

SEMI-AUTOMATED MOBILE TELEVISION
INTERACTIVE APPLICATION GENERATION
BASED ON XHTML AND JAVA ME

By Moxian Liu

A Thesis Submitted for the Degree of Doctor of Philosophy

School of Engineering and Design, Brunel University

December 2010

I. Acknowledgements

There are many people who gave me advice, guidance and support throughout these years of my Ph.D. study. First and foremost I would like to thank my supervisor and mentor Dr. Emmanuel Tseklevs who apart from his duties as a supervisor, in consulting, guiding, encouraging and reviewing my work with all his efforts and patience, introduced me to the wonderful world of Digital TV and believed in my potentials.

Second I would like to thank my second supervisor Prof. John Cosmas for his kindly encouragement, experienced advice and unselfish support during these years. I would also like to thank Dr. Mozhen Li, Senior Lecturer of School of Engineering and Design, who gave me suggestions and advices during my research study.

In addition, I would like to thank two of my friends Dr. Mingxu Xia and Dr. Yang Liu for their kindly help and support during these years.

Finally, I would like to thank my parents and my wife for their encouragement, understanding and endless patient, unconditional love and support all the way through this difficult journey.

Moxian Liu
Uxbridge, UK
July 2010

II. Table of Content

I.	Acknowledgements	2
II.	Table of Content	3
III.	List for Publications.....	6
IV.	Abstract.....	7
1.	Chapter 1: Introduction	8
1.1	Research Scope	8
1.2	Analogue, Digital and Mobile Television.....	9
1.2.1	Analogue Television – A series of TV programmes	9
1.2.2	Digital Television – A series of bidirectional TV services	10
1.2.3	Mobile Digital Television – Enriched DTV services anywhere	11
1.3	DTV and MDTV Service Market Overview	13
1.3.1	DTV services in the market	13
1.3.2	MDTV services in the market.....	14
1.3.3	The MDTV service development status	15
1.4	Motivation.....	16
1.5	Challenges and Aims	19
1.6	Thesis Organization	20
2.	Chapter 2: MDTV service Creation and Implementation	23
2.1	A Glance on DTV and MDTV Global Standards	23
2.2	A Generic DTV Service Asset Lifecycle.....	24
2.3	DTV Service Consumption	27
2.3.1	DTV service end-user terminal architecture	28
2.3.2	Middleware Introduction	29
2.4	Global DTV Standards Middleware Solutions	31
2.4.1	Stationary DTV service middleware solutions	31
2.4.1.1	<i>DVB Multimedia Home Platform and Globally Executable MHP.....</i>	<i>31</i>
2.4.1.2	<i>Other widely adopted middleware solutions.....</i>	<i>34</i>
2.4.2	Mobile DTV service middleware solutions	36
2.4.2.1	<i>Looking forward to the Open Middleware Solution – JSR 272.....</i>	<i>36</i>
2.4.2.2	<i>Standard-specific middleware solutions.....</i>	<i>39</i>
2.4.2.3	<i>Middleware-like solution in 3G.....</i>	<i>43</i>
2.4.2.4	<i>Solution for MDTV service over DVB network</i>	<i>45</i>
2.5	MDTV Service Creation and Implementation on the Application Layer	47
2.5.1	Standard-based solution.....	48
2.5.2	Commercial solution.....	49
2.5.2.1	<i>Middleware-oriented solution.....</i>	<i>50</i>
2.5.2.2	<i>Integrated software system solution.....</i>	<i>51</i>
2.5.2.3	<i>Limitation of the commercial solutions</i>	<i>56</i>
2.5.3	Looking for the universal solution.....	56
2.6	Software Engineering Fundamental.....	59
2.6.1	Software testing	61
2.6.1.1	<i>Test level.....</i>	<i>62</i>

2.6.1.2	Testing method	62
2.6.1.3	Testing case design.....	63
2.7	Introduction of proposed methodology	64
2.7.1	Reference model for MDTV service creation and implementation (MDTV-SIM).....	64
2.7.2	Software process model for MDTV service (MDTV-SPM)	66
2.7.3	Introduction of the Proposed Methodology	67
3.	Chapter 3: Proposed semi-automatic MDTV service generation.....	72
3.1	Motivation.....	72
3.2	MDTV service Presentation disscussion.....	75
3.2.1	MDTV Application Model	75
3.2.2	MDTV service presentation.....	80
3.2.2.1	Visual presentation technology candidates and discussion.....	81
3.2.2.2	Interaction presentation technology candidates and discussion.....	85
3.3	Semi-automatic MDTV service creation	86
3.3.1	Semi-automatic service creation tool.....	86
3.3.1.1	XHTML and Java ME based MDTV service presentation	91
3.3.1.2	Semi-automatic service creation software architecture.....	94
3.3.2	Semi-automatic MDTV service creation process	99
3.3.2.1	Data flow through the semi-automatic service creation tool	99
3.3.2.2	Comparison and discussion	102
4.	Chapter 4: Proposed MDTV Client Implementation Environment Based on Java ME	106
4.1	Motivation.....	106
4.2	MDTV Terminal Service Platform Discussion.....	107
4.2.1	Select the browser-based service platform structure.....	107
4.2.2	Service platform development and discussion.....	110
4.2.2.1	XHTML based MDTV service rendering discussion	111
4.2.2.2	Interactive application rendering and handling.....	114
4.3	MDTV Server in The MDTV-CIE	115
4.4	MDTV-CIE System Architecture	116
4.4.1	Proposed terminal device service platform in MDTV-CIE.....	119
4.4.1.1	The Browser MIDlet.....	120
4.4.1.2	The URL Input.....	121
4.4.1.3	The Rendering and Retrieving Block	122
4.4.1.4	The Parsing Block.....	125
4.4.1.5	The ID Event Manager.....	127
4.4.1.6	The Data Exchange Manager	128
4.4.2	The Interactive service managing server in MDTV-CIE	129
4.4.2.1	The Data Exchange Server.....	130
4.4.3	Service implementation data flow in MDTV-CIE	131
4.4.4	Test the proposed service platform in Nokia N95.....	134
4.4.5	Comparison and discussion	136
5.	Chapter 5: Test and Evaluation	139
5.1	Introduction and Motivation.....	139

5.2	Testing methodology.....	140
5.2.1	Test and emulation environment.....	141
5.2.2	Test case design of semi-automatic service creation tool	142
5.2.3	MDTV service terminal software platform testing	155
5.2.4	Integration testing of the service creation tool and the terminal platform	165
5.3	Testing Execution of the Semi-automatic Service Creation Tool.....	169
5.3.1	Test Preparation	169
5.3.2	Test Result	169
5.3.3	Test result evaluation	177
5.4	Testing Execution of the MDTV Service Terminal Software Platform	178
5.4.1	Testing Preparation	178
5.4.2	Test results	178
5.4.3	Test result evaluation	187
5.5	Integration Testing of the Service Creation Tool and Service Platform	188
5.6	Acceptance Testing	189
5.6.1	Alpha testing preparation.....	189
5.6.2	Results and evaluations	191
5.6.2.1	<i>Demographics</i>	<i>191</i>
5.6.2.2	<i>Test result evaluation of Task 1</i>	<i>191</i>
5.6.2.3	<i>Suggestions to the proposed semi-automatic service creation tool after Task 1</i>	<i>193</i>
5.6.2.4	<i>Testing result evaluation of Task 2</i>	<i>193</i>
5.6.2.5	<i>Suggestions to the proposed MDTV service platform after Task 2.....</i>	<i>195</i>
5.7	Comparison with INSTINCT service creation process	195
5.8	Test Conclusion.....	199
6.	Chapter 6: Further work and conclusion	201
6.1	Further Work.....	201
6.1.1	Semi-automatic service creation tool.....	201
6.1.2	The MDTV service platform	203
6.1.3	Further software testing procedure	206
6.2	Thesis Summary	206
6.3	Lessons Learned & Conclusion.....	208
V.	References	212
VI.	Appendix 1: Consent Form	219
VII.	Appendix 2: Participant Document.....	221
VIII.	Appendix 3: Acceptance Test Result	226

III. List for Publications

Published

Moxian Liu, “A DVB-H Service Client side Browser Interface Based on J2ME”, *SED Research Student Conference 2008*, London, 2008, pp.31-33.

Moxian Liu, E. Teskleves, J.P.Cosmas, “Semi-automatic creation of graphically-rich mobile Television services and applications using an XHTML browser and J2ME”, in *2010 IEEE International Symposium on Broadband Multimedia System and Broadcasting (BMSB)*, Shanghai, 2010, pp.1-7.

Submitted

Moxian Liu, E. Teskleves, J.P.Cosmas, ”XHTML and Java ME based Semi-automatic creation of Mobile TV service”, *IEEE Transactions on Broadcasting*, submitted.

Moxian Liu, E. Teskleves, J.P.Cosmas, ”Prototype Multimedia Service Creation Tool and Execution Platform”, *Elsevier Entertainment Computing*, to be submitted.

Moxian Liu, E. Teskleves, J.P.Cosmas, ”A Prototype Software Platform for Digital TV Multimedia Service”, *Springer Multimedia Tools and Applications*, to be submitted.

IV. Abstract

Mobile Digital TV (MDTV), the hybrid of Digital Television (DTV) and mobile devices (such as mobile phones), has introduced a new way for people to watch DTV and has brought new opportunities for development in the DTV industry. Nowadays, the development of the next generation MDTV service has progressed in terms of both hardware layers and software, with interactive services/applications becoming one of the future MDTV service trends. However, current MDTV interactive services still lack in terms of attracting the consumers and the service creation and implementation process relies too much on commercial solutions, resulting in most parts of the process being proprietary. In addition, this has increased the technical demands for developers as well as has increased substantially the cost of producing and maintaining MDTV services. In light of the aforementioned situation, the Thesis has contributed to this field, by proposing an innovative MDTV service creation and consumption system based on XHTML and Java ME. On the head-end it introduces a semi-automatic creation mechanism to facilitate a less technical and more efficient interactive service creation process. This enables designers and creative individuals to be actively involved in the MDTV service creation process and to develop interactive-rich MDTV service. On the client-end it employs an open-source software environment as the interactive service MDTV consumption platform, rendering the MDTV service implementation process as less proprietary as possible. Furthermore, the Thesis offers a discussion on the different MDTV interactive application models currently used and based on the proposed software, a novel MDTV service presentation method is further introduced and adopted instead of the Rich Media and ECMAScript based methods. Finally, a series of qualitative testing procedures have been implemented with regards to conducting an essential evaluation on the operability of the proposed software system.

1. CHAPTER 1: INTRODUCTION

1.1 RESEARCH SCOPE

Modern Digital TV (DTV) systems broadcast not only traditional Television (TV) content through the broadcast networks but also provide viewers with a series of interactive TV services. This improvement has so far succeeded in encouraging people to change from simply “watching” the content offered in their TV sets to “interacting” with it. Moreover, the advent of Mobile Digital TV (MDTV) has been realized and implemented today, due to the popularization of personal mobile devices such as mobile phones, PDAs (Personal Digital Assistants) or even notebooks and netbooks.

In order to ensure that the users of MDTV are able to experience the same services as those on the conventional DTV, substantial research work has been conducted in the area, a few resulting in some approaches to be proposed and placed into practice. In short, the current MDTV related work and solutions mainly focus on shifting existing DTV services to MDTV, even though the characteristics between DTV and MDTV systems vary considerably. More precisely, research has been conducted in recent years on constructing the service distribution (broadcasting) and reception environments especially for MDTV, with respect to the mobile service operation features from both hardware and software perspectives. The development results on hardware environment and lower layer protocols are promising with two main reliable solutions (broadcast networks and mobile convergence networks) being formed, whilst the implementation and adoption of higher layer standards and specifications are running relatively behind. Discussions and attempts on introducing interactive-rich services into the MDTV domain and in particular focusing on the creation, display and interaction aspect of these services are currently ongoing.

The research in this Thesis focuses mainly on developing an executable solution for

MDTV interactive service creation and consumption by proposing a new MDTV service creation and consumption environment as well as its associated software tools.

1.2 ANALOGUE, DIGITAL AND MOBILE TELEVISION

Broadcast multimedia is a common denominator for networked multimedia platforms that involve the use of a unidirectional broadcast channel to convey high-speed and high-quality audio-visual services to consumers. The most common example of broadcast multimedia is that of the television [2]. Switching from Analogue Television (ATV) to DTV, and to MDTV, has brought new elements, challenges and evolution on both technologies and markets. The following sections provide a brief overview of the key historical milestones of the TV medium.

1.2.1 ANALOGUE TELEVISION – A SERIES OF TV PROGRAMMES



Figure 1.1: Analog Television service [<http://11takes.blogspot.com/2007/12/free-digital-tv-converter.html>, 6/12/2007]

Starting from the first conventional Analogue Television (ATV) service broadcasted by the British Broadcasting Corporation (BBC) in the UK in 1936, broadcasters began to distribute the initial form of TV services by providing a series of TV programmes through different signal channels, using analogue signals. People consumed the TV services by watching these programmes and often switching between them. As the time went by, ATV service was improved significantly by the introduction of a variety of TV programmes that brought massive amount of information to the users such as

premade recordings (e.g.: movies), live shows, news reports and even some simple auxiliary interactive applications. So far ATV has helped to establish a passive visual information environment and has become the predominant consumer medium for the majority of the population and has become a vehicle in exploring the world around us.

1.2.2 DIGITAL TELEVISION – A SERIES OF BIDIRECTIONAL TV SERVICES



Figure 1.2: Sky Digital TV service [<http://stuff.tv/blogs/future/archive/2008/05/29/first-look-sky-s-new-hd-epg.aspx>, by Mark Wilson, 29/5/2008]

Digital Television (DTV) was adopted as the next generation of broadcast television technology in the early 1990s, based on the transmission of digitized TV signals. Since then, with the introduction of digital technology in the production, distribution, and reception of TV services, the trend of digitalization has influenced the entire TV broadcasting industry greatly in less than two decades. Digitalization has been a technological phenomenon from the standpoint of transmission capability, of efficiency of network distribution network, of image quality, and of flexibility and variety of performances, which for the very first time widen the TV set's field of use well beyond the programmes' traditional fruition [4]. TV sets thus have become a new portal for the electronically mediated information exchange. Above all, the novelty of DTV has improved the relationship between the audiences and the broadcasters as well as the nature of the medium and its production processes. The audience now becomes the user, starting to become more active within the TV service environment by accessing a series of interactive applications and other auxiliary contents. This

resulted in an evolution of the TV from a passive medium towards a fully interactive environment supporting a variety of innovative service schemes. As a result of this, TV broadcasting has established a new range of conceptual series of highly customizable integrated interactive multimedia DTV services, in contrast to the ATV's fixed and linearly scheduled TV services. Figure 1.2 illustrates a typical DTV service user interface, where various TV programmes and interactive applications (Electronic Program Guide for example) can be recognized.

1.2.3 MOBILE DIGITAL TELEVISION – ENRICHED DTV SERVICES ANYWHERE



Figure 1.3: Mobile Digital TV service [<http://www.persian-forums.com/f81/dvb-h-esg-simulator-beta-4-a-20162/> 4/8/2009]

MIPCOM, one of the most famous media content events held in Cannes every year, described Mobile TV in October 2006 as the most significant wireless trend for the mobile industry in the coming years. After a successful adoption on the live broadcast of FIFA 2006 World Cup in Germany, Mobile Digital TV (MDTV) has become a promising reality for the world market. Several leading organizations of telecommunication industries, such as the ITU, ETSI and 3G Partnership forum, have contributed several recommendations on MDTV, which resulted in the rapid implementation of other related components in the mobile industry, such as handsets, chip sets, spectrum, operating system, software applications, transmission technologies and MDTV service content.

Mobile TV is defined as the transmission of TV programmes or video for a range of wireless devices ranging from mobile TV-capable phones to PDAs and wireless multimedia devices [2]. Currently within the broadcast TV domain, there are two different modes of mobile TV, based on the different formats of the transmission signal; namely the Mobile Analog TV (MATV) and the Mobile Digital TV (MDTV). MATV is a mobile/portable ATV service that enables users to watch TV programmes by receiving the traditional analogue TV signal through an ATV system using mobile devices with ATV receiver chipsets built-in. MDTV offers enhanced DTV services to MDTV-capable devices through digitized broadcast network or telecommunication network system. Just like DTV has several advantages over ATV, equally MDTV offers substantial advantages over MATV, with the digitalization of TV broadcasting system. MDTV, compared to MATV is a more interactive, highly customizable and offers multimedia-rich services that provide the user with a higher level of mobile TV service experience.

Among the components in a MDTV system, the terminal device is the most distinguishable feature of MDTV comparing with other forms of DTV. With the “mobilization” trend of most of the personal electronic devices nowadays such as mobile phones, PDAs and game consoles, there are more and more mobile devices being adopted as the MDTV service terminal. Moreover in recent years, researchers as well as manufacturers have placed great effort in developing mobile devices with more functions but smaller chip sets, more powerful processing abilities but less power consumption. Meanwhile the network operators have also started adjusting and updating current networks (or have even been building up entirely new networks) to ensure that the terminal devices are capable of receiving mobile TV services. All these developments further encourage the popularization of MDTV and based on a forecast in the “Global Mobile TV Forecast to 2013”, the number of the mobile TV subscribers worldwide will grow to reach 570 million by the end of 2013 [24].

MDTV derives from DTV but constitutes a new information portal on a mobile device.

On one hand, its portability is higher than the conventional DTV, offering wide service coverage and better reception ability of MDTV signals at a high speeds, when compared to traditional DTV networks. Furthermore, when incorporated on a mobile phone device, MDTV is able to use the cellular networks for the provision of an interaction path that can offer a wider variety of multimedia services. If the conventional DTV's most attractive point is the high audio-visual quality experience, then MDTV greatest advantage is the excellent mobility and integration of multimedia in a single portable platform.

1.3 DTV AND MDTV SERVICE MARKET OVERVIEW

1.3.1 DTV SERVICES IN THE MARKET

DTV services have become available in many countries worldwide. The initial steps of DTV service implementation were aimed at repurposing ATV programmes in the digital domain. Therefore the DTV value-added services were realized as the next milestone in the DTV evolution. Apart from the traditional types of services found in the ATV, such as TV advertising, subtitles and conditional access (CA), DTV has set up a series of services rich in interactive functions, enabling users to interact with the DTV content or even with other users through a telecommunication-based/internet-based return channel. A basic list of different categories of interactive DTV services currently in practice is shown in Table 1.1:

Category	Interactive DTV Services	
TV-specific services	Electronic Program Guide (EPG)	
	Pay-per-View (PpV)	
	Video-on-Demand (VoD)	
	Personal Video Recorder (PVR)	
Value-added services	Information Portal	Government
		Health
		Weather information
		Community services
	Communication	Instant message
		E-mail
	E-commerce	TV-shopping
		Interactive advertising
	Finance	TV banking
	Game	Simple/Multi-player Game
Voting and betting		

Table 1.1: Interactive DTV services

1.3.2 MDTV SERVICES IN THE MARKET

With the development of modern telecommunication technology, especially after the implementation of the third generation mobile telecommunication technology (3G) and the mobile broadcasting technology, the transmission of digital audio/visual media to mobile devices became possible. MDTV was launched as a commercial service in 2004 [20] and since then it has been experiencing an initial implementation phase with the aim of broadcasting a cut-down version of DTV service on the MDTV system platform. Meanwhile the development of the MDTV technologies never stopped and trials have been carried out for pilot study purpose in many countries. Several MDTV standards have so far been published and adopted and the MDTV service has become popular and known as a new form of DTV.

Table 1.2 shows the MDTV services deployed in several countries of different regions in the world: in Asia, South Korea and Japan are at the forefront of MDTV development and basic services such as Free-to-air were launched since 2005; Italy have been the lead on MDTV service in Europe by using 3G network whilst in the UK, several trails based on DVB-H and DAB/DMB have been launched with regard to collect practical experience for the actual deployment; the USA has launched their MDTV service via several popular technologies (as shown in Table 1.2) and the service deployment has been well-operated so far with the supports from many famous content providers (ESPN, NBC, FOX, etc) as well as network providers (AT&T, Alltel, etc). [7], [9], [12], [17] - [23]

	Asia		Europe		North America
	South Korea	Japan	Italy	UK	USA
Bearer Technology	S-DMB, T-DMB, 3G	ISDB-T 1seg, 3G	3G	3G, DVB-H, DAB/DMB	3G, MediaFLO, ATSC-M/H, DVB-H
Typical Service Model	Free-to-air, Information service (e.g. news, weather), ESG.		Free-to-air, PayTV, PpV	PayTV	Free-to-air, VoD, PayTV, PpV

Table 1.2: World MDTV services

1.3.3 THE MDTV SERVICE DEVELOPMENT STATUS

Based on a brief overview on the global MDTV service market, it has become clear that:

1. Currently there are two groups of transmission technologies being used for distributing MDTV services: namely mobile telecommunication technologies (e.g. 3G) that support Unicast mode (pier-to-pier delivery), Multicast mode (pier to multiple piers delivery) and Broadcast mode (pier-to-coverage delivery), and digital broadcast technologies (e.g. DVB-H and ATSC-M/H). These different technologies will be discussed and explained in more detail in Chapter 2.
2. Due to a number of MDTV services and trials deployed in recent years, MDTV has become a common service for several mobile device (e.g. mobile phone) users.
3. Many of the current MDTV services in Europe and North America are carried by 3G networks whilst the broadcast technologies have not yet been adopted fully with a few of the services and trials succeeded; In contrast, MDTV over the digital broadcast network has been adopted well in Japan and South Korea.
4. Free-to-air MDTV services have not yet been widely applied in Europe or North America, which is one of the reasons why the MDTV has not become that popular in these regions when compared to Japan and South Korea;
5. Current MDTV service types are mainly free-to-air, PayTV and VoD. Comparing with the various service types of conventional DTV, MDTV services are still limited and lack of attracting large numbers of subscribers. Several additional MDTV services are also required, such as:
 - a. *Information portals: weather, news, stock and etc.;*
 - b. *Multiplayer gaming;*
 - c. *Voting applications;*
 - d. *Online lotteries and Gambling;*
6. There is no universal MDTV service implementation solution being proposed or adopted globally so far and equally importantly is no open-standard based MDTV

service design solution has been implemented.

