the set-top box was powered down. The application manager was scheduled as the lowest priority thread if no application is running.

It kept a data table in volatile-memory (RAM) during run-time. The data table maintained the signaling information of running or paused Xlets. Each record in the data table contained a class loader, Xlet object, current state, and initial class of the running or paused Xlet.

Whenever an Xlet changed its state, the application manager had to be notified about the state change so that it could track the state of the Xlet. The application manager could not force an Xlet to provide its service. Downloadable Xlets could only be activated via broadcasting and viewer' request. Xlets could be terminated at any time. There is no limit on simultaneous applications running in the set-top box memory as long as there is enough memory.

A system configuration file was used to record the states and configurations of Xlets, initialize the system, keep shared resources, save user's profiles, tuning information, etc. The data information of resident Xlets was got, when it was downloaded from the bootloader. The application manager was responsible for getting the AIT table information from the data carousel for the interactive services. In the testing system, we used a local binary (bit

stream) file instead of transport streams sending from data carousels.

This configuration file was kept in cache during the system running. There are several shared variables for each application's running state, which are used by Xlets to notify the application manager about their state changes.

Xlets accessed the resources and shared variables (e.g., video plane, graphics plane, system properties, etc.) provided by the set-top box via the XletContext interface passed by the application manager. The application manager also completed the adaptation of platform display and remote control key event operation, e.g., resize the video size, add and remove the key listeners for the Xlets.

The Navigator was initiated by pressing "*Navig*" button on remote control. If the viewer wanted to leave the Navigator to select the Digital Teletext service by pressing "*text-TV*" button, the application manager did the garbage collection for the Navigator and loaded the Digital Teletext code without starting a new Java virtual machine.

Similarly, the Ice Hockey application was started when viewer pressed the "*App*" button on the remote control. The Java bytecodes were downloaded from the Internet by reading the AIT table, and at the same time the application manager wrote the application state in the system configuration file dynamically.

5. RESULTS

Some performance results from the system execution are shown in the Table 1, Table 2, Table 3, and Table 4. The total resident storage of resident applications and system software except for interactive program (Ice Hockey) is 14727 KB (cf. Table 1). It is possible to port them to the Flash memory (16 MB).

Software	Storage (KB)
Linux	12000
Kaffe Java virtual machine	411
Kaffe APIs	1075
XML parser APIs	713
JMF APIs	643
Application manager	17
Navigator	121
Digital Teletext	81
Ice Hockey	77
Total	14727

Table 1. Storage of resident system software and applications.

The memory consumption of Kaffe Java virtual machine, The JMF video player, and three applications is shown in Table 2. The total memory consumption is 12200 KB (nearly 12 MB). The three applications, which were simultaneously together with the application manager, consume about 4720 KB. It is possible for set-top box with minimum memory configuration (i.e., 16MB) (cf. Section one) to run applications simultaneously.

Software	RAM (KB)
Kaffe Java virtual machine	2360
JMF video player	5120
Application manager	712
Navigator	2360
Digital Teletext	1336
Ice Hockey	312
Total	12200

Table 2. Memory consumption of resident system software and applications at run time.

The start-up latencies of applications are shown in Table3. The results are acceptable. The test system used JMF video player as TV program instead of video decoder in real broadcasting environment. Therefore, the delay can be improved in the future.

Applications	Start-up delay (second)
JMF video player	5.52
Application manager	0.11
Navigator	6.32
Digital Teletext	5.17
Ice Hockey	3.23

Table 3. Start-up latencies of the applications.

Table 4 shows the switching times between applications while three applications were running in the set-top box. The results shown are also acceptable.

Applications		Switching delay (second)
From	To	
Navigator	Digital Teletext	1.38
Digital Teletext	Navigator	2.25
Navigator	Ice Hockey	0.84
Ice Hockey	Navigator	1.85
Digital Teletext	Ice Hockey	0.93
Ice Hockey	Digital Teletext	1.57

Table 4. Switching latencies between applications.

6. CONCLUSIONS

This paper presented a system integration model and a service architecture for future set-top box. In particular, a working prototype system and services were designed and developed. The system components consisted of Linux

operating system with selected modules, necessary drivers, Kaffe Java virtual machine, and minimum Java classes (APIs) including XML parser, an application manager, and two resident applications.