Consequently, the current situation in the MDTV service market is very interesting since MDTV is now well-known but not very popular amongst users and consumers. More precisely MDTV has been widely adopted as one a common mobile device service although it is currently under the initial implementation phase, since several of the fundamental technologies (specifications, protocols, hardware, middleware and software, etc.) are still being developed, improved and deployed. The development and design of MDTV services and its content and applications are still rather poor in terms of quality and quantity.

The reason for the current situation is considered to be mainly due to the fact that even though MDTV has been implemented for several years, it is still under a development phase, with several problems arising, such as the rising demand to the telecommunication resources (such as spectrum). Markets in different regions have presented their own MDTV systems and services however most of them have not yet been fully developed with several pilot studies still being conducted. Moreover time is required to achieve further evolvement of the technology at a certain level that would contribute to the development of the entire MDTV industry and corresponding business models.

1.4 MOTIVATION

Following the discussion above it is clear that there are challenges and opportunities for researchers in the MDTV field for further development and innovation. One of the key areas and challenge identified in this area is that current MDTV services lack in terms of attractive and fully interactive applications, which has in turn provided us with the motivation for this research project. The current developments as far as MDTV services are concerned are mainly aiming their efforts in adapting existing DTV services to the MDTV domain and so far only a small group of TV programmed based services (free-to-air, PpV, VoD as well as basic information services) have been

adopted when compared to the current DTV services listed in Table 1.1. Moreover, most of the current MDTV services and applications are unidirectional. This situation is in contradiction to one of the key advantages of MDTV: 1) Conventional DTV services used to be unidirectional on the early stage of their release and the service finally became interactive with the addition of telecommunication wired networks; 2) As far as MDTV, mobile communication networks is a key and given benefit of most of MDTV terminal devices (e.g. mobile phones), making it in turn the best platform for the adoption of interactive MDTV services. Therefore, with a native and integrated bidirectional network, MDTV ought to provide users with more interactive applications than DTV. Unfortunately, it is the opposite case with MDTV currently offering a rather limited set of interactive options for users.

More importantly, MDTV should develop interactive services and applications especially according to its own characteristics and in turn develop a MDTV specific interactive service system. However, as the expected mobile multimedia gateway of the future, MDTV is still in lack of a promising interactive service model with integrated functions and reliable quality. Several aspects need to be considered as the reasons of this situation, as follow:

1. Comparing with conventional DTV, MDTV has different hardware and software environments. Since most of the MDTV terminal devices have limited battery supply, most of the hardware parts such as CPU are designed to save more power, which results in a more limited processing/performance ability in terms of hardware (e.g. small size of screen). Thus the software technologies used in a MDTV terminal device are usually developed specially for power limited devices with limited functions being supported. Such environments have restricted the direct adoption and adaptation of DTV interactive services and applications in the MDTV domain, with most of them needed to be redesigned to fit the unique characteristics of this environment;
2. Another related issue is that of many of the software development technologies for power limited device are relatively hard to use with much less functions support

comparing with their full version. It becomes even worse when developers try to use these specific mobile targeted software environments to develop MDTV services, in which case a very specific MDTV software layer knowledge is required and developers have to be trained extensively before they start the development work.

3. In terms of the service creation process this too demands designers and creative individuals to also acquire technical knowledge on the software platform that a service is designed for. This means that designers cannot be involved directly in the service creation process as they do not have the software skills to develop the graphical user interfaces, look and feel and interactive features of a service. In turn, this advanced knowledge demand has therefore increased the time and resource expense and moreover prevented designers and professional creators from being involved into the process of the MDTV interactive service/application design and creation;

4. Also regarding the software development technologies currently used in the MDTV field, most of them have poor compatibility with each other (e.g. LAsER and Java ME). The MDTV technology development entities have realized this problem and have already standardized the MDTV system's application layer according to the underlying transmission standards (e.g. DVB-H, ATSC-M/H). However, even though unification has been achieved within the scope where the same transmission standard is utilized, different MDTV middleware/software standards are still not compatible. Designers have to use different corresponding standards and technologies even when they intend to design the same applications;

5. As a result of the above situation, the current MDTV service development mainly relies on commercial solutions and many of them are able to support multiple MDTV standards. However the service creation process and implementation method of these solutions are usually proprietary with limited types of interactive service available.

As a result, various proprietary commercial solutions dominate the current MDTV

interactive service implementation field whilst the current middleware/software specifications from different MDTV standards have limited contribution on developing a universal solution or open-standard on the MDTV's application layer. This paradox remains one of the current issues limiting the speed of evolution and the richness of services, especially in terms of interactive applications within the MDTV field.

1.5 CHALLENGES AND AIMS

The main technical challenges are further highlighted from the motivation as follows:

1. How to reduce the technical knowledge demands on the designers, during the service creation process and how to facilitate such process to encourage the development of MDTV interactive-rich service?
2. How to improve the inter-compatibility of service development technologies as well as their outputs (e.g. service content/applications) under different MDTV standards?
3. How to avoid proprietary characteristics from current service creation and implementation process?

In light of these challenges, we intend to contribute to the current MDTV service development and implementation field by proposing an optimized MDTV interactive service/application creation and consumption solution. The aims of the Thesis are set as follow:

- 1) To optimize the process of MDTV service creation, demystifying and simplifying it, making it less technical and more efficient by developing a semi-automatic service creation tool within the service creation lifecycle. This enables the involvement of more design-oriented and creative professional in the service creation process.

2) To employ an open-source software environment as the platform of research and development work, to improve the MDTV service creation and consumption process by converting it from proprietary-based to a relatively open standard-based one. An open-source based service creation process could introduce more inter-compatible features. This would also enable the scalability and future extensions of the currently developed platform and its interaction with other open standards (MHP, JSR-272, etc).

1.6 THESIS ORGANIZATION

The Thesis has been organized into six chapters.

Chapter 1 offers an introduction to the research work. The research scope is briefly stated at the beginning, followed by a quick overview of the key milestones of TV's history and its service development. An overview of DTV as well as MDTV services found currently in the market and a brief discussion about the current status and the future of the MDTV services is also being offered here. Then by highlighting the technical challenges, this chapter further sets the aims of the research work. Lastly, this chapter ends up with the Thesis organization.

Chapter 2 is the chapter of the Literature review. It starts with a glance on the global DTV/MDTV standards and then is the background knowledge related to the MDTV service creation and implementation. After a brief introduction on the classical DTV service asset lifecycle, a review on global DTV middleware solutions from both international standards and commercial aspects follows, with the context and the emphasis placed on MDTV middleware. The main body of this chapter consists of an additional review and an in-depth discussion of global MDTV service creation and implementation solutions within the software engineering scope. By concluding the discussion, an introduction of thesis' methodology is briefly presented, linked to the implementation section of the Thesis.

Chapter 3 offers the first part of the implementation section by presenting the

proposed semi-automatic MDTV service creation process. The motivation behind introducing the semi-automatic service creation is firstly presented and followed by a discussion on the current MDTV service presentation technologies. The system architecture of the proposed process as well as its contribution is stated as the main body of the chapter, where the semi-automatic service creation tool is presented as the main software component. A qualitative comparison between some current solutions and the proposed work is presented lastly.

Chapter 4 offers the second part of the implementation section by presenting the proposed MDTV Client Implementation Environment including a service platform on the terminal device and a server. This chapter also contains the relevant motivation, methodology and the statement of the implementation process. Moreover in the statement of implementation, the MDTV service platform is introduced as the main software component in the proposed environment according to the software construction and the main functionalities of each part are fully presented. The data flow within the proposed environment is presented and a brief discussion on the contribution of the proposed software concludes this chapter.

Chapter 5 conducts the software testing and evaluation on the proposed software components. An introduction of the motivation and aims of this test procedure is stated at the beginning and the testing methodology is presented after that. The main body of this chapter consists of the testing conducted on the proposed software including the semi-automatic service creation and the service platform. Especially, the result of the acceptance testing procedures of the proposed software which involves 10 actual testers is presented after the conventional software testing cases. Besides, an evaluation and discussion based on the results of the testing procedures is offered at the end of each testing case.

Chapter 6 briefly introduces the possible further and future work of the project and lastly concludes the Thesis.

Summary:

Having introduced the background of MDTV and the development of its service, this chapter further discussed the situation of the current MDTV service and after targeting the main problems as well as highlighting the challenges, this chapter has set the aims of the research work with regards to making effective improvements to the current MDTV service implementation process.

2. CHAPTER 2: MDTV SERVICE CREATION AND IMPLEMENTATION

2.1 A GLANCE ON DTV AND MDTV GLOBAL STANDARDS

Since the Digital TV became widely available worldwide, the development of Digital TV has brought a new series of technical specifications for most of the components within the DTV system; ranging from the description of coding algorithms to that of modulation procedures, transmission parameters, hardware components, device interfaces, protocols and techniques for adding interactivity as well as their corresponding software platforms and applications. Many research and standardisation organizations such as the Moving Pictures Experts Group (MPEG), International Telecommunications Union (ITU), International Standardisation Organization (ISO) and many more have all played an important role in creating a unity on the concepts and developing common standards for global digital broadcasting. So far, there are three main DTV standards adopted globally, which are the Digital Video Broadcasting (DVB) from Europe, the Advanced Television Systems Committee (ATSC) from USA and the Integrated Services Digital Broadcasting (ISDB) from Japan. China has also put strong effort on developing its own standards in order to formalize and standardize the current broadcasting industry within China [30]-[35].

	DVB	ATSC	ISDB
Via Terrestrial	DVB-T(2)	ATSC-T	ISDB-T
Via Satellite	DVB-S(2)	ATSC-S	ISDB-S
Via Cable	DVB-C(2)	ATSC-C	ISDB-C
To Mobile Device	DVB-H /DVB-SH	ATSC-M/H	1seg

Table 2.1 Main DTV standards in the world with their main transmission solutions

With Regards to the different hardware and software requirements between MDTV service and conventional DTV service deployment, several aspects of DTV

technology have been modified and optimized to fit DTV services to MDTV. In addition to this a new series of MDTV standards are being developed and deployed by leading DTV service operators, 3G cellular mobile network operators and even wireless broadband operators throughout the world. As listed in Table 2.2, there are two groups of MDTV network technologies: Mobile Broadcast Network technologies and Cellular Mobile Communication Network technologies. The former technologies usually evolve from their corresponding version for conventional DTV. For example, DVB-H is inherited from DVB-T and ISDB-T 1seg is based on ISDB-T. The Cellular Mobile Communication Network technologies are developed as the cellular network is evolved and they have been improved to become more and more advanced to meet the increasing requirement of video casting service over the cellular network. In addition, DVB-IP Datacast, DMB-IP Datacast and Open Mobile Alliance Mobile Broadcast Services (OMA-BCAST) are also developed to define relative protocols and handle the services on the IP layer.

Mobile Broadcast Network technologies and standards				Transport Layer specifications
DVB-Handheld (DVB-H)	ISDB-T 1seg	Digital Multimedia Broadcast (T/S-DMB)	ATSC-Mobile/Handheld (ATSC-M/H)	DVB-IPDC DMB-IPDC OMA-BCAST
Cellular Mobile Communication Network technologies and standards				
Unicast	Multicast/Broadcast	Media Forward Link Only (MediaFLO)		

Table 2.2 Mobile Broadcast Network technologies and standards

2.2 A GENERIC DTV SERVICE ASSET LIFECYCLE

Developed on these network technologies and standards, MDTV service has now enabled users to watch a wide range of programmes when and where they wish as well as offers a personalized and interactive experience. So far and since its first commercial release, MDTV has become an attractive novelty within the mobile phone environment and has been initially accepted by the market (see the market study in Chapter 1). However in the vertical market of MDTV services, improving the efficiency and reliability of the entire MDTV service production workflow and system,

so as to get service of higher quality, is still one of the key priorities and main issues worth considering. By considering this issue, it is firstly necessary to familiarise oneself with the structure of a MDTV service system and look into how the system components cooperate with each other in the MDTV service lifecycle. The focus and motivation of our research would be thus placed at these key components and their interactions within a MDTV system.

Since MDTV derives from DTV, their overall system lifecycles are very similar. As to the DTV service, a generic DTV asset lifecycle includes the entire process consisting of tasks required to produce the generic DTV service types: a digital A/V service combining value-added applications. Its structure corresponds to the generic networked multimedia asset lifecycle: preproduction, production, postproduction, delivery, consumption and interaction/transaction. Therefore a generic DTV asset lifecycle can be illustrated as in Figure 2.1 and in order to make an improvement to any DTV service like MDTV, efforts can be put on these six segments below:

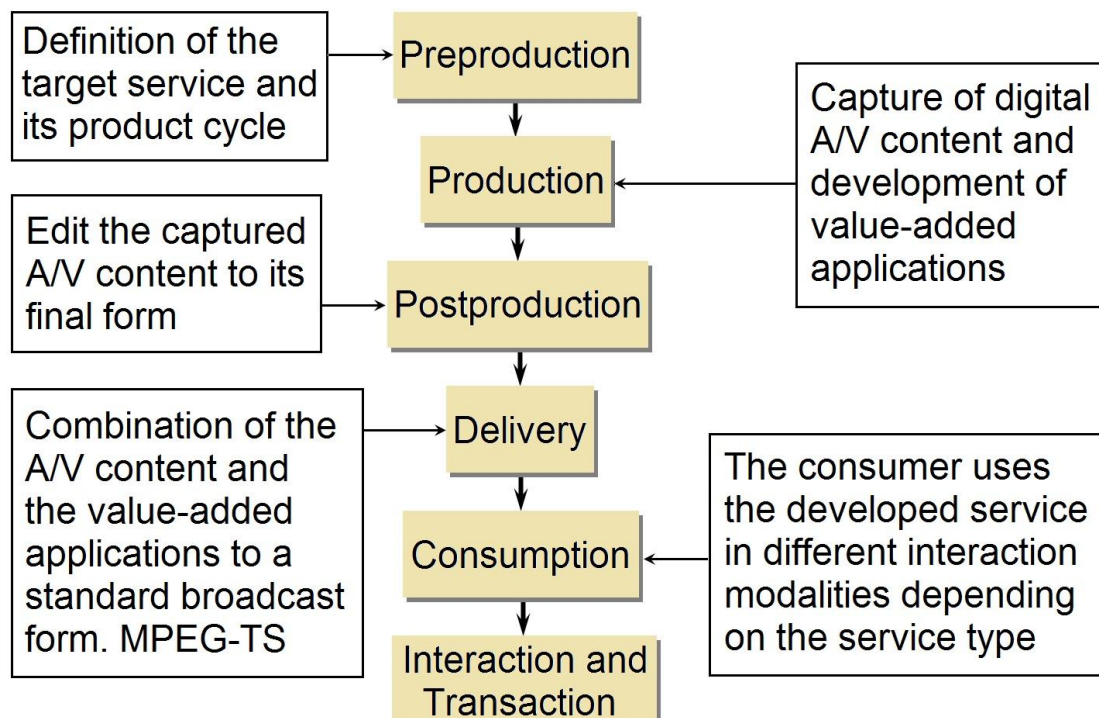


Figure 2.1: Generic abstract DTV asset lifecycle (adopted from [2] Chapter 2, page22, Fig 2.6)

DTV related business entities and service providers are involved along in the lifecycle

and the transactions between them to complete the process of DTV service production workflow. These commercial entities establish the service provision model and relative service providers realize the technological solutions for it. When the DTV services are created and edited ready to be distributed, different service providers (Broadcast service provider (BSP), Interactive service provider (ISP)) make the necessary agreements to deliver and implement the functionality of the corresponding DTV services. Prior to the consumption of the DTV services, transactions and agreements between the relevant commercial entities and the consumers are established, to achieve the final end-to-end business. Taking the DVB standard as an example, a practical system architecture is shown in Figure 2.2:

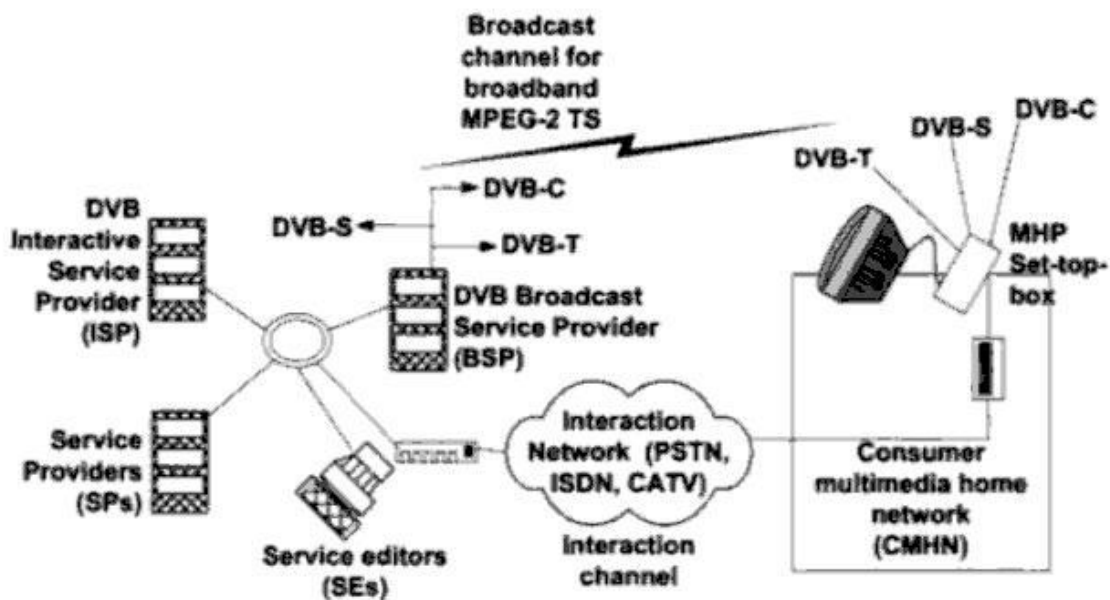


Figure 2.2: An extended DVB system architecture (adopted from [2] Chapter 2 p13)

Central Components	Functions
DVB Broadcast Service Provider (BSP)	Delivers broadband MPEG-2 transport streams (TSs) over different physical broadcast mediums (satellite, terrestrial, cable) to the consumer.
DVB Interactive Service Provider (ISP)	Provides functionalities and facilities for non local interactivity.
Service Providers (SPs)	Create service content and act as feedback network partners. BSP broadcast the content, Interaction channel and ISP help in achieving interactivity between the ISP and consumer.
Service Editors (SEs)	SEs are responsible for creating the overall services by implementing applications and A/V content delivered by the broadcasters

Broadcast Channel	The different physical broadcast mediums (satellite, terrestrial and cable).
Interaction Channel	A feedback channel for bi-directional information exchange provided by the ISP through various wired and wireless networks.
Consumer multimedia Home Network (CMHN)	The consumer access the MPEG-2 TS service with Multimedia Home Platform (MHP)-compliant set-top box.

Table 2.3: Functions of central components in DVB system architecture

There are two sections of the generic lifecycle that this Thesis would be focusing on, namely: the production and consumption, which correspond to the service editor and consumer home network of the above Figure and Table.

2.3 DTV SERVICE CONSUMPTION

The consumption of DTV service can be considered as a practical paradigm of Human-Computer Interaction (HCI), which has been defined as “*the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them*” [56]. Within this context it refers to the DTV service consumption environment, where the HCI model consists of the DTV terminal facility, the consumer and the actions between them.



Figure 2.3: Reference DTV service consumption HCI model

In this model, the DTV service broadcaster firstly forwards the default service content to the consumer as per contract agreement. The initial part of the consumption is accomplished when the consumer receives and experiences those contents through

certain DTV service terminal devices such as a TV with a set-top box or a mobile phone with a DTV feature unit. Then the consumer will either watch those DTV content or register actions with the service terminal device according to the types of the DTV content; for example, pressing a button on a controller to join in a live quiz show, to pay to view a specific piece of premium content, to personalize own profile and many more. This interaction between the consumer and the service providers is achieved through the interaction channel. Additional consumption is accomplished when the service provider responds according to these consumer/user requirements by creating and sending modified/selected content to the consumer or recording the consumer's preferences through the broadcast/interaction channel.

2.3.1 DTV SERVICE END-USER TERMINAL ARCHITECTURE

During the consumption of a DTV service, one of the most essential medium is the end-user terminal device. It is designed to include the abilities of receiving the service signal streams, demodulating and decoding the service content for the display unit to retrieve the A/V and added-on service, connecting the server and the client, and handling the interactions between them. There have been different types of DTV terminal devices in the market and the two most typical types are the set-box for most of stationary digital TV services and the MDTV featured mobile devices for MDTV services. The software stacks of both of these are very similar as shown below:

Set-top Box Architecture Example	MDTV End-user Device Architecture Example	Stack Components	Layers
GUIs, Interactive applications	GUIs, Interactive applications	Applications based on the middleware APIs	Application Layer
DVB-HTML, DVB-J, BML,	MIDP, BML	Application programming interface (API)	Middleware Layer
MHP, BML, DASE,	BML, MIDP	Middleware for specific operating system	
Linux, Windows CE	Linux, Symbian, Android, Windows CE	Operating System	OS & Hardware
Hardware Drivers	Hardware Drivers	Drivers	
Demodulator, Demultiplexer, Decoder	Demodulator, Demultiplexer, Decoder	Hardware	

Table 2.4: General DTV set-top box or featured unit software stack

The Operating system (OS) and Hardware presents the bottom layer of the software stack. Hardware units like antenna, demodulator and decoder are required to perform the fundamental DTV functionalities such as receiving signals and unpacking them into A/V and service stream, when there is also a CPU and memory to assist the processing of commands and manage the hardware environment within a DTV feature unit. The OS has the duty of connecting software and applications with hardware and enabling software to use the hardware functionality. Also there are necessary hardware related APIs being installed in the OS so as to make the corresponding hardware aware and available to the system.

Middleware and applications are part of the software layer. Middleware is a special designed software layer that is between the OS/hardware and the service applications, employed as a platform to hide the differentiations of OSs (Linux, Symbian or Android) and hardware structures from upper components. All the DTV service applications are then programmed according to the Application Programming Interface (API) provided by middleware and executed on the DTV terminals that implement this middleware API.

2.3.2 MIDDLEWARE INTRODUCTION

When creating DTV service applications, developers need to consider a series of issues, such as in which DTV service the applications belong to, which model the DTV terminal device is, what its functional ability is, what the configuration of the hardware is and which operating system it comes with. Solutions have to be proposed after balancing among those factors or limitations, since different hardware and operating system conditions may result in different performances for the same application.

Thus in the early years of DTV, before the definition of middleware, most of these works had to rely on the support from commercial entities due to the high-cost and complex characteristic. The service creation and terminal device production of a

digital TV services were in the proprietary domain for a number of specific broadcasters at a time, which limited the development of the DTV services in the horizontal market. Then the middleware came into our view by acting as an additional software layer between hardware and service application software, offering uniform interfaces to both of them. This means that any service applications based on this middleware will have the same performances including visual, audio and control experience as well as other functional mechanisms on any type of hardware where this middleware is installed on. The introduction of middleware to DTV unifies the service execution procedure and thus reduces the complexity during the service production.

Moreover, when different DTV standards had their corresponding middleware standardized, the middleware is eventually accepted as an important part within the DTV system. Terminal device manufacturers, service providers and broadcasters thus have middleware standards as the reference during the production of DTV services. The development and the evolution of DTV service is further encouraged, resulting in the new generations of DTV services such as interactive DTV services, which are quickly becoming available to more and more end users.

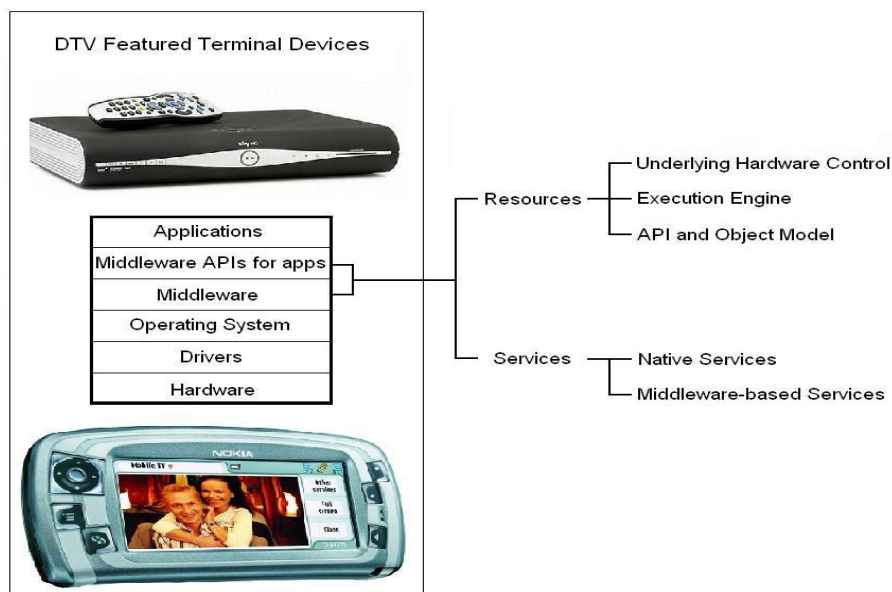


Figure 2.4: Reference middleware architecture (adopted from [57] Figure 1)

A middleware consists of application programming interfaces (API) and mechanisms relevant to support upper applications and abstract lower layer of the terminal device components. The common reference middleware function architecture is as in Figure 2.4. Two aspects of these functions are:

- Resources contain the functions that control underlying hardware; a group of engines (Java Virtual Machine and etc.) in which applications and other assistant script codes can be parsed and executed; reference APIs and models for application (creation and execution of interactive applications, GUI, etc.).
- Services contain native services that are executed by the underlying hardware directly and the middleware-based services that are created and run on top of the middleware.

2.4 GLOBAL DTV STANDARDS MIDDLEWARE SOLUTIONS

2.4.1 STATIONARY DTV SERVICE MIDDLEWARE SOLUTIONS

All global leading DTV standards have developed their own middleware solutions for stationary DTV. These are: the Multimedia Home Platform (MHP) for the Digital Video Broadcast (DVB) standard, the Advanced Common Application Platform (ACAP) for the Advanced Television Systems Committee (ATSC) standard and the Association of Radio Industries and Businesses Standard (ARIB STD) B23 for the (Integrated Services Digital Broadcasting (ISDB) standard. Besides these, there are a few middleware solutions supported by service providers or network providers such as the Multimedia and Hypermedia Experts Group – 5 (MHEG-5).

2.4.1.1 DVB Multimedia Home Platform and Globally Executable MHP

The DVB Project realised in 1997 that it would be a natural progression to build a set of specifications that would result in an open standardized middleware solution for their broadcast standard with its core based on Java technology. The first demonstrations of MHP took place at the IFA show in Germany in 1999, followed by

the first version of MHP published by ETSI in July 2000 and in 2002 the first MHP services were launched on the DVB-T platform in Finland.

The Multimedia Home Platform (MHP) middleware standard defines a comprehensive platform that enables interactive television services to be deployed, that are interoperable across any manufacturer's implementations of the standard ([58] 4.1.1). The key advantages of MHP are [59]:

- An open standard with multiple suppliers at all pointers in the value chain;
- A mature standard with many commercial and trial deployments;
- Allows true interactivity with actual television content, not just text and graphics;
- Works with all CA and DRM systems;
- A flexible standard which is proven to evolve with internet technologies;
- Specified for use in conjunction with all DVB transmission system specifications;
- Comes from the DVB Project, a tried and trusted source of DTV standards.

So far there have been three versions of the MHP published. All these versions support the enhanced broadcasting profile and interactive broadcasting profile. Version 1.1 adds the Internet access profile and the latest version 1.2 adds the IPTV profile as an upgrade. Several key components defined in MHP version 1.2.2 are listed in Table 2.5:

MHP Components	Description
MHP Transport Protocols	The protocols on both broadcast channel and interaction channel aspect are defined to provide a generic solution for a variety of broadcast only and interactive services through different network types (e.g. DVB networks including terrestrial, cable and satellite, Integrated Service Digital Network, etc).
Content Formats	DTV service content formats including audio, video, image, text and streaming formats as well as font and colour.
Application Model (DVB-HTML + DVB-J)	<ul style="list-style-type: none"> • Defines the application model based on DVB Mark-up Language (DVB-HTML) as well as the technologies employed in this model including eXtensible Mark-up language (XML), eXtensible Hypertext Mark-up Language (XHTML), Cascading Style Sheets (CSS), Document Object Model (DOM), and European Computer Manufacturers Association Script Language (ECMAScript); • Defines the application model based on DVB Java (DVB-J) Xlet as well as the technologies employed in this model; • Defines the lifecycle of the broadcast MHP applications.
Application Signalling	<ul style="list-style-type: none"> • Defines how the terminal receiver identifies the applications associated with a service and finds the location from which to retrieve them; • Defines the signalling that enables the broadcast to manage that lifecycle of applications;

	<ul style="list-style-type: none"> • Defines how the receiver can identify the sources of broadcast data required by the applications of a service.
DVB Java (DVB-J) Platform	Defines all the functional APIs and the execution environment of MHP service applications based on DVB-J.
Security	Define the following areas of security: <ul style="list-style-type: none"> • Authentication of applications; • Security policies for applications; • Authentication and privacy of the return channel communications; • Certificate management.
Graphics Reference Model	Defines the rules of displaying the DTV interactive service contents including video, interface components such buttons and lists, and raw graphical primitives on the screen of a DTV featured terminal unit.

Table 2.5: Key components of MHP [58]

Globally Executable MHP (GEM)

The Globally Executable Multimedia Home Platform (GEM) was firstly published in January 2003 and it was developed as a common interoperable core middleware platform whereby it was essentially the overlap between Open Cable Application Platform (OCAP) from CableLabs and Multimedia Home Platform (MHP) from the DVB Project. In June 2009 GEM and MHP have been refactored with MHP now referencing GEM, the primary DVB specification. GEM deployments now include OCAP on cable network and Advanced Common Application Platform (ACAP) on terrestrial networks in the USA and South Korea and ARIB B23 in Japan. South Korea is also the location for the first rollout of GEM-IPTV. Besides in Brazil, where ISDB-T has been adopted, GEM was the basis for the development of the Brazilian DTV GINGA-J specification.

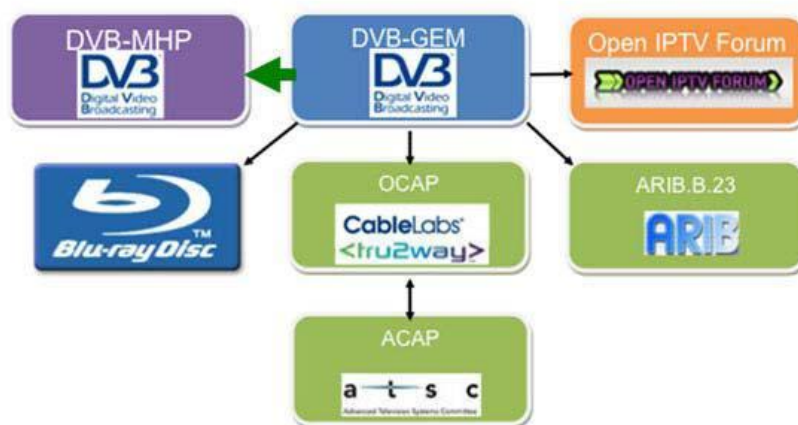


Figure 2.5: Relationship between GEM and GEM-based specifications [63]

GEM adds a technical solution for the user terminal that enables the reception and presentation of applications in a vendor, author and broadcaster neutral framework. Here “neutral” includes scenarios that consider legacy infrastructure. Applications from various service providers are interoperable with different GEM implementations in a horizontal market, where applications, networks, and GEM terminals can be made available by independent providers. The GEM specification aims to ensure interoperability between GEM applications and different implementations of platforms supporting GEM applications. [62].

The profile defined by GEM consists of three targets: Broadcast, Packaged Media and IPTV, of which the first two can be further subdivided into two application areas: Enhanced Broadcasting and Interactive Broadcasting, Enhanced Packaged Media and Interactive Packaged Media. The broadcast target is used by GEM-based terminal specifications in a broadcast environment such as MHP, OCAP, ATSC and ARIB. The packaged media profile targets on where the media is packaged on a physical medium which is possibly read-only such as Blu-ray. The IPTV target is used in the environment where media is transmitted over a bidirectional broadband connection such as MHP or Open IPTV.

In practice, a wealth of GEM applications are available for more traditional iTV applications like EPGs/IPGs, ESGs, email, chat, SMS, enhanced TV, news tickers, weather, games and etc.. These GEM applications can be run directly from standard web servers and can easily support web 2.0 features like RSS feeds, P2P and user-contributed contents [63].

2.4.1.2 Other widely adopted middleware solutions

Multimedia and Hypermedia Expert Groups (MHEG-5)

The International Standardisation Organization (ISO) set up the Multimedia and Hypermedia Expert Groups (MHEG) in 1997 to create a standard method of storage, exchange and display of multimedia presentation. MHEG-5 is a standard devised for

the middleware of digital Teletext services in the United Kingdom. MHEG-5 uses the model of multimedia presentations, where a multimedia presentation is a group of scenes, which include a collection of objects such as buttons, graphics, text, and links that define the process triggered by user interaction [3]. MHEG-5 was designed to be supported by systems with minimal resources, which makes it a suitable middleware product for low-end digital set-top boxes. [4] The MHEG standard has had different version: MHEG-1 to 4 were on the road of developing MHEG-5, which are now rarely used. MHEG-5 makes MHEG the first horizontal market in DTV in the world and now it is being employed by several British broadcasters, of which the most famous one is the BBC. Applications like EPGs, superteletext and other interactive service are supported; MHEG-6 is an extension to MHEG-5 allowing the creation of Java-based applications; MHEG-7 has defined the test and conformance procedures of MHEG-5 applications and MHEG-8 provides XML scripting for MHEG-5[8][3].

MediaHighway

MediaHighway was one of the first proprietary middleware solutions developed by the Canal+ research and development department in 1994 for the launch of the first launch of the first French Digital Satellite TV service in 1998. Since its launch MediaHighway has been mainly employed by the Satellite providers of the Canal+ Group, that is all, the national variations of Canal+ in Italy, Spain, Netherlands, Finland, Poland, and so forth. MediaHighway's system architecture is not openly available, since it is a proprietary solution. It does however support a number of DTV applications, such as EPG, VOD, and pay-per-view functionality. In its current version MediaHighway supports Sun Microsystems Java language as a programming language. The MediaHighway Virtual Machine is hardware independent and implements the MediaHighway API in compliance with the Canal+ Technologies (former Canal+ R&D) specifications. Towards the end of 2003, the MediaHighway Company was acquired by the NDS middleware provider [8].

OpenTV

The American Company OpenTV Inc. is a middleware provider with the largest number of deployments worldwide, currently the leading middleware player in the vertical market, reaching over 27 million set-top boxes produced by more than 30 suppliers worldwide [8]. In the UK, one of the largest service provider SKY employs Open TV as their middleware solution [94]. The core middleware architecture developed by OpenTV is said to be hardware independent, modular, and extensible. The execution layer provides compatibility with applications author in C, HTML, and Java code. The C-code execution engine is provided as a part of the basic middleware package, and allows for execution of C-code applications. HTML and Java execution engine are offered as options in OpenTV in order to support DVB-MHP/GEM. Furthermore OpenTV offers a range of development tools for creating interactive Television applications for OpenTV middleware [3] [8].

Due to the great number of set-top box manufacturers and service providers that employ OpenTV, it has to support many different conditional access systems and offer a range of interactive applications. In this respect, OpenTV supports Near-Video-On-Demand, Pay-per-view, Electronic Program Guide (EPG), PVR functionality, and downloading of data and applications [8].

2.4.2 MOBILE DTV SERVICE MIDDLEWARE SOLUTIONS

2.4.2.1 Looking forward to the Open Middleware Solution – JSR 272

Java is well known as one of the most widely used object-oriented programming languages with the promise of “Write Once, Run Anywhere”. Besides the Java Enterprise Edition (Java EE) and the Java Standard Edition (Java SE), Java released the Java Micro Edition (Java ME) for mobile devices and embedded systems in 2006. Its core specification group, Mobile Information Device Profile (MIDP) has the goal of defining an enhanced architecture and the associated APIs required to enable an open, third-party, application development environment for mobile information devices (MIDs) such as mobile phone or PDAs [69]. The MIDP specification was

defined through the Java Community Process (JCP) by an expert group of more than 50 companies including the leading device manufacturers, wireless carriers and vendors of mobile software [70] and now Java Micro Edition and MIDP have been well adopted in the mobile industry field. Operated on top of Connected Limited Device Configuration (CLDC), MIDP provides a Java runtime environment for dynamically and securely delivering graphical, networked, and portable applications, for designing games, multimedia and messaging applications, for supporting dynamic deployment and updating of applications over the air, and for providing a robust security model built on open standards [71].

Besides the core specification API in the JSR118 package, JCP have been developing assistant optional API packages for MIDP, regarding to the requirements and improvements of the MID's functions, such as JSR82 for Bluetooth, JSR172 as J2ME web services specification, JSR209 for advance graphics and user interface, JSR927 for Java TV API, and JSR280 for XML API. Moreover, with regards to the development and deployment of mobile digital TV in recent years, there has been a requirement of a standard middleware solution between MDTV service applications and the environment within the mobile devices so that the MDTV services can be developed and operated regardless of the underlying hardware/software configurations of different MIDs.

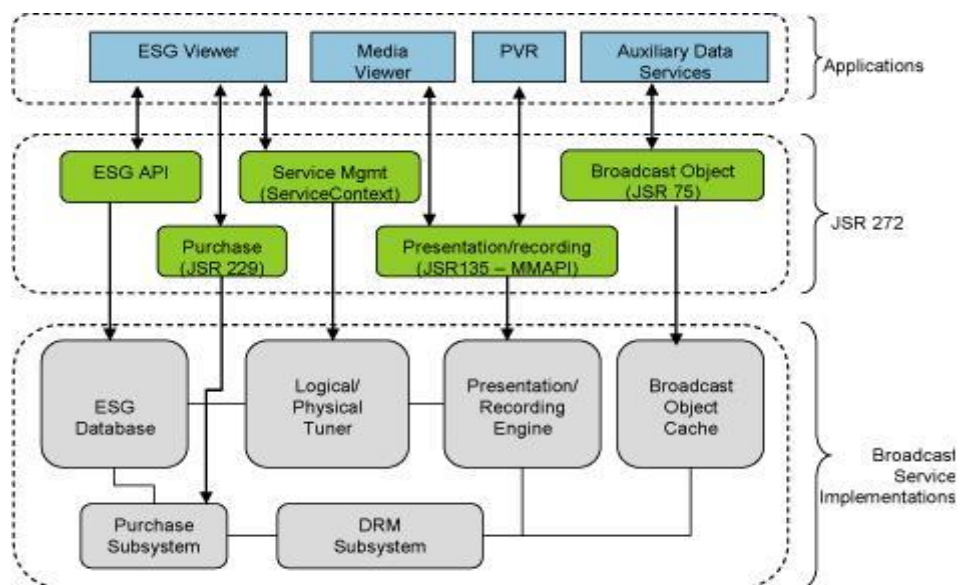


Figure 2.6: The architecture of a typical JSR272 implementation [74]

JSR272, the Mobile Broadcast Service API for Handheld Terminals, was then developed by an expert group of 22 members leading by Nokia and Motorola. The JSR272 package provides the functionalities that enable applications to receive and view digital television. In addition, the API gives access to electronic service guide (ESG) and file carousel, which are the fundamental features of digital television [72]. After a series of specified configurations, the JSR272 API works well with a wide range of MDTV underlying technologies, such as DVB-H, DMB, OMA-BCAST, MediaFLO and TV-Anytime. The architecture of a typical JSR272 implementation is shown in Figure 2.6.

A typical implementation of JSR272 provides the main functionalities described below [74]:

- An Electronic Service Guide (ESG) viewer that allows a user to navigate, discover and select. JSR272 abstracts differences in different schemas of different broadcast specifications.
- Service and programme selection is supported and once a programme is selected, the system allows the user to “turn into” the broadcast channel that the programme is transmitted.
- The media content of the selected programme can be presented to the user with different viewing options. The system may also allow the user to record the programme for future viewing.
- A programme may also carry auxiliary content/data that adds to the viewing experience, for example, a slide show of images.
- Security mechanism are built in and digital rights management (DRM) is supported.
- Purchasing can be conducted in corporation with JSR229.

Compared to the functionalities of a typical DTV middleware, JSR272 is uncompleted as a MDTV middleware since several definitions are missed such as the Application

Model and Graphics Model. Even though JSR272 is developed in Java ME environment and thus supported by other functional JSRs, those two functions have yet to be defined. Extra work is therefore needed to define or develop those missing parts if JSR272 is employed as the middleware solution for MDTV services, which could be an important reason why JSR272 has still not been widely accepted since its final release in 2008.

2.4.2.2 Standard-specific middleware solutions

ATSC-Mobile/Handheld standard A153 part4, part5

ATSC-M/H is the sub standard group of ATSC digital TV for mobile DTV aspect. It defines the most specifications for mobile TV including RF/Transmission System Characteristics (part2), Multiplex and Transport Subsystem Characteristics (part3), Announcement (part4), Application Framework (part5), Security Protection (part6) and Audio/Video System Characteristics (part 7 and 8). In the standard group, part 4 and 5 defines most of the issues concerned with middleware functionality.