The work has shown that it is ideal to execute the applications to ensure the cross-platform interoperability based on this system architecture and the selected system components. And also the applications developed, which were based on the service architecture, can be well managed by the system software – application manager in the set-top box.

The configurations of the hardware and software environment of the set-top box are compatible with standard set-top boxes. The results in Section 5 are acceptable and satisfactory. Our work can be a valuable basis of the vendors of future set-top box.

We shall improve the system performance both from hardware and software resources, e.g., further removing unnecessary components of Linux operating system, booting Linux operating system from Flash memory using Compact Flash card, storing part of system components and configurations in Flash memory in stead of hard disk, analysis of caching part of Digital Teletext pages in memory, etc.

We shall also set-up a server that is used to simulate the data carousel mechanisms to delivery applications, SI, and AIT information to the set-top box. Further more, the more important one is to add fault-tolerant function to the system to ensure the reliability. Finally, performance according to the criteria and implementation guidelines of the standards shall be measured.

7. ACKNOWLEDGMENT

The Future TV project (http://futuretv.uta.fi) is funded by the National Technology Agency of Finland together with major Finnish television, telecommunications, and digital media companies. The author Chengyuan Peng would like to thank Nokia Oyj foundation for the support during the research work.

8. REFERENCES

[1] B. Fox, "Digital TV comes down to earth," *IEEE Spectrum*, October 1998.

[2] M. Milenkovic, "Delivering interactive services via a digital television infrastructure," *IEEE Multimedia*, vol. 5, no.4, Oct./Dec. 1998, pp. 34-43.

[3] J. - P. Evain, "The Multimedia Home Platform - an overview," *EBU Technical Review*, spring 1998.

[4] Gerard O'Driscoll, "*The essential Guide to Digital Set-top Boxes and Interactive TV*," Prentice Hall PTR, 2000.

[5] NorDig II, "*Digital Integrated Receiver Decoder Specification, for use in cable, satellite and terrestrial networks*," Version 0.9.

[6] DVB-MHP, "Multimedia Home Platform," *European Broadcasting Union*, February 2000.

[7] T. Lindholm, F. Yellin, The Java Virtual Machine Specification, *Sun Microsystems, Inc.*, 1999.

[8] P. Vuorimaa, "Digital television service architecture," *Proceedings of IEEE International Conference on Multimedia and Expo, ICME2000*, New York City, NY, USA, July 30 - Aug. 2, 2000, pp. 1411-1414.

[9] ETSI ETS 300 706, "Enhanced teletext specification," *European Telecommunication Standardization Institute (ETSI)*, 1997.

[10] ETSI TR 101 200, "Digital video broadcasting (DVB) – a guideline for the use of the DVB specifications and standards," *European Telecommunication Standardization Institute (ETSI)*, 1997.

[11] ETSI EN 300 468, "Digital Video Broadcasting (DVB; Specification for Service Information (SI) in DVB Systems," *European Telecommunication Standardization Institute (ETSI)*, 1998.

[12] P. Vuorimaa and C. Sancho, "XML based text TV," *Web Information Systems Engineering*, *WISE2000*, Hong Kong, June 19-20, 2000.

[13] C. Peng and P. Vuorimaa, "A Digital Teletext Service," *Proceedings of the 9th Int. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision*, *WSCG'2001*, Czech Republic, February 5 - 9, 2001, pp. 120-125.

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

[15] L. Ayers, "Kernel 2.2's Frame-buffer Option," *Linux Journal*, issue 36, January 1999.

[16] Transvirtual, "Wherever you want to run Java, Kaffe is there," *white paper*, 1999.

[17] R. Lehrbaum, "Using Linux in Embedded and Real-Time Systems," *Embedded Linux Journal*, issue 75, 2000.

[18] C. Peng, P. Vuorimaa, "Digital Television Application Manager," accepted to *the IEEE International Conference on Multimedia and Expo*, Tokyo, Japan, August 22-25, 2001.

[19] G. Sivaraman, P. Cesar, and P. Vuorimaa, "System software for digital television applications," accepted to *the IEEE International Conference on Multimedia and Expo*, Tokyo, Japan, August 22-25, 2001.