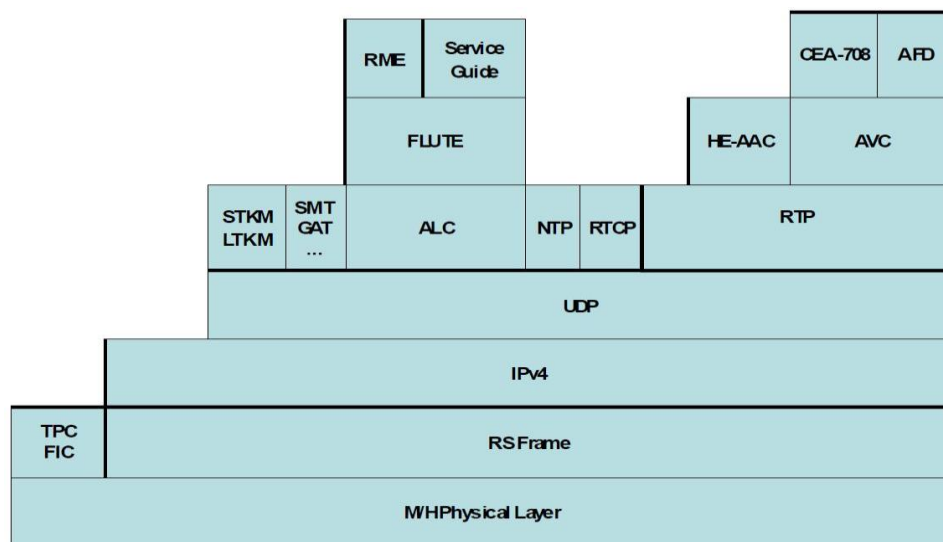


Figure 2.7: ATSC-M/H system protocol stack [50]

Announcement (part4) System Specifications define the data formats and delivery mechanisms used to announce the content and services being delivered, or scheduled for delivery, in the M/H broadcast stream.

In an ATSC-M/H system, the Services available from a broadcast are announced by using Service Guide (SG). A SG is a special M/H Service that is declared in the Service Signalling subsystem. An M/H receiver determines available SGs by reading the Guide Access Table for M/H (GAT-MH). This table lists the SGs present in the M/H broadcast, gives information about the service provider for each guide, and give access information for each guide. The ATSC-M/H SG is an OMA BCAST Service Guide, with constraints and extensions as specified in this standard.

Application Framework (part 5) is similar to the definition of the application model in other middleware solutions. The primary objective for the M/H platform is to deliver a set of audio/or video services from a transmission site to mobile or portable devices. The Application Framework thus enables the broadcaster of the audio-visual service to author and insert supplemental content to define and control various additional elements to be used in conjunction with the M/H audio-visual service. Furthermore, it enables the broadcaster to send remote events to modify the presentation and to control the presentation timeline. The Application Framework further enables coherent rendering of the service and input fields, and event handling and scripting associated with such buttons and fields. When the interactive application framework is included, the broadcaster, service provider, or content provider is able to create and control the presentation aspects of the service by applying the interactive aspects of the Application Framework.

M/H's Application Framework is defined based upon the OMA Rich Media Environment (OMA RME). It also extends and constrains some OMA RME components such as the use of audio/video/image Element, ATSCGlobal Interface, SVG Global Module and DIMS Session Description Protocol (SDP). The OMA RME in turn builds on W3C SVG Tiny 1.2, 3GPP Dynamic Interactive Multimedia Scenes (DIMS), and the Mobile Profile of ECMAScript which is a rich scripting language [50].

DMB - Multimedia Application Terminal Environment (MATE)

As mentioned in Chapter 2, Digital Multimedia Broadcasting (DMB) service is based on standard group of Digital Audio Broadcasting (DAB). In August 2009, ETSI published the “Digital Audio Broadcasting (DAB) Middleware: Part 1: System aspects and Part 2: DAB” as middleware solution for mobile digital TV services.

MATE is a middleware standard for a platform-independent environment, where executable applications can be signalled and transferred to a receiver via a broadcast network and executed on the receiver. It does not support the exclusive use of a specific broadcast network but defines the commonly-required specifications among diverse broadcast networks. It includes the definitions of basic data formats, protocols to deliver data, to signal downloadable applications and to download them, ways to denote resources on broadcast networks, and detailed interfaces among receiver platform, broadcast and communication, and the applications.

The MATE standard takes an approach of specifying abstract models for external entities such as broadcast network to broaden the range of external environments. Thus, MATE is not a valid standard for any implementation by itself and the abstract models need to be concreted according to the external environments.

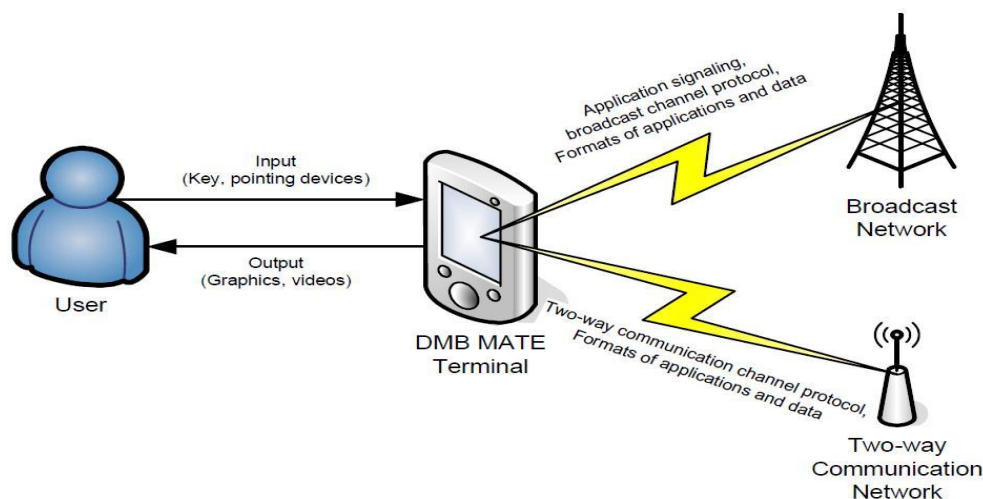


Figure 2.8: Environment external to MATE terminal [75]

MATE defines its **transport protocol** from broadcast and communication channel aspects. For broadcast channel protocols, file transport protocol, packet streaming protocol, and trigger protocol for synchronization of timed media are supported.

Multimedia Object Transport (MOT) protocol for DAB and DSM-CC object/data carousel are adopted when defining those three sub protocols. For the communication channel protocol, UDP, TCP and HTTP which are based on IP layer are involved. The **Locator model** defined in MATE is the model of locator represented as a string to indicate an object (a broadcast channel, or a package/video/audio stream and so on) for applications to identify it on a broadcast network in MATE. The **Security model** is to guarantee the integrity of application in a receiver and authenticate as well as authorizing applications when they are being consumed. The **Graphic system model** defines two layers while presenting visual contents: video plane at the background for one or more videos and graphics plane at the front with transparent ability.

MATE chose Java Micro Edition (ME) MIDlet as its **application model** and its application execution environment, which consists of a Java Virtual Machine (JVM) and APIs, is based and extended on Java ME MIDP 2.0 (JSR118) and Java ME Personal Basic Profile 1.1 (JSR217). PDA optional package (JSR75) and Mobile Media API 1.1 (JSR135) are involved as the optional packages. One of the extensions of DMB which has made it to MIDP for MATE is the Graphic user interface (GUI) API. With the help of additional rules and APIs, DMB-MIDP applications can be displayed and shared with in the same screen.

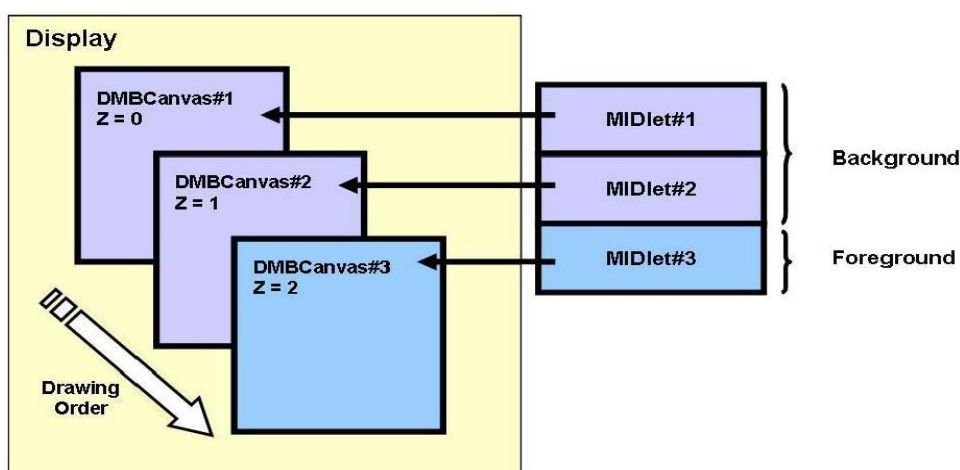


Figure 2.9: Multiple applications display mode [76]

ISDB mobile DTV middleware solution

ARIB provides the operational guidelines in STD B24 Volume 2 for implementing basic services (STD B24 appendix 2) and extended services with different receiving condition including fixed receiving system (STD B24 appendix 3), portable receiving system (STD B24 appendix 4) and mobile receiving system (STD B24 appendix 5). The STD B24 appendix 5 contains the operational guidelines for implementing and integrated service that provides broadcasting services and services via the Internet for a mobile receiving system, by using the XML-based multimedia coding scheme specification responsible for the data broadcasting scheme, part of the digital data broadcasting scheme specified as the standard in Japan [29].

2.4.2.3 Middleware-like solution in 3G

Regarding to the deployment of MDTV service over mobile telecommunication networks such as a 3G network, there is no middleware definition against MDTV, particularly since MDTV service is only one of the multimedia services over the 3G network. Functions defined in a typical DTV middleware standard (refer to Table 2.5) are defined in different categories that refer to the broadcasting and the streaming service specialities over the 3G network. However, the solution for enabling multimedia services is also considered important and the relative standards on such solution have been developed and published, which include the support of MDTV services. Two leading standards are the Dynamic and Interactive Multimedia Scenes (DIMS) from the 3GPP and Rich Media Environment (RME) from OMA.

3GPP-DIMS

The dynamic and Interactive Multimedia Scenes (DIMS) is a dynamic, interactive, scene-based media system which enables the display and interactive control of multimedia data such as audio, video, graphics, images and text. The motivation of developing such system is the development of the next generation mobile infrastructure and the generalization of TV content to new mobile environments. The first version of DIMS is v7.0.0 approved in 2007 and the latest version is 9.0.0 revised

in 2009.

DIMS v9.0.0 mainly defines the architecture of a rich media system, the media-type used in the system, the supports to the interaction and scripting events, and the transport mechanisms that support the delivery of rich media. The **rich media system** is actually a client-server architecture comprising of 3 main components: the rich media server, transport mechanisms and the rich media client, as illustrated in Figure 2.10. The server prepares the rich media contents and encapsulates them; then the system utilizes various transport mechanisms based on “Pier-to-Pier” (e.g. PSS) or “Pier-to-Many” (e.g. MBMS) protocols for download, progressive download and streaming scenarios; finally the content is retrieved on the client allowing for local and remote interactivity of feedback and data requests. The **media-type** used in the system is based on the Scalable Vector Graphics Tiny (SVG-T), the **interaction** event is handled using the Document Object Model (DOM) and the **scripting** event is handled using ECMAScript. In the end, the **transport mechanisms** support rich media delivery in the following modes: unicast download, broadcast/multicast download, unicast streaming and broadcast/multicast streaming [86].

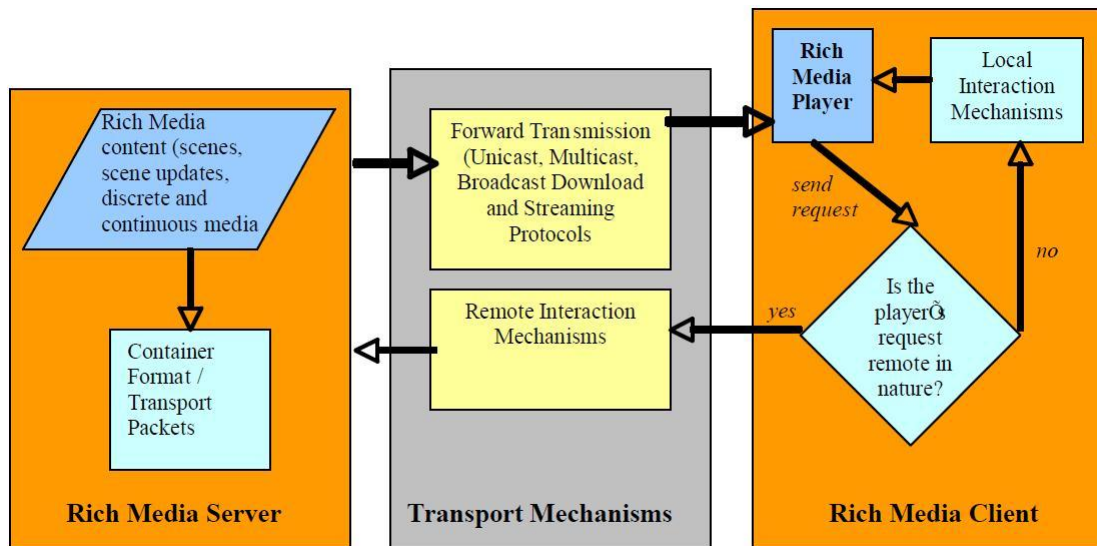


Figure 2.10: General architecture of the rich media system [86] Figure 4-1

OMA-RME

The Rich Media Environment is similar to DIMS, as it has been also developed by the Open Mobile Alliance to address enhanced rich media services to mobile devices including rich-media content and service creation, deployment, distribution and presentation. Also, OMA-RME is designed to be able to compatible with 3GPP-DIMS but a big difference between them is that RME is agnostic of any bearer networks including cellular network (e.g. PSS and MBMS) and non-cellular network (e.g. DVB network and ATSC network) whilst DIMS is not.

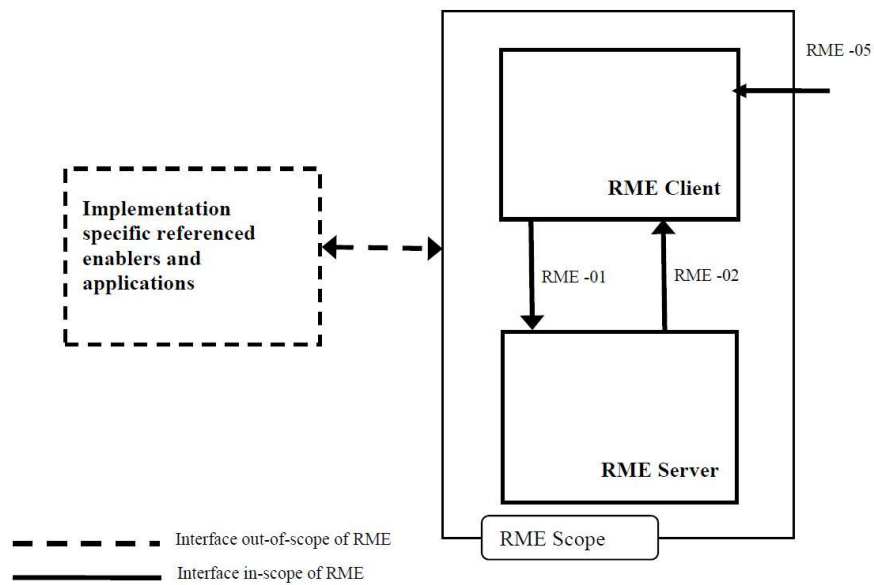


Figure 2.11: RME client-server architecture and RME scope [55]

RME system is also based on the client-server architecture. The RME Client typically resides on the mobile terminal and provides the capability to display RME data, handle dynamic updates to the RME Scene as well as local remote interaction with the scene objects. Typically the RME Server is the source of rich media data and provides data to the client. RME also employs SVG-T as the scene description language and MPEG-4 part 20 (LAsER) for scene updating commands definition. ECMAScript is the scripting language used in RME [55] [96] [77].

2.4.2.4 Solution for MDTV service over DVB network

Considering DVB broadcasting to handheld devices including DVB-H service and DVB-Satellite Services to Handheld Devices (DVB-SH), even if there is specification

known as IPDC defined upon the network layer, there is no middleware or software standard on the upper layer of MDTV system which defines specifications for service applications implementations. Specifications such as application model, application signalling mechanism, application execution platform and graphics model remain undefined for MDTV service over the DVB network. Therefore the implementation of MDTV services over the DVB network usually rely on commercial solutions during service contents/applications creation, service software platform design on the terminal devices and interactive service implementation mechanism between server and user-ends. Besides, DVB broadcast services are also planning to harmonize with OMA BCAST, which can raise the adaptability of DVB underlying technologies with OMA application layer technologies.

Nokia Mobile Broadcast Solution (MBS)

The Nokia Mobile Broadcast Solution is a globally deployed DVB-H server platform for commercial mobile TV services. These services can utilize the current TV content with minimal impact to the existing production system. Its latest version MBS 3.3 (2Q2008) is compatible with the DVB IPDC 1.0 and OMA BCAST 1.0 sets of specifications. It supports real-time streaming control for multiple content sources, clear-to-air/paid broadcast services and flexible purchase mechanisms and multiple business models. Besides it provides service protection with Internet Protocol Security (IPSec), Secure Real-time Transport Protocol (SRTP) and Internet Streaming Media Alliance Encryption and Authentication (ISMAcryp), content protection, both IPDC and BCAST ESG solution with configurable formats, and a set of open APIs that allow seamless integration of the solution to all the relevant content creation, monitoring and subscriber management systems.

The Air Interface between the terminal and the networks is the most important interface in Nokia's mobile broadcast architecture. It defines how terminals can access the broadcast services, which includes the broadcast interface and the cellular radio interface for interactive services as well as the applications protocols on top of the

transmission channels. Nokia continuously publish the mobile TV implementation guidelines (Mobile TV Implementation Guide (MTV-IG), MTV-IG 3.0 for the latest version) for its air interface, which includes the definition of network and transport protocols, service guide solution, service purchase and protection specification, A/V streaming and File delivery. In order to reach the interoperability, Nokia also offer many alternative options within their Implementation Guidelines for different commercial implementation.

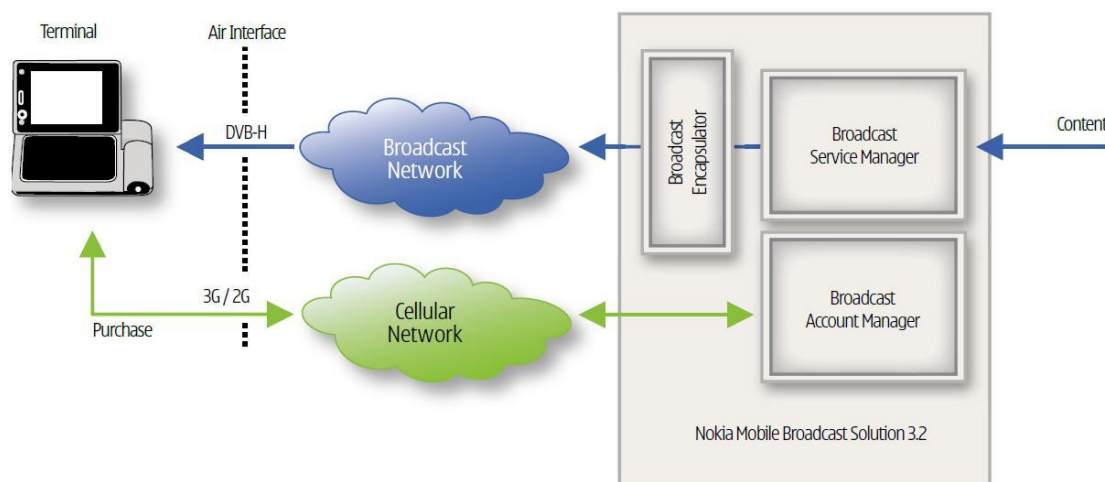


Figure 2.12: Nokia Mobile Broadcast Solution 3.2 [78]

The middleware layer solution has not been published as an individual specification in Nokia MBS and the solution is treated as a part of Nokia terminal device solution. The Nokia MBS Air Interface defines most of the middleware functionality. Nokia DVB-H featured mobile phone such as (Nokia N92 and N96) usually has a MDTV reception unit and a software platform for ESG, pay TV, recording and other basic interactive applications [77] [78] [79].

2.5 MDTV SERVICE CREATION AND IMPLEMENTATION ON THE APPLICATION LAYER

Based on the aforementioned review on the middleware solutions of MDTV service, we can conclude that not all the MDTV standards have middleware specifications. More precisely, standards such as ATSC-M/H, ISDB-T 1seg and DMB have

middleware specifications whilst DVB does not. The 3G network system does not define a middleware but enables MDTV services with middleware-like software system. This situation is different from the conventional DTV, in which middleware have been standardized and become a formal part of the system. Most of the DTV service deployments tend to follow the middleware standard and the service creation and implementation processes are more industrialized. Commercial entities therefore act more like implementers of a standard. As to the service creation and implementation of the MDTV service, there are various solutions with different scopes and aims in the current market. Commercial entities are not only the implementers, but are also involved in technical development and provision of different solutions with excellent performance. There are mainly two categories of solutions for the current MDTV service creation and implementation: standard-based solution and commercial solution.

2.5.1 STANDARD-BASED SOLUTION

For MDTV standards such as ATSC-M/H, ISDB-T 1seg and DMB, which have the middleware specifications defined as a part of their standards, the service creation and implementation of these standards tend to follow the corresponding middleware specification (e.g. ATSC-M/H Application Framework, ISDB-BML and DMB-MATE). As far as the MDTV service over 3G networks, as there are software specification such as 3GPP-DIMS and OMA-RME enabling the service instead of middleware, the service creation and implementation based on 3G networks tend to take those software specifications as the reference.

Commercial entities that apply this kind of solutions have minimal involvement in any technical development work and most of their work is based on the execution of the middleware/software specifications as required. Examples of these refer to applying the transport protocols in the specification, collecting and producing MDTV service content according to the relative content format definition, creating and

designing the service applications under the pre-defined application model with the specified technologies, designing the service software platform on the terminal device according to the application execution platform definition in the specification and many more.

The advantage of such a solution is that the service creation and implementation processes are relatively easy to achieve. No extra time or expense is spent on developing/deciding those technical components defined in a middleware/software specification. Therefore this kind of solution is usually an economical and quick option for the implementation of a MDTV service. However, those middleware/software specifications are usually compliant only to their corresponding MDTV network standards, thus these specification-based service creation and implementation processes then restrict their output services which become incompatible with each other. Therefore the relative commercial entities may have to adopt different schemes in order to create/implement the same service (such as a TV programme service) for each of its possible underlying MDTV network standards, resulting in the same MDTV service having to be repurposed several times for each distinct standard. This leads to an increase of the time, costs, as well as resources required for the production of an MDTV service.

2.5.2 COMMERCIAL SOLUTION

Actually during the implementations on different layers of MDTV system, the commercial solutions usually coexist with corresponding standards (specifications). Both standards and commercial solutions aim to provide the methods to solve the problem, although standards tend to ensure a normative implementation (last section) but commercial methods seek for a quicker and efficient deployment in the market. There is also a trend of harmonization between them so that commercial solutions tend to support the corresponding standards and the standards are updated to cover technologies which are approved by the commercial solutions. In spite of this, the

commercial solution still plays its role within the MDTV industry field: namely to provide the most effective and economical solution to their customers and to actualize the implementation with more practical features or even novelties. In this case, the commercial entities act more like the developers against the specific topics (such as the solution of MDTV service creation and implementation).

As to the MDTV service creation and implementation, the commercial solution usually defines a similar software mechanism (including the application model) as middleware/software standards. But regarding the different business strategies of the commercial entities, actually different scopes of solutions have been developed and adopted. Some focus on the application layer, offering a range of packages of MDTV middleware and software environment solutions with ESG, A/V and interactive applications frameworks; some even address in wider scopes, providing the integration of broadcast system solutions from the actual implementation of the MDTV service from the content provider/broadcaster, to the network operator and service operator as well as the terminal device hardware, middleware and software platforms.

2.5.2.1 Middleware-oriented solution

These solutions only provide middleware solutions rather than an integrated system.

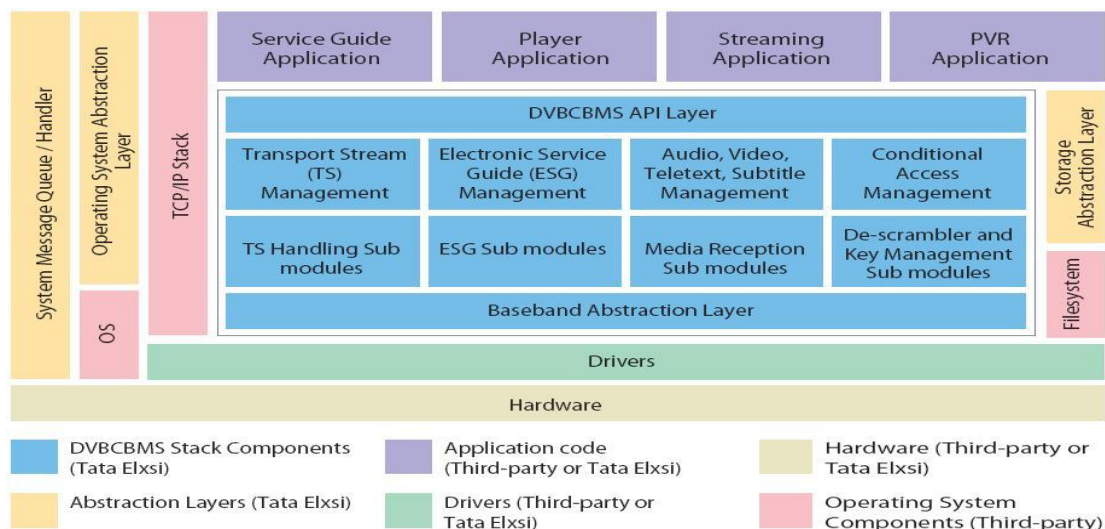


Figure 2.13: TATA ELXSI DVB-CBMS stack [87]

TATA ELXSI provides a middleware solution for DVB-H IPDC. Its DVB-H stack is compliant with the DVB-CBMS specification and is written in ANSI-C. This solution is operating system agnostic and featured with APIs for integration with various VA modules, multimedia engines, decoders and applications [87].

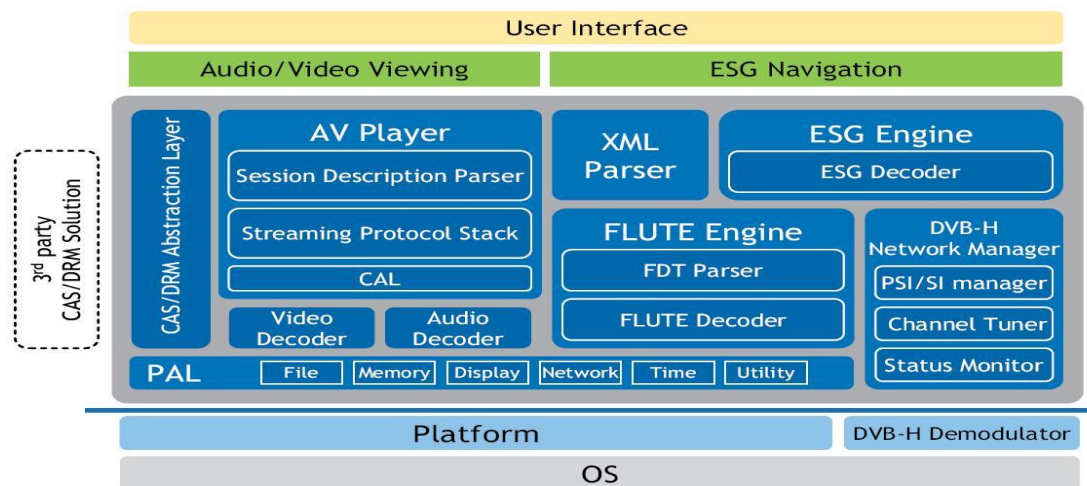


Figure 2.14: TMI Mobile TV middleware solution on DVB-H – thinDVB-H [88]

TMI Mobile TV middleware solution produced by Thin Multimedia Inc (tmi) has two sub sections facing to DVB-H and T-DMB. Both of them implement the basic service requirements of the corresponding service system including basic UI, ESG, A/V codec, PVR and purchase mechanism [88].

2.5.2.2 Integrated software system solution

The integrated software system solution is more popular in the service creation and implementation field because it offers an integrated system for service implementation that spans the entire service production lifecycle: all or most of the necessary components on the MDTV application layer such as middleware, service development APIs or predesigned service applications, or even the software platform on the server side as well as client side; many of solutions provide the service application authoring tools, which is more convenient for an easy service creation. Besides, most of these solutions have seamless mechanisms for an easy adoption on various MDTV terminal devices. More precisely for some of the solutions, the client software systems are preinstalled in the mobile devices before publishing, whilst some

of them are even developed by the terminal device manufacturers such as in case of Nokia, which is much easier to deploy.

According to the different architectures of the client-side service software platform in these integrated system solutions, this type of solutions can be categorized into two groups: Generic browser based and Rich Media based.

Generic browser based

A generic browser based client-side service platform is mainly a software mechanism based on the concept of a browser. All the MDTV service contents including the AV stream, interactive application and other multimedia are parsed and presented to the user through this browser. Its structure is similar with a common web browser or based on some of them (NetFront for example). Content and interactive services are pre-defined within a mark-up language (such as XML, XHTML or BML) based application model and can be synchronized with the AV stream. Due to the mark-up language based service outline, which is widely adopted in the web development field, it requires less design and development time and cost during the commercial software process. Besides, this group of solutions is good in providing basic interactivities such as voting by using hyperlink through SMS and http. However, it is relatively too basic to handle advanced interactive applications like online gaming and advanced UI components like SVG. The UI experience and graphics quality is not as good as with the rich media based solution.

HisTV is a browser based MDTV service solution based on DVB-H. It provides service managing server and client side service platform. The main part of the service platform is the display engine that parses and presents all the MDTV service contents. Basic interactions (SMS and connection via http) through return channel are supported. Content authoring reference and APIs are provided as well [89].

NetFront Browser is the leading product of ACCESS and its mobile TV profile faces

to several leading MDTV standard like DVB-H, ISDB-T 1seg. This profile combines the NetFront Browser DTV Profile with the FastESG service guide from EXPWAY and it is compliant with DVB-IPDC and OMA-BCAST standards. ACCESS also provides the SDK which is an HTML porting/customization kit for ISDB-T based DTV and set-top box field [90].

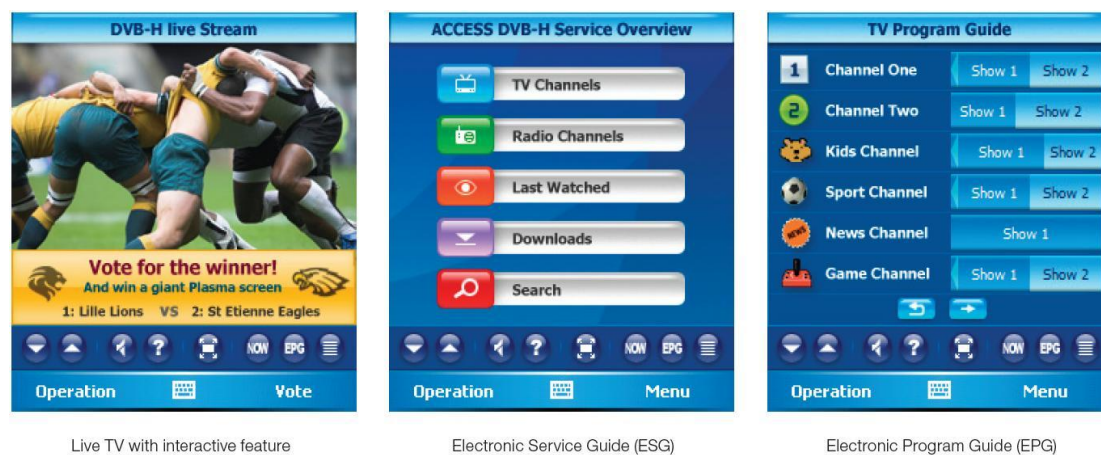


Figure 2.15: User interface of NetFront Browser for MDTV [90]

onHandTV from Silicon and Software System delivers a MDTV software client compliant with DVB-IPDC and OMA BCAST that enables broadcasters, mobile operators and device manufacturers to deliver interactive video and data services through mobile and handheld devices. It has a middleware section consisting of ESG (DVB, OMA and T-DMB standards) engine, subtitles renderer, file delivery and datacasting, conditional access abstraction layer, file transport protocols, video recorder and interactive services manager. Along with middleware, the application platform is also provided with a TV and audio Browser for rendering MDTV contents and a service application generating environment compliant with various mobile programming language such as C++, C#, Java, Symbian C++, etc [81].

Rich Media based

Rich media is today becoming more and more popular in content/service design as well as terminal service platform design. 3G multimedia services have incorporated rich media to their mobile services and several rich media standards are published:

3GPP-DIMS and OMA-RME. Recently, rich media has been adopted in the MDTV service platform development. The SVG-Tiny graphics and animation based UI experience is the key feature. The SVG based MDTV contents/services are programmed in the XML language format model and thus they are much easier to design, pack and transport in the broadcast stream. A rich media engine is needed to parse and present all the services. The interactive components in the MDTV service are supported by scripting languages like the ECMAScript and a rich media server. However, the design and development of the UI graphics require custom made proprietary authoring tools, which on one hand result in more design production costs and on the other hand require additional time invested in training to use these custom tools, limiting the number of creative individuals and designers who can be involved in the design process. In short, this group of solutions is relatively expensive and hard to maintain.

FastESG EXPWAY The EXPWAY's mobile TV solution provides end-to-end solutions ranging from platforms for operators and broadcasters, to software engines for device manufacturers. EXPWAY works closely with leading provider of Conditional Access System, Digital Right Management systems, Head-End Systems, Video Encoders and Decoders, Graphical User Engine and Video Player and its mobile TV solutions are fully compliant with international Mobile TV standard such as ATSC-M/H, DVB-H, DVB-T, DVB-SH, OMA-BCAST and 3GPP.

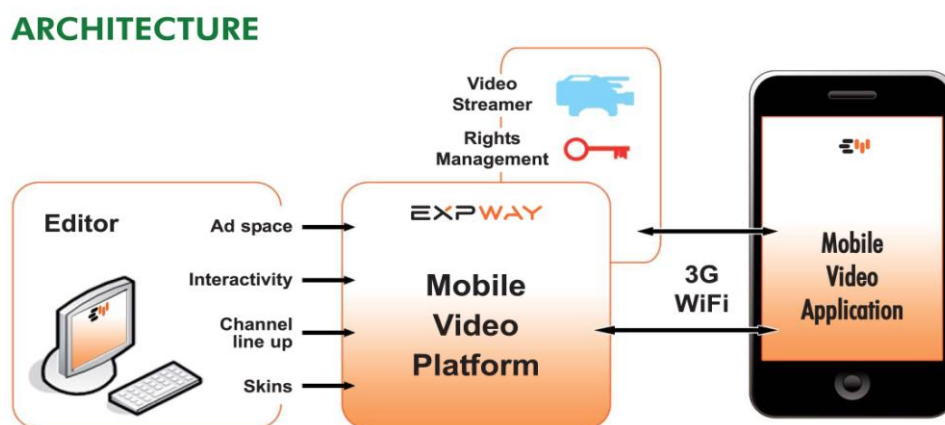


Figure 2.16: Architecture of FastESG Mobile TV platform [80]

FastESG Mobile TV platform is a complete solution for creating, aggregating and delivering advanced Mobile TV services over broadcast and cellular networks. It provides Content Providers, Commercial Operators, Network Operators and Broadcast Operators with a management platform for ESG, interactive and datacasting services. The **FastESG Mobile TV engine** is an integrated software solution for mobile devices manufacturers, chipset manufacturers, and middleware providers. It provides advanced mobile TV service to end-users on any type of terminal, regardless of the network, including ESG, advertising, interactive and datacasting services and audience measurement tools.

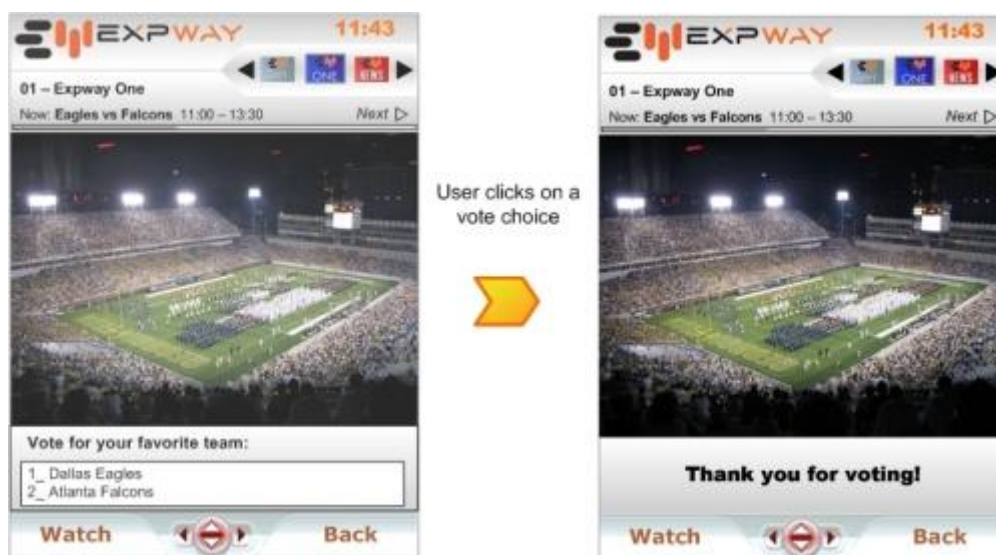


Figure 2.17: Interactivity on FastESG [80]

EXPWAY has so far been in cooperation with many relevant technology and commercial entities such as Motorola Networks, Accenture, Access and many more. FastESG is also adopted in many MDTV services like 3G Italy, Orange, and Vodafone and device manufacturers like Apple, LG and Samsung has applied FastESG in their MDTV featured devices [80].

Streamezzo Interactive Mobile TV is a rich media based MDTV service platform solution (see Figure 2.18). It was originally designed for 3G based MDTV but can be adapted for other standards such as DVB-H, DMB and MediaFLO. Many MDTV services as well as value-added service mechanisms are supported (EPG, VoD, voting,

purchasing and PVR). The Streamezzo solution provides support to most of the phases in the MDTV service software process from content/service creation, service managing server and terminal service platform [91].

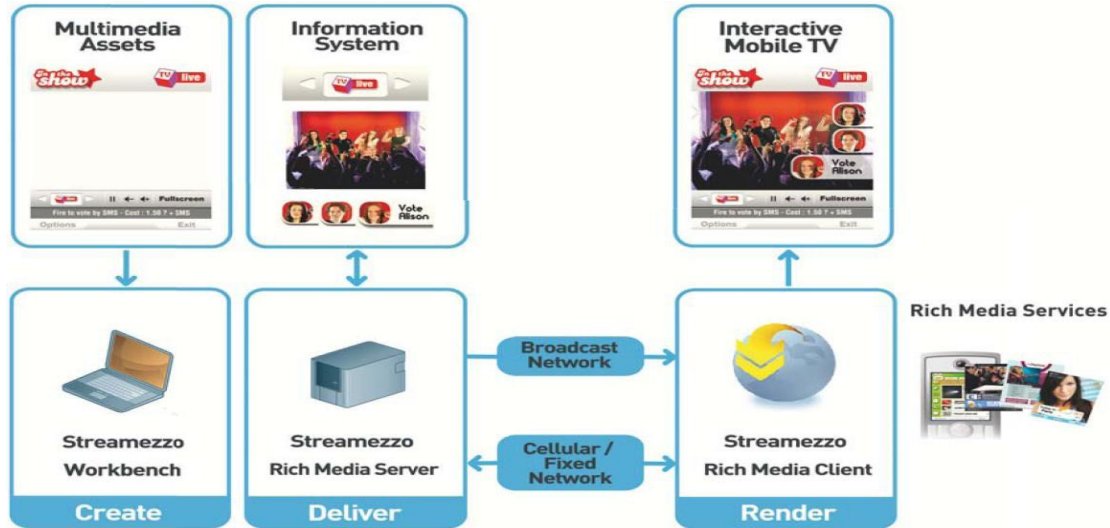


Figure 2.18: Streamezzo mobile TV solution architecture [91]

2.5.2.3 Limitation of the commercial solutions

From a review of several typical commercial solutions, we can realize that even though the service compatibility through different MDTV standards can be resolved in these solutions by developing proper compatible mechanisms, these solutions are however proprietary. Corresponding commercial entities own the copyrights of the products as well as the relative technologies and maintenance or update operations have to be under the entities' authority. This proprietary character may then prevent the commercial solution itself from achieving wider industry support and adoption. Moreover, a proprietary solution usually requires individuals with more technical expertise to develop the services using customized and specialized tools which render the difficulty and the complexity during the maintenance and evolution of the solution.

2.5.3 LOOKING FOR THE UNIVERSAL SOLUTION

Having discussed two categories of solutions for MDTV service creation and implementation on the application layer, we list all the pros and cons in the following table:

	Standard-based solution	Commercial solution
Pros	Standardized within its parent MDTV standard; easy to be followed and implemented by content providers, service providers, and terminal device manufacturers.	Straight and efficient solution to meet the MDTV service requirements.
Cons	Lacks of compatibility; hard to adopted through different MDTV standards.	Heavy proprietary that results in the raise of cost, life-cycle, difficulty and complexity throughout the software process of the three software targets.

Table 2.6: Pros and Cons of MDTV service creation and implementation solutions

Since both of these types of solutions have their pros and cons, there is a requirement for a universal solution that could join all the pros but prevent all the cons in order to cope with the aforementioned challenges in Chapter 1. Considering the standard-based solution, in order to prevent the poor compatibility, the universal solution would have to be compliant with most of the leading MDTV standards such as DVB-H, ATSC-M/H, ISDB-T 1seg, DMB and 3G. As to the commercial solution, in order to prevent it from being proprietary, the universal solution would have to be based on open-standards.

Due to the complexity of developing such universal solution, there are only a few related research pieces of work from the academic research field. One related work is a proposal of interoperable middleware architecture for digital broadcasting [98]. This architecture offers the flexible management of middlewares from different conventional DTV standards (e.g. DVB-MHP, ATSC DTV Application Software Environment, ARIB-BML, etc.) and is able to execute the various middleware applications that were developed for a specific standard. Even though this proposed solution is related to conventional DTV service, it provides a valuable reference for developing the universal solution. It concludes that by integrating the multiple middlewares, the incompatibility between different DTV standards can be reduced and even resolved. Regarding the Standard-based solution for MDTV service creation and implementation, this proposal could be helpful for improving the standard-based solution to become the universal solution.

Another related work has given a review of a system architecture and interactivity

model for Mobile TV applications [99]. The paper analysed the current common existing software platform of digital TV and mobile phones that are being used for the software applications and service development and also analysed the requirements of interactive mobile TV applications. Moreover based on the analysis of the results, the author made a comparison of several existing software platforms including MHP, DASE, BML and MIDP. An evaluation of such a comparison pointed out that Java MIDP has several advantages over other platforms to fulfil the requirements for achieving the interactive applications of mobile TV. Based on the results of that work, we can conclude that Java MIDP, as an open standard software system, is an ideal software platform for developing MDTV service applications as well as the service platform on the terminal devices such as mobile phones. This can be an inspiration when seeking the solution for preventing the proprietary problem in the aforementioned commercial solution.

Based on the literature review so far in this chapter as well as the related work in the research filed, we have realize that JSR272 could be one of the potential choices for a universal solution. JSR272 is currently under proprietary development but already a standardized MDTV middleware solution. It was developed under the Java Community Process (JCP) by a group of leading content providers, service providers and terminal device manufacturers such as Nokia and Motorola. It is agnostic in terms of the underlying MDTV standards such as DVB-H, DMB, OMA-BCAST, MediaFLO and TV-Anytime. All of these ensure that JSR272 is well support by the MDTV industry field and well-compliant with most of the MDTV standard; Developed upon the Java ME MIDP platform, JSR272 is natively transplantable throughout different mobile devices supporting Java ME MIDP and CLDC. Besides with the help of other JSR functional packages like Payment JSR229, Multimedia API JSR135, Scalable 2D Vector Graphics API JSR226, and newly released Scalable 2D Vector Graphics API 2.0 JSR287, JSR272 can meet most of the current service requirements such as ESG, purchase, DRM and A/V content play back. All of these ensure the functionality and adaptability of JSR272. Therefore when the JSR272

package is available as an open standard like most of the other JSRs from JCP, it would be well sufficient as the fundamental technology of a universal solution for MDTV service creation and implementation.

A functionality that is though missing and is not formally defined when JSR272 becomes the middleware solution is the service application model. Even though the DMB-MATE has defined its own application model based on Java ME MIDlet model, its MDTV service implementation follows a download-and-play structure that is probably not very suitable or reliable for fast, dynamic and interactive service applications under non-ideal network conditions. Besides, due to the professional character of developing MDTV service applications in the Java ME MIDP environment, the cost of the service creation is expensive as well as the higher technical knowledge and skill demands to the authors of a service.

2.6 SOFTWARE ENGINEERING FUNDAMENTAL

Since on the application layer of MDTV service, the service creation and implementation are mainly about application and software development and adoption, we can thus take the principles within software engineering as the theoretical reference. Within the scope of software engineering, three fundamental concepts have to be defined:

- **Software Engineering** is commonly defined as an engineering discipline which is concerned with all aspects of the practical software production.
- A **software process** is the set of activities and associated results that produce a software product.
- A **software process model** is a simplified description of a software process that presents one view of that process. The software process model may include activities that are part of the software, software products and the roles of people involved in software engineering.

One of the most widely used process model in current software engineering practice is

the waterfall model which is illustrated in Figure 2.19:

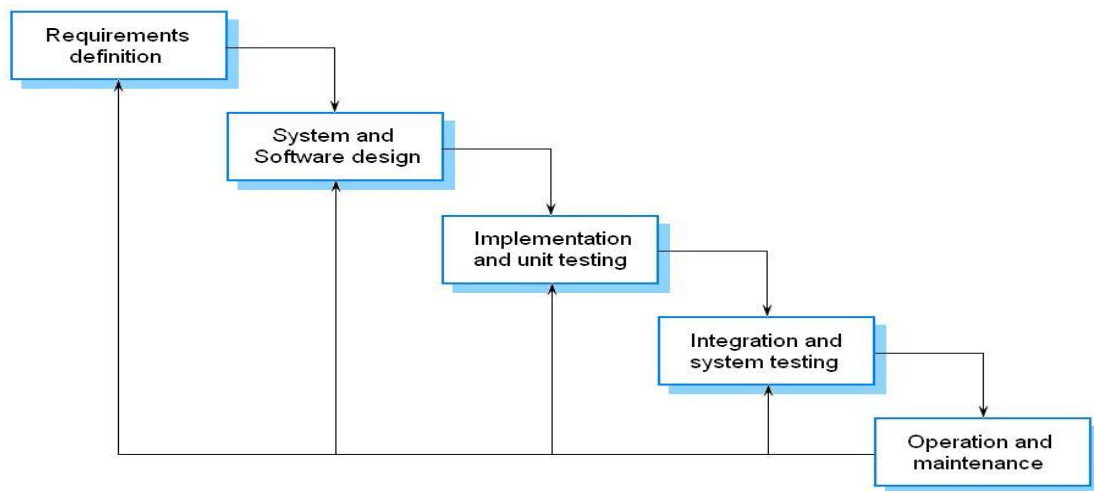


Figure 2.19: The waterfall software process model (adopted form [84] chapter 4 Figure 4.1)

- **Requirements and definition:** The system's services, constraints and goals are established by consultation with system users. They are then defined in detail and serve as the system specifications.
- **System and software design:** The system design process partitions those requirements to either hardware or software system and establishes an overall system architecture. Software design involves identifying and describing the fundamental software system abstraction and their relationships.
- **Implementation and unit testing:** During this stage, the software design is realised as a set of programs or program unit. Unit testing involves verifying that each unit meets its specification.
- **Integration and system testing:** The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, the software system is discovered to the customers.
- **Operation and maintenance:** Normally this is the longest process model phase. The system is installed and put into practical use. Maintenance involves correcting errors which were not discovered in earlier stages of the model, improving the implementation of system units and enhancing the system's

services as new requirements are discovered. In a waterfall process model, the following phase should not be started until the previous phase has finished and the last phase can involve any of the previous phases during the software system operation and maintenance [84].

2.6.1 SOFTWARE TESTING

During a software process, the software testing is an important part of work. As mentioned in the previous section, software testing exists in almost every step of a software process from implementation of software units and integration of system before the software product is delivered to the user, to the software maintenance after the software is deployed.

Speak within the software engineering scope, when source code has been generated, software must be tested to uncover and correct as many errors, also known as bugs, as possible before delivery of the software to the end-user. The main two goals of software testing are [126] – [131]:

- To demonstrate to the developer and the end-user that the software meets its requirements.
- To discover faults or defects in the software where the behaviour of the software is incorrect, undesirable or does not conform to its specification.

To conduct a software testing procedure, there are various principles and solutions according to the characteristics of the target software but essentially there are three main approaches that should to be considered: namely the test level, the test method and the test case design. The test level indicates the order of the testing targets and the affiliation between different targets; the test method defines which testing model and which testing tool and method one will employ to conduct a more efficient testing. Having selected the proper testing steps and testing method, the tester should design a series of test cases according to the software's features so that the target software can be tested more comprehensively to ensure any further improvement and evaluation.

2.6.1.1 Test level

	Description
Component testing	Component testing aims at exposing faults and further verifying that each component functions properly against its design requirement.
Integration testing	Integration testing is a systematic technique for constructing the software architecture while at the same time conducting tests to uncover errors associated with interfacing.
System testing	Function testing and performance testing will be conducted on the system testing level. Function testing evaluates the system to determine if the functions described by the software requirement specification are actually performed; performance testing is used to test the run-time performance of the software system within its context, with regards to the hardware and software in the end-user's actual working environment.
Acceptance testing	Acceptance testing is conducted with data supplied by the end-user rather than with simulated test data. Two main ways of acceptance testing are alpha testing , which is conducted by user with the software developer observing; and beta testing , which is conducted entirely by the user without the developer being present.

Table 2.7: Descriptions to software test levels

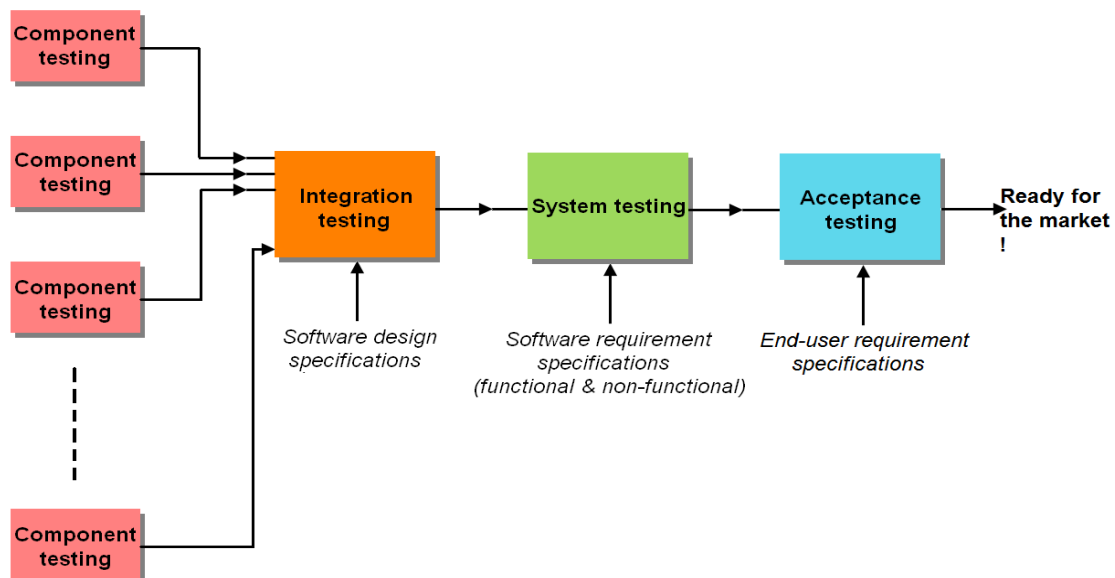


Figure 2.20: Reference software testing level flow diagram

2.6.1.2 Testing method

There are mainly two methods that software testers use to describe how they approach their testing. These are the black-box testing and the white-box testing. **Black-box testing** is conducted under the premise of knowing the specific function that a software has been designed to perform and testing is conducted to demonstrate that each function is fully operational while at the same time searching for errors in each function; **White-box testing** is conducted under the premise of knowing the internal workings of a software and testing is conducted to ensure that the internal operations

are performed according to specifications, and all internal components have performed effectively. The difference between these two testing methods is that black-box testing examines some fundamental aspects of a system with little regard for the internal logical structure of the software while white-box testing is predicated on close examination of procedural detail.

	Aims	Methods
White-box testing	<ol style="list-style-type: none"> 1. guarantees that all independent paths within a module have been exercised at least once; 2. exercises all logical decisions on their true and false sides; 3. executes all loops at their boundaries and within their operational bounds; 4. tests internal data structures to ensure their validity. 	Path testing; Condition testing; Data flow testing; Loop testing.
Black-box testing	Derives sets of input conditions that will test all functional requirements for a software in order to find: <ol style="list-style-type: none"> 1. incorrect or missing functions; 2. interface errors; 3. errors in data structures or external data base access; 4. behaviour or performance errors; 5. initialization and termination errors. 	Requirement-based testing; Equivalence partitioning; Boundary value analysis.

Table 2.8: Testing summary of white-box testing and black-box testing

2.6.1.3 Testing case design

The goal of the test design process is to create a set of test cases that are effective in discovering program defects and showing that the system meets its requirement [129]. The recommended procedure is to develop test cases using the black-box methods and then develop supplementary test cases as necessary with white-box methods [133]. According to the different test level that the testing target belongs to and the feature of the software target, the tester is able to choose the proper testing methods and achieve the testing case design. Moreover, some special components or architectures of a software system such as graphical user interface (including different types: software application GUI, web application GUI, etc), client/server architecture usually require a special software testing strategy and testing procedure.

Graphical user interface (GUI) testing: GUI is one of the most important parts of today's software, its correct execution is essential to the correct execution of the overall software and further to ensure the entire software system's robustness and usability. Therefore a GUI testing procedure is indispensable during the development

of the software's GUI. GUI testing requires that test cases are generated and executed on the GUI. Currently test cases may either be created manually by a tester or automatically by using a model of the software derived from its specifications. However most of these techniques for obtaining GUI test cases are resource intensive, requiring significant human intervention. The most common GUI test technique is **record-playback**. In this, a test designer interacts with the GUI, generating mouse and keyboard events and at the same time uses specialised tools to record the user events, capture the GUI session screenshots and then store the session. The tester later plays back the recorded sessions to recreate the events with different inputs to conduct the GUI test.

Client/server testing is required when there is a client/server architecture in a software system and due to the complex nature of such architecture, relevant tests usually demand more time and have higher costs. Commonly, the testing of client/server software system has several approaches namely as: application function tests, server tests, database tests and transaction tests.

Network communication tests: These tests verify that communication among the nodes of the network occurs correctly and that message passing, transactions, and related network traffic occur without error. Network security tests may also be conducted as part of these tests.

2.7 INTRODUCTION OF PROPOSED METHODOLOGY

2.7.1 REFERENCE MODEL FOR MDTV SERVICE CREATION AND IMPLEMENTATION (MDTV-SIM)

Based on the generic DTV service asset lifecycle of Section 2.2 and with reference to the system architectures of different MDTV standards as well as service commercial solutions, a reference model for MDTV service creation and implementation on the application layer can be illustrated as in Figure 2.21. The five entities in the chain are

the: Content Provider (CP), MDTV Service Provider (MSP), Broadcast Network Operator (BNO), Cellular Network Operator (CNO), and Terminal Device Manufacturer (TDM).

The key component in the model is the MDTV service application layer agreement. This agreement usually consists of most of the definitions in a MDTV middleware standard (e.g. DMB-MATE) such as service content (audio, video, and other auxiliary data) formats, service application model, application execution platform, application signalling, transport protocols, graphics model and security mechanism. Here this agreement is set as a standard-based rather than a commercial-solution-based one because of the agreement based on standards such as MDTV middleware standards (e.g. DMB-MATE) or MDTV service software enablers (e.g. OMA-RME) which is more normative and reliable. CP, MSP, TDM can therefore take this agreement as the reference during their work within the chain model.

CP is usually responsible for collecting MDTV service materials such as recording TV programs, providing assistant information and completing the initial productions to TV programmes; MSP on one hand is to used to collect MDTV contents from various CPs, further produce them to become MDTV services by using relative service authoring tools and distribute them to the end-user with the help of BNO. On the other hand, MSP is responsible to set up the server-end for the MDTV services it provides, and meanwhile to cooperate with CPs to actualize the implementation solutions for interactive applications. TDM has to follow the agreement and produce the MDTV terminal devices that are able to meet the MDTV service requirements within the chain such as capable of receiving services from BNO, capable of retrieving the services provided by MSP, capable of implementing any interactive applications by cooperating with CNO and MSP.

More precisely when CP, MSP and TDM are taking the application layer agreement, CP needs to produce all the audiovisual and auxiliary content based on the application model and content formats defined in the agreement. MSP refers to the application

model, application signalling and transport protocol definitions when adding or creating interactive functions in the service content passed on from CPs. TDM has to consider the requirements from CPs and MSPs when configuring the middleware and software elements in the terminal devices. An integrated software system will be designed according to the agreement. Several necessary additional applications such as service browser, ESG handler, plug-in value-added service enabler and specific media codec, will be developed and pre-installed by TDM in order to construct the MDTV service consumption platform.

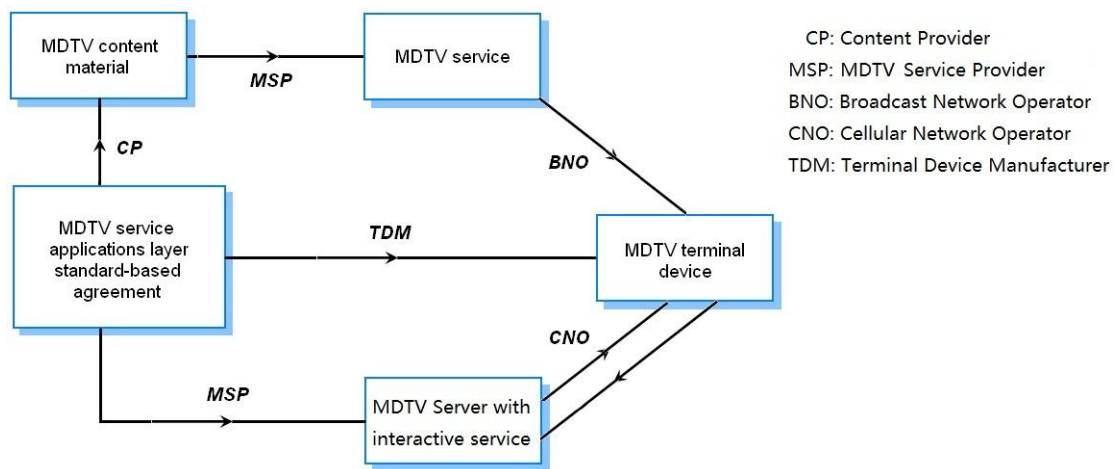


Figure 2.21: MDTV service implementation model (MDTV-SIM)

2.7.2 SOFTWARE PROCESS MODEL FOR MDTV SERVICE (MDTV-SPM)

Under the concept of software engineering and due to the universal characteristic of the waterfall model in the software industry, here we take the waterfall software process model (illustrated in Figure 2.19) as the macroscopic reference model for realizing the MDTV service creation and implementation on the application layer (illustrated in Figure 2.21). Once the waterfall model is adopted in the MDTV service implementation process, the MDTV service application layer agreement (in Figure 2.21) can be practically detailed as the MDTV relative middleware/software specifications. This agreement can further correspond to the requirements and definitions phase in the waterfall model in Figure 2.21. The remaining phases in Figure 2.21, which are MDTV content, service, service managing server and terminal

device, are all involved as target software in the waterfall model in Figure 2.19. The proposed process model is shown in Figure 2.22. However in the practical or commercial environment, this model can be modified in specific phases or enhanced by involving other assistant software process models.

In this model, **content and service** are treated as one target in the software process. The **service managing server** (server side) and **terminal device service platform** (client side) are the other two targets. **The contributions of this Thesis are mostly concerned with the second phase – the system and software design, and the third phase – Implementation, integration and system testing. Moreover, the methodology of our project will be stated under this model as well in the next section.**

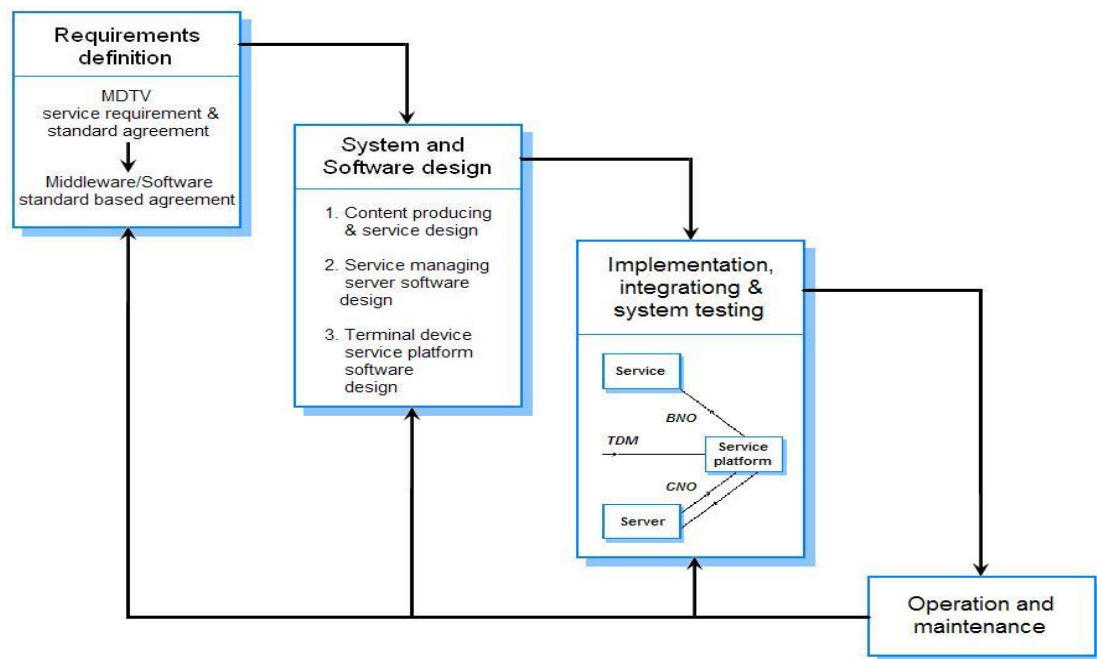


Figure 2.22: Software process model for MDTV service (MDTV-SPM)

2.7.3 INTRODUCTION OF THE PROPOSED METHODOLOGY

Regarding the requirement of developing a universal solution for MDTV service creation and implementation and after a discussion on related research works and potential technologies, **here we propose a MDTV service creation and implementation solution that has JSR272 as the underlying middleware standard.**

This proposed solution takes the MDTV-SIM model as the service implementation system reference and the software process model of Figure 2.22 as the software engineering reference, as illustrated in Figure 2.23.

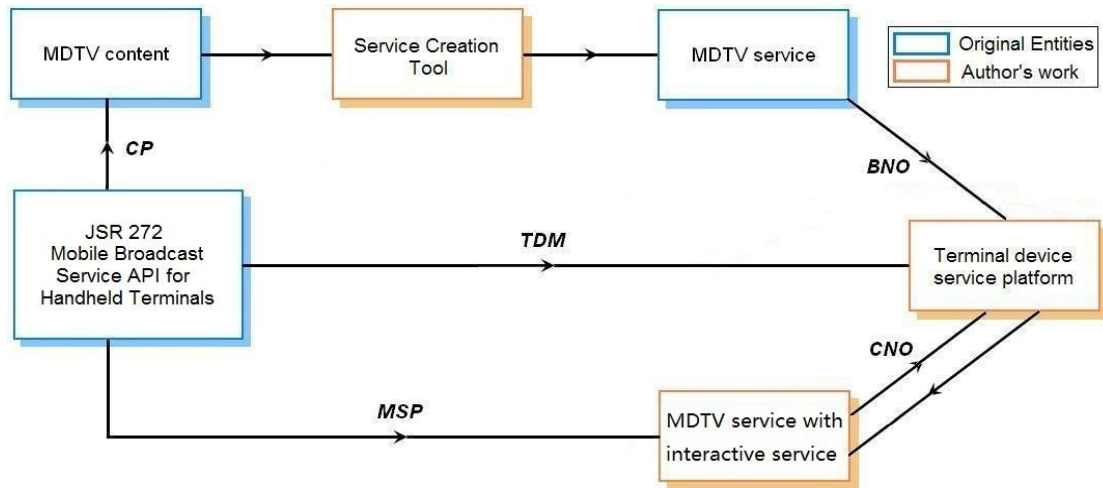


Figure 2.23: Thesis's proposed solution architecture

In the proposed solution architecture, we assume that all the MDTV service relative entities including CP, MSP and TDM have employed JSR272 as the fundamental elements of their middleware standard based agreement. Under this precondition, the boxes in orange in the figure are the author's work for the proposed solution. The author's work is the MDTV service creation and consumption system that mainly consists of a semi-automatic service creation tool and a service platform for the terminal device. The semi-automatic service creation tool stands between the MDTV content and service, functionally merging the works of CP and MSP when creating the MDTV service. The service platform is implemented with a MDTV server for handling interactive applications. This server is only a prototype version developed for testing purposes.

As discussed in previous sections, most of the current MDTV service creation and implementation solutions cannot cope well with the existing issues and challenges mentioned in Chapter 1: standard-based solutions suffers from the inter-compatibility issues when deploying services over different MDTV standards; many of the commercial solution are developed based on their existing products and technologies

thus their proprietary character cannot be avoided from the very beginning; even though most of the commercial solutions have adopted multiple MDTV standards as their development reference to ensure the solutions' inter-compatibility, they usually construct their own service creation and consumption environment to ensure their own commercial benefits. Such kind of solution usually requires additional knowledge and specific skills and therefore is not able to reduce the technical demands during the MDTV implementation. In contrast, through the proposed universal solution, the aims set in Chapter 1 are able to be addressed and the listed challenges can be resolved:

By introducing the semi-automatic service creation mechanism into the MDTV implementation process, the technical demands on the designers can be reduced. Without requiring to have specialised MDTV technical knowledge, designers in CP/MSP can easily create and modify the service contents semi-automatically through convenient operations such as drag-and-drop, rather than the traditional complicated programming environments. The MDTV service creation is thus demystified and becomes more efficient by using the proposed semi-automatic creation tool. Moreover, a novel service content presentation method based on open-source technologies (Java ME MIDP and XHTML) is also developed so that on one hand, it further reduces the technical demands during service creation process; since most of these open-source technologies are already familiar to the designers and commonly used in the IT field. This solution would encourage more design oriented and creative professionals to get involved in the MDTV service creation process. In addition to this, more open-standard and inter-compatible features will be introduced into the process, to enable the scalability of the service creation system, the inter-compatibility through different MDTV standards and also to prevent more proprietary issues.

The other main part of the proposed system, the terminal device service platform addresses the aims by providing a MDTV interactive-rich service consumption software environment. Its browser-based framework reduces the difficulty of the

service content creation for different terminal device screen sizes; being developed in Java ME MIDP further ensures it is extensible to underlying middleware (e.g. JSR272) and also its inter-compatibility through various device hardware and software conditions. In addition, a novel interaction handling method is also developed using the MIDP instead of ECMAScript with regard to reducing the complexity during the creation of the interactive services/applications on the head-end as well as processing them on the client-end.

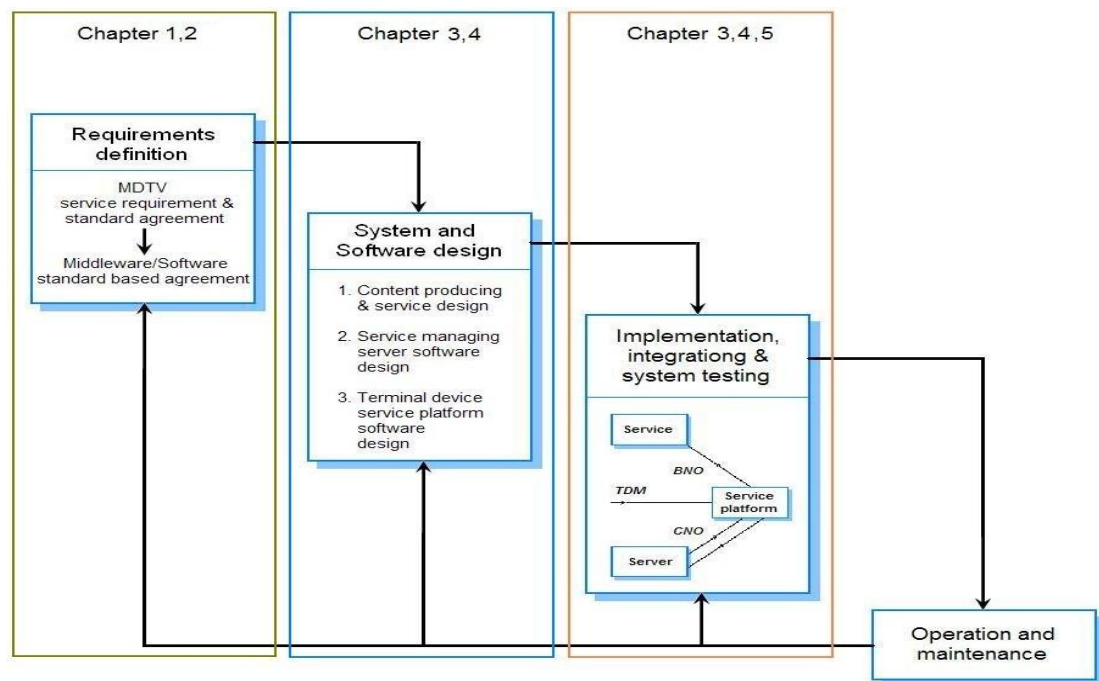


Figure 2.24: Thesis's organization under the MDTV-SPM model

To follow the software process in Figure 2.22, the Thesis has been organized as illustrated in Figure 2.24: Chapter 1, 2 present several pieces of research related to MDTV and its corresponding work and offers a discussion on why JSR272 has been chosen as the underlying the middleware standard in the proposed solution. The following two chapters, chapter 3 and chapter 4, focus on the “system and software design” phase and the “Implementation” phase in the MDTV-SPM model. Chapter 3 presents the semi-automatic MDTV service creation mechanism of the proposed system including a novel service presentation method. Chapter 4 describes the methodology and design of the terminal device service platform as well as an interactive service server prototype. Chapter 5 focuses on the “integration and system

testing” phase in the model, where a series of software testing procedures with corresponding evaluations will be conducted to the proposed software system.

One issue that needs to be highlighted is that since we still wait for the official release of JSR272 as an open-source MDTV middleware solution, we have designed our proposed MDTV service creation and implementation system to be as open as possible for JSR272 and treat the implementation of JSR272 functionality elements as part of future work.

Summary:

This chapter has offered a literature review within the MDTV service creation and implementation field, discussed the pros and cons of various current solutions and further concluded that there is a requirement of developing a universal service creation and implementation solution in order to prevent most of the existing problems. This chapter has thus introduced the proposed methodology in the end that to propose a novel MDTV service creation and consumption software system (going to be stated in Chapter 3 and 4) in order to address the Thesis’s aims

3. CHAPTER 3: PROPOSED SEMI-AUTOMATIC MDTV SERVICE GENERATION

3.1 MOTIVATION

As discussed in the previous chapters, the return channel network of most of the MDTV solutions, which is employed for the provision of interactivity, has not yet been fully utilized and exploited, resulting in a shortage of available MDTV interactive services, such as gaming, accessing additional information and shopping. A contributing factor to the limited availability of interactive MDTV services lies in the relatively isolated and hard to maintain service creation process.

Regardless the different types of current MDTV service creation solutions (including standard-based and commercial solutions), a common problem that contributes this situation has been identified as the high technical demands on the creators/designers during the MDTV service creation. Typically MDTV service creators/designers, are required to have very specialised technical knowledge and skills especially in the following three different areas: technical knowledge and skills for software development on mobile device, design and graphics skills on developing web-like content and applications, and technical knowledge and skills on developing MDTV services/applications based on different MDTV standards. This is due to the fact that a MDTV service is a product of the integration of digital TV service, mobile service and web-like services.

Therefore, to involve a creator/designer in the MDTV service creation process demands knowledge and skills from all these fields. More precisely, for mobile service development, a designer needs to know the capability of mobile devices and how to use mobile software technologies such as Java ME, C, and even technologies from commercial entities like Symbian C. For web-like service development, which is already widely adopted in Information Technology (IT) fields such as the World Wide

Web (WWW) and Electronic Commerce, a designer must be familiar with common web content design technologies including visual processing skills (e.g. how to use Photoshop and Flash), mark-up languages (e.g. XML, HTML and WML), and scripting languages (e.g. Javascript, ASP and PHP). Lastly, for digital TV service development, a designer also has to know and understand the relative middleware and software specifications of different DTV standards (e.g. 3G, DVB-H, DMB, ATSC-M/H).

As a result, when developing a MDTV service, such a high demand on the background knowledge and skills hinders creative and non-technical individuals and professionals from making a quick start on the service creation resulting to a considerable amount of training to be required. Those creative and professional designers in the IT field are therefore discouraged from getting involved due to the gap in the technical knowledge and very specialised skills required. This situation contributes further to the lack of human resources in the MDTV service creation field, hindering in turn the development of advanced MDTV services creation.

Moreover according to the analysis and discussion in the literature chapter, none of the two types of solutions, including standard-based solution and commercial solution, are capable of resolving the existing challenges effectively. Standard-based solutions suffer from inter-compatibility issues among different MDTV standards due to their different technical backgrounds. As to the commercial solutions, even though the commercial entities manage to cope with the technical gap by offering themselves methods along with the self-developed proprietary development toolkits, the expense of such kind of solutions are usually high and the performance of the final outcome are not that satisfying. For instance if we take one of the most popular commercial solution, the Streamezzo solution (Section 2.5.2.2), even though the solution supports multiple MDTV standards such as 3G, DVB-H, DMB and MediaFLO, almost all of its MDTV service implementation enabling segments, including the content/service creation, MDTV service application server and terminal device service platform, are

under the authority of Streamazzo. Also, only basic interactive applications (e.g. VoD, basic voting and purchasing and PVR) are supported in this solution. Besides, their core technology LAsER could increase the difficulty in updating and maintaining the service content. For example, since the Rich Media technologies like LAsER does not support well the different screen sizes of the diverse MDTV terminal devices, the service content layout has to be designed several times for each different possible screen size. This repetition of work results in extra time and resource expenditure.

Motivated by this situation, the semi-automatic service creation process is developed as the first component of the proposed MDTV service creation and consumption system, with regards to provide solutions to all the aforementioned problems. Figure 3.1 illustrated the main methodology. The main idea of the proposed semi-automatic service creation process is to integrate a “service creation tool” in the original creation process, which can assist the relative entities (such as CP and MSP) to generate interactive-rich MDTV services and applications semi-automatically. This in turn results in more creative professional such as designers to be involved in the MDTV service creation process, since the semi-automated service creation tools deals with the complexity and technical aspects of the MDTV service creation. Meanwhile a new MDTV presentation method is adopted to ensure the corresponding creation outputs (such as an interactive-rich MDTV service page) are open-source and inter-compatible with most of the MDTV standards.

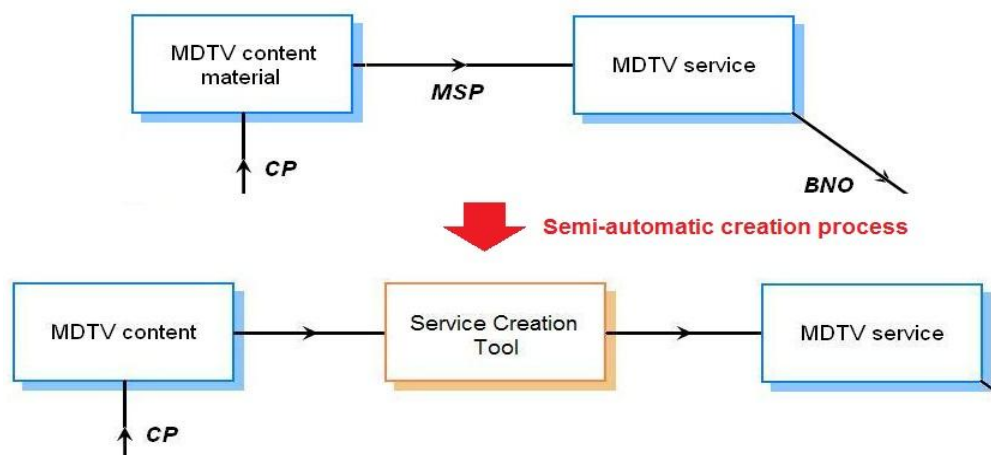


Figure 3.1: Semi-automatic service creation process

3.2 MDTV SERVICE PRESENTATION DISCUSSION

A typical MDTV service consists of Audio/Video data and auxiliary data service, whereas the auxiliary data service contains content in multiple forms such as text, image, animation, and interactive applications like voting, gaming, or chatting. These are created against different service requirements. To present the MDTV service audio-visual components and interactive applications through the I/O units of a MDTV terminal device such as monitor, keyboard and screen pointer to the service consumer, a service presentation mechanism is necessary. With reference to the literature on the middleware and software layer of the DTV/MDTV service (as presented in the previous chapter), such presentation mechanism usually consists of the application model that defines the format, the implementation method of the service application, and the application execution platform that provides an execution environment for the interactive MDTV applications.

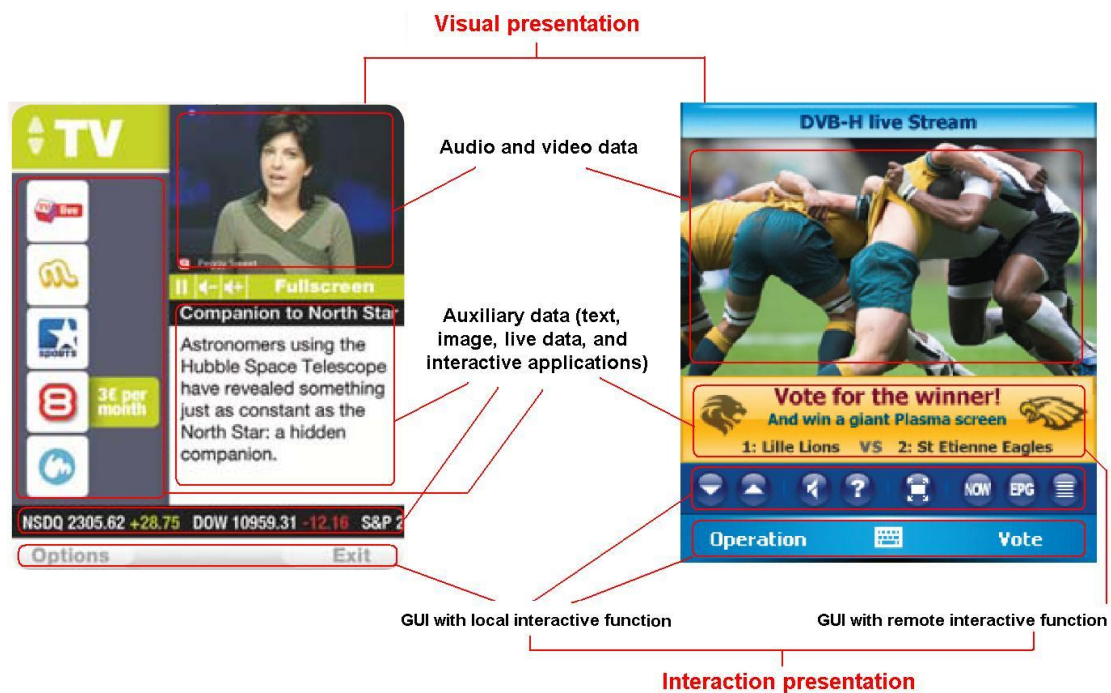


Figure 3.2: Typical MDTV service components (marked in black letters)

3.2.1 MDTV APPLICATION MODEL

As discussed in the previous chapters, most of the MDTV services have their own

application model as the reference for **MDTV service creation** (including content creation and service development), and **MDTV service implementation** (including service rendering on the terminal device and the implementation of interactive service applications). A typical application model defines the following items:

- The application format;
- The application lifecycle;
- The implementation method of the interactive application (how to get, store, run, update and remove the application in the terminal device).

The application format is the first item that needs to be defined as it determines which software application development language and tool are to be employed, what programming language and APIs need to be used during the creation of applications, and how to retrieve those applications in the service terminal device. The definition of the lifecycle and the implementation method of the service application both depend on this format. As Table 3.1 illustrates most of the DTV standards adopt GEM as the middleware solution, of which the application format is DVB-HTML and DVB-J and some standards involve ECMAScript as an optional choice. For MDTV, even though there is no middleware defined in the DVB-H standard, most of the commercial solutions tend to choose rich media as the application format, whilst 3G has adopted OMA-RME as its native application environment for rich media application and ATSC-M/H also selects its middleware based on the same environment. DMB is an exception in that it chooses Java ME MIDlet as their application format.

	DTV			MDTV			
Standard	DVB-T/S/C	ATSC-T/S/C	ISDB-T/S	DVB-H	3G	ATSC-M/H	DMB-T/S
Middleware	GEM			Commercial solution	OMA-RME	OMA-RME based	DMB-MATE
Application model	DVB-HTML, DVB-J Xlet, (ECMAScript)			Commercial solution	Rich media	Rich media	MIDlet

Table 3.1: Application models of DTV and MDTV standards

Therefore within the MDTV service scope, two types of application formats are being adopted: rich media and MIDlet, with each of them leading to two different application models. Herein this chapter, as well as in the Thesis, we refer to them as the “stream” model and the “let” model.

The “stream” model

The MDTV service application in the “stream” model is usually delivered in a stream and a browser-based service platform, which is set in the terminal device for consumption purposes such as browsing and user interaction. A “stream” application model usually supports download-and-play and progressive rendering while downloading. This means that the application can either be downloaded first to the terminal device and run, or can be rendered and displayed on the service platform browser progressively while being downloaded. Here we take the rich media and BML as the examples.

The rich media model is used by 3G, ATSC-M/H and is known as the OMA-Rich Media Environment (OMA-RME). The RME addresses enhanced rich media services and those services often include service aggregation of various kinds of content in a single interface (graphics, text, audio, video), and service implementation methods based on client-server real-time interaction. The MDTV is a typical example of this enhanced rich media service. Even though the OMA-RME does not actually define the application model, all the key points of the application model are defined in the standard. The MDTV service application in the OMA-RME model is defined in a **format** of rich media: employing SVG Tiny for the visual presentation and the ECMAScript for user interaction. The service applications can be distributed either through the MDTV broadcast network in the broadcast stream, or the return channel network (cellular network) in the point-to-point stream. The **application lifecycle** depends on its bound service, which means that the application starts once the service is launched on the terminal device, stopped and destroyed once the user exits the application. The MDTV service **implementation method** in the OMA-RME is based

on the RME system, consisting of the RME server and the RME client. The RME server is the rich media data source and support for implementing user interactions in the RME client. The RME client has the capability to display MDTV service application in RME format as well as handling interaction with the client itself (local interaction) and with the RME server (remote interaction). Besides, the actual form of a RME client can be a pure rich media browser or an integrated XHTML browser with enhanced functionality [54] [100 [96].

The ISDB application model consists of the ARIB-J model extended from GEM and BML document model. The BML document model is defined in the ISDB STD-B24 “XML-based multimedia coding scheme” specification and can be adopted in the ISDB MDTV field. The ISDB MDTV service content are defined in the BML documents with the help of DOM and CSS for the purposes of visual presentation, and ECMAScript is employed for handling the user interaction. The MDTV service **implementation method** in the BML model is called the Interruption Event Model. This denotes that the interactive applications are in the form of an “Interruption Event” in the BML document browser of the terminal device and each interruption event corresponds to a respective event handler locally or remotely that could execute the required functions [29].

An example of this in the stationary DTV field is the DVB-HTML application model in DVB MHP.

The “let” model

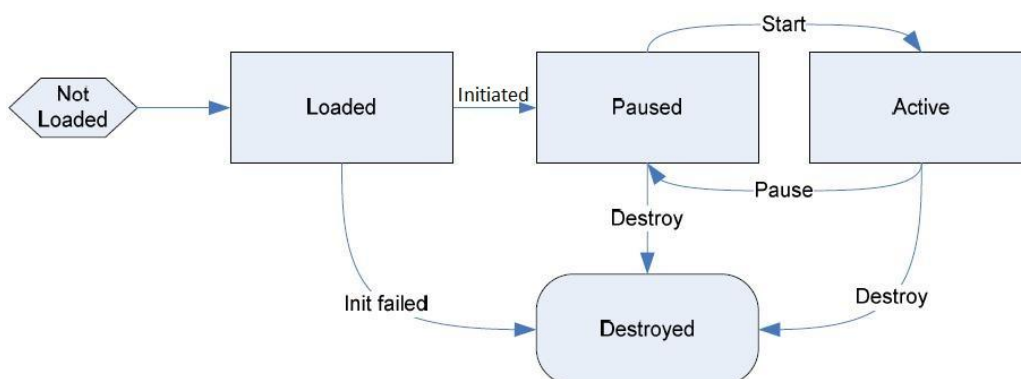


Figure 3.3: DMB application lifecycle [75]

The MDTV service application in the “let” model is usually delivered in a specifically defined application program package and the corresponding execution platform is set in the terminal device for the execution of the MDTV service application. A “let” model based service application does not support progressive rendering while downloading, and it must be initially downloaded to the terminal device before launched on the execution platform. Here we take MIDlet as a representative example.

The MIDlet model is used by DMB and defined in the DMB middleware standard DMB-MATE. Since this MIDlet model is based on the Java ME MIDP, the application format is also based on MIDP. “Lowest Common Denominator User Interface” (LCD UI) API or SVG Tiny API are employed for the visual presentation of the content and MIDP along with other assisting JSRs such as Mobile Media API (MMAPI) are employed for the user interaction. The MIDlet application consists of an “application module”, which is also the basic unit for the transport of an application through either the broadcast or communication network. The application lifecycle as illustrated in Figure 3.3 is also similar to the MIDP MIDlet lifecycle. The application implementation method is based on the application execution platform, the Java Virtual Machine (JVM) in Java ME MIDP. All the MDTV service application MIDlets run on top of the JVM and DMB manages to enable the simultaneous execution of multiple applications rather than only one in contrast to the case of Java ME JVM [75] [120].

A corresponding format to the above example in the stationary DTV field is the DVB-J Xlet in the GEM standard.

Application Model in commercial solutions

In the MDTV service commercial solutions, the functionality of the Application Model is also defined. Those proprietary Application Models (as illustrated in Figure 3.4) are usually based on Application Model specifications of different MDTV

standards and encompass additional technologies (such as Symbian C SDK from Nokia) as well as some additional functions.

APPLICATION MODELS IN MDTV COMMERCIAL SOLUTIONS		
Commercial MDTV service solution	Compliant MDTV standards	Application model bearer
Nokia MBS	DVB-H, DVB-IPDC, OMA-BCAST	Nokia proprietary SDK
EXPWAY FastESG	DVB-H, 3GPP, ATSC-M/H, OMA-BCAST	BinXML, BinSVG owned by EXPWAY
ACCSEE NetFront	DVB-H, ISDB-T lseg, OMA-BCAST	FastESG based or MDTV standard based
Streamezzo	DVB-H, DMB, MediaFLO	Rich Media (LAsER)

Figure 3.4: Application Model in commercial solutions

3.2.2 MDTV SERVICE PRESENTATION

As discussed above there are two types of application models in the MDTV field. The “stream” model is more reliable as the service stream can be progressively rendered and displayed while downloading, which reduces the error risk and the latency during service consumption. Whilst the “let” model is not as reliable for fast, dynamic and interactive MDTV service consumption under non-ideal network conditions, where applications based on this model must be completely downloaded before they could be launched for use. As a result of this, here we choose the “stream” model as the reference model during the proposed semi-automatic service creation process.

Under the definition of the “stream” Application Model, there are many technologies being applied according to different MDTV service design requirements. Refer to the functionality of the MDTV service application format, these technologies can further be categorized into two functional groups: visual presentation technologies and interaction presentation technologies. As mentioned in the last section, the visual presentation technologies are applied to format and present all the visual components of the MDTV service whilst the interaction presentation technologies enable the interactive components and applications of the MDTV service and implement the interactions between the service and the user. Including the Rich Media technologies,

several candidates for each category are to be discussed in the following section.

3.2.2.1 Visual presentation technology candidates and discussion

XML and MDTV service presentation

Extensible Mark-up Language (XML) is a simple text-based format for representing structured information [117] and it is the basis of many popular standards such as the Really Simple Syndication (RSS), Extensible HyperText Mark-up Language (XHTML) and SVG. XML is widely used as a description language in different layers of the MDTV service, regardless of the underlying standards. The Electronic Service Guide (ESG), as a key component of MDTV service, is a good example of the XML format adoption in the DTV field. However, basic XML is designed to transport and store data rather than designed for displaying data like HTML or XHTML. In order to display the content in a XML format, there is always need of a specially-designed presentation engine to assist in parsing and rendering the data [118].

So far there are some cases/products that employ XML as the visual presentation technology such as XUL from Mozilla, UIML from Oasis, [111] and [112]. Those new versions of XML are usually customized by the developer for specific requirements. In turn, the authoring tool as well as the rendering engine on the client side has to be designed for each specific customized XML description language. This results on one hand on specific solutions that cannot be accepted universally and on the other hand requires extra resources for implementation across different platforms. However in most cases in the MDTV service creation, XML is used only for the purpose of transporting and storing data, whilst several advanced XML-based languages like the Synchronized Multimedia Integration Language (SMIL), SVG and XHTML are used for the visual presentation and rendering of the service.

Regarding the MDTV service presentation, the terminal device service platform is responsible for rendering and displaying the service contents including the ESG, A/V and auxiliary multimedia data. Typically, the relevant audio-video codec, such as

MPEG-2, MPEG-4, etc, are used for the rendering and display of the audio-visual material, whilst Rich Media and XHTML are used for the visual presentation and graphical user interface (GUI) of the service content and Document Object Model (DOM), Synchronized Multimedia Integration Language (SMIL) and ECMAScript are adopted as the assisting technologies to Rich Media and XHTML for handling the service synchronization and the interactivity during the service presentation.

Rich Media technology

Scalable Vector Graphics (SVG) [113] is defined for describing two-dimensional graphics in XML format and SVG Tiny is its sub-version for providing the ability to create a whole range of graphical content, from static images to animations to interactive Web applications on small devices like PDA and mobile phones. As one of the key components of Rich Media, SVG Tiny is also widely used in MDTV standards such as 3GPP, ISDB 1seg and ATSC-M/H and it is also used in a number of research [103] [104] [105] and commercial solutions on MDTV service implementations such as the 3GPP-DIMS, MPEG-LASeR, MORE, Streamazzo and EXPWAY FastESG.

Flash [114] is a proprietary Rich Media standard owned by Adobe that can manipulate vector and raster graphics. **Flash Lite** is an optimized version of Flash for presenting A/V and interactive elements on the mobile and portable devices. Commercial MDTV service implementation solutions such as the NetFront MDTV solution have employed Flash Lite as its GUI technology.

Even if Rich Media technologies provide excellent graphics and animation experience, a Rich Media featured MDTV service platform suffers from several drawbacks in both the browser-based and the rich media based cases. These are summarized below:

- **Incompatible scene size.** Although by using Rich Media, the service interface or the platform interface can be designed easily to fit different device screen size, each individual solution is only suitable for one screen size resulting in that a

MDTV service has to propose different Rich Media UI solutions regarding the different types of terminal device screen sizes and resolutions. This may lead in additional development work on UI design and service layout design and in turn result in the increase of project costs and the complexity of system maintenance and update.

- **Requirement on using specific/customized authoring tools and advanced technical knowledge.** Flash Lite is a proprietary Rich Media format that needs the end-to-end proprietary development and deployment including the underlying technology, authoring tools, server and the Adobe Flash Player client. Although SMIL and SVG can be edited in a plain text-editor, the corresponding authoring tools are usually necessary for an efficient and effective design. However, three main sets of Rich Media technologies, MPEG4 BIFS, MPEG LAsER and SVG Tiny, which are currently used in the MDTV service development, offer different graphic features and in turn they are incompatible and require different sets of authoring tools [119]. Specialized professional knowledge is thus needed when using either of them. Moreover there are no promising Rich media authoring tools suitable for MDTV service and MDTV UI design and most of rich media based service development still depends on proprietary and customized tools. The situation becomes worse when considering the different kinds of MDTV standards in that there is hardly any universal solution.

XHTML

Other than Rich Media, the **Extensible HyperText Mark-up Language (XHTML)** [116] is another option for the visual presentation of a MDTV service and the GUI design. Derived from HTML, XHTML has been defined as an XML application with a stricter and cleaner syntax than HTML. The motivation for XHTML Basic is to provide an XHTML documents type that can share across communities from desktop to consumer devices with limited resources such as mobile device, and that is rich enough to be used for simple content authoring.

As to the MDTV service industry field, many leading MDTV standards such as 3G, ISDB 1seg, and DMB have employed XHTML as an optional element of their application model. With the same motivation of choosing JSR272 as the underlying middleware specification, here we choose XHTML as the service presentation technology so as to seek a general universal solution for enabling a compatible MDTV service visual presentation solution throughout different MDTV standards.

Comparing with Rich Media, the XHTML based MDTV service content have similar characteristics with the classic web service. Along with the browser-based terminal service platform, an XHTML MDTV service UI and layout is more flexible to fit in any device screen size with different resolutions. This means that as long as the service platform is compatible with different sizes, the MDTV service that is in XHTML format is relatively flexible with regards to the layout design. This advantage saves further time and cost in a MDTV service developing life-cycle and simplifies service maintenance.

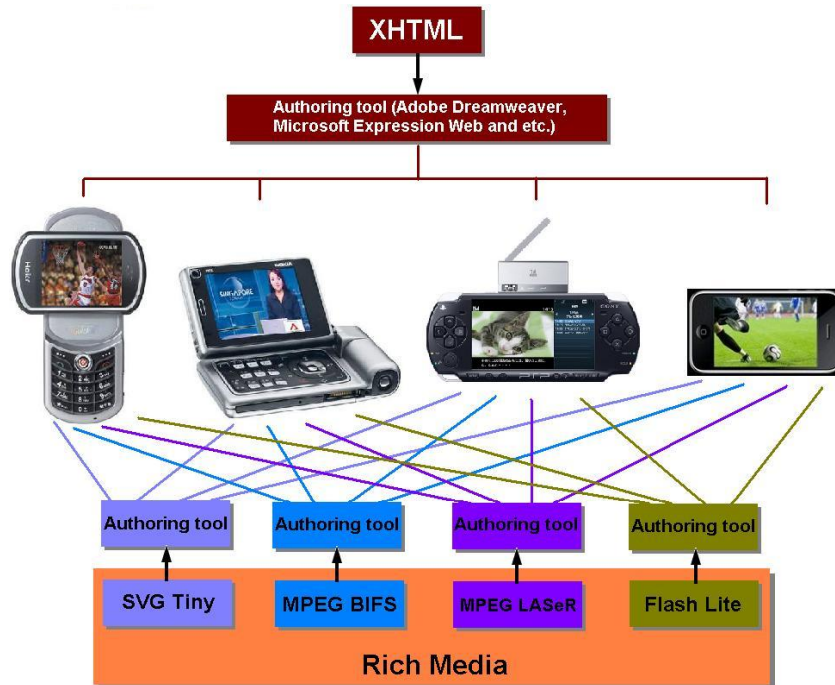


Figure 3.5: Comparison of authoring tools between XHTML and Rich Media

Due to the popularization of XHTML technology in the IT field, the selection of the authoring tool for XHTML is also much easier when compared to Rich Media.

Software such as Microsoft Expression Web and Adobe Dreamweaver, as well as several other freeware tools offer sophisticated SDKs for XHTML based design. Moreover, designers and developers that are experienced and active in web service design can be attracted and involved in the MDTV service design for an easier and more efficient development process. Thus this enables large numbers of designers in the web design field to design services for MDTV as well. Besides, with the help of the proposed service creation tool that handles different features (for example the service signalling mechanism), the service creation becomes a standard-independent and service-independent and a universal MDTV service presentation solution.

The main drawback of XHTML when comparing with Rich Media is that XHTML originates from the Desktop PC world and it has not yet been fully adapted to the mobile device environment. Furthermore, due to the complexity of such adaptation work, a few of promising results have been achieved especially in the MDTV service creation field. [102]

3.2.2.2 Interaction presentation technology candidates and discussion

ECMAScript is a widely adopted script language for handling and facilitating the interactivity between service content (e.g. web service page content) and users. Several famous script languages such as Java Script, Jscript are the implementations of ECMAScript. It was firstly adopted as the web scripting language that enlivens web pages under the web-based client-server architecture. Many scene presentation technologies such as HTML and Rich Media (e.g. LASeR) employ ECMAScript as assistant. Moreover, most of the DTV and MDTV standards have involved ECMAScript as the interaction presentation technology in their corresponding middleware/software specifications (e.g. DVB-GEM, 3GPP-DIMS, OMA-RME, DMB-MATE, and OMA-ESMP). [101]

Most of the MDTV standards (ATSC-M/H, 3G, and DMB) have employed scripting languages such as ECMAScript in their service for the implementation of interactive

functionality for service applications. In the application model of those MDTV standards, a scripting language is usually used to assist markup languages like XML, SVG or XHTML since these markup languages are not designed to support the following capabilities:

- The ability to apply mathematic and procedural logic locally to document data;
- Providing access to facilities of the device such as messaging function on a phone;
- The ability to generate messages and dialogs locally, reducing the need for expensive round-trip for alerts, error messages, confirmations etc;
- The ability to handle events;
- The ability to allow the dynamic creation and/or modification of documents on the client [101];

However due to the review of MDTV middleware technologies in the previous chapter (Section 2.5.2.1), **Java ME**, with the excellent native compatibility and supported by many industrial mobile device manufacturers such as Nokia, can be an ideal choice other than ECMAScript that all these abilities above can be handled by programs developed using the MIDP API. Besides if a scripting language is involved in the application model, its corresponding scripting code parsing mechanism is needed in the terminal device service platform, which means additional resources including processing threads, memory and power that are going to be required. This approach is against the common software development strategy on the limited capability of a mobile device.

3.3 SEMI-AUTOMATIC MDTV SERVICE CREATION

3.3.1 SEMI-AUTOMATIC SERVICE CREATION TOOL

It can be concluded from all the previous discussions that the current MDTV service creation solutions based on the corresponding service Application Model and selected service presentation technologies have illustrated a series of problems and drawbacks

such as weak inter-compatibility and high cost. It can also be realised that selecting more suitable presentation technologies such as XHTML and Java ME may be an effective way of making some improvements to this situation. Thus when creating the MDTV service by using these two technologies, the XHTML page will be the basis of the MDTV service page visual presentation components whilst the Java ME will be used for constructing the interactive applications. However in this case, even though these two technologies ensure the corresponding MDTV service to be open-source and less proprietary, knowledge and skills related to XHTML and Java ME are still required on the designers' end and some programming work may be needed. On the other hand, most of the MDTV standards have defined their own metadata formats and functional API (for example: the URI for audio/video streaming) for service implementation. Therefore during the service creation, relevant metadata or functional code for various service applications (e.g. interactive application) need to be added in the XHTML code according to the design requirement. This part of the service creation may also need extra programming work, where the additional professional skills and knowledge (e.g. programming skills and MDTV middleware knowledge) are required.

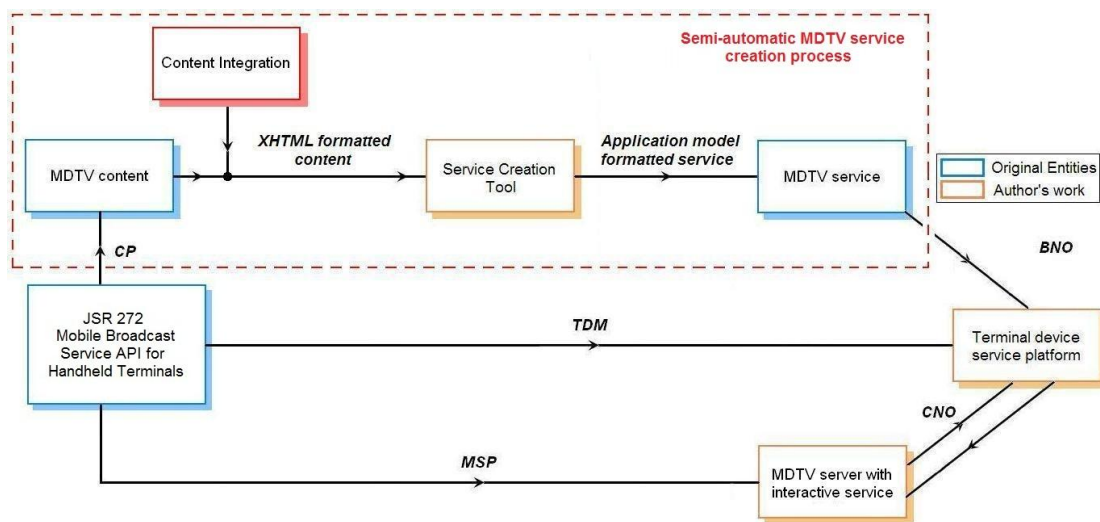


Figure 3.6: Semi-automatic service creation process

As a result, even though the technical demand during the service creation has been reduced by choosing XHTML and Java ME as the presentation technologies, the

creation process is still too manual with low efficiency. This therefore motivated the introduction of the semi-automatic creation tool, which is the core software component in the proposed semi-automatic service creation process. Service editors can use this tool to further manipulate the XHTML pages, make modifications and add necessary MDTV features (e.g. metadata, functional commands or interactive applications) to the pages, and finally generate the integrated MDTV service pages more easily without the need to do any programming themselves.

In the MDTV service implementation entity chain, the MDTV content creation is usually done by the Content Provider (CP), and the service creation is usually done by the Multimedia Service Provider (MSP) (refer to MDTV-SIM model mentioned in section 2.7.1). By utilizing the proposed semi-automatic service creation tool, these two creation processes can be merged into one aggregation as illustrated in Figure 3.6. In this case, the MDTV content/service creation may be performed by only the CP. The content integration is an aggregation of the technical works and the relevant professionals that integrate the MDTV content and format the content into XHTML pages. More precisely, after the MDTV content have been collected and post-produced by the CP, the designers in the content integration segment will firstly format these contents into XHTML code. This process is relatively open to web UI and service designers, enabling them to easily design service UIs that include text, graphics, links and layout information on popular commercial and professional XHTML authoring tools, such as Dreamweaver. A service designer can then use the proposed semi-automatic service creation tool to import and post-design the produced UIs and service layouts, as well as add MDTV features (URIs, interactive applications, and service metadata for ESG) into the XHTML code to eventually transform them into a MDTV service ready for implementation. Therefore, without the need of in-depth MDTV or software engineering background, a service designer can add from a library of predefined applications, actions related to the designed MDTV interactive service on any element on the UI in a drag-and-drop manner. Thus instead of reprogramming from scratch every time a new application/service has to be created,

the service creation tool helps designers and creative professionals to construct or modify MDTV interactive services semi-automatically, which results in a faster and more effective service development lifecycle.

On other hand and with regards to the functionality of the proposed creation tool during the MDTV service provision, we have taken the “Content and service creation and management sub-system” developed in the INSTINCT (IP-based Networks, Services and Terminals for Converging systems) Project 2006 (Figure 3.7) as the reference of our creation tool development.

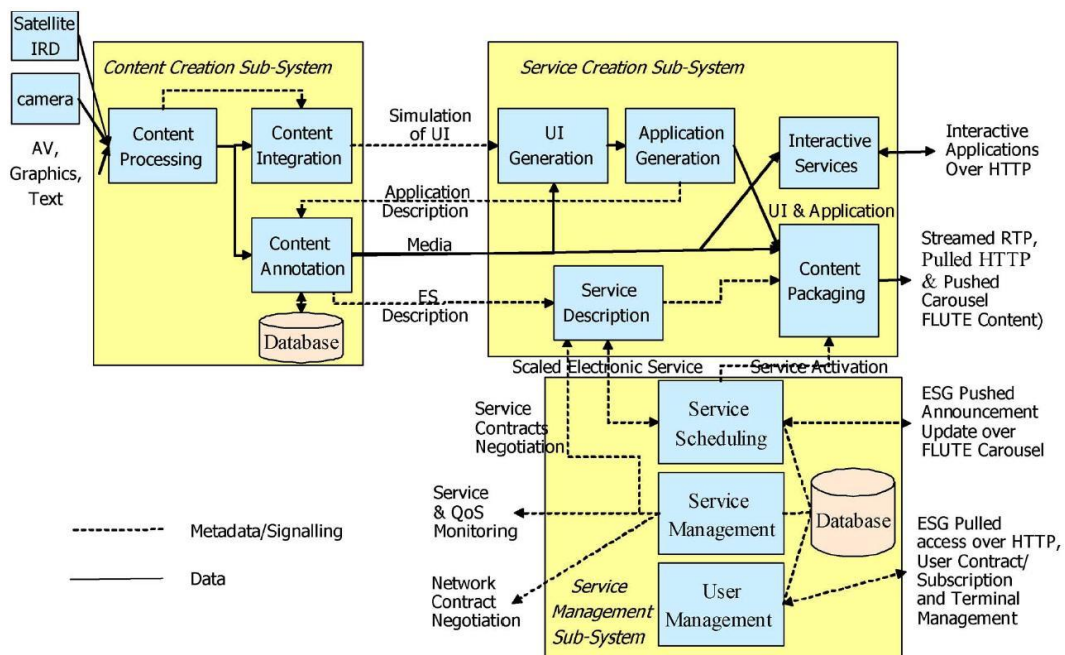


Figure 3.7: INSTINCT Project Content and service creation and management sub-systems [55]

INSTINCT [110] is a European project in line with the objectives of DVB Convergence of Broadcast and Mobile Services (DVB-CMBS). It is committed to assist DVB in realizing the commercial provision of convergent service in mobility with special focus on the DVB-T, DVB-H and DVB-MHP standards in conjunction with the concept of wireless communications networks (notably GPRS and UMTS) combined with terrestrial DVB broadcast networks. INSTINCT aims to a carrier grade full specified and open final platform for the delivery of convergent services in collaborating wireless communications and terrestrial broadcast networks. A detailed system architecture has been developed by the project according to the project’s aim

and the “content and service creation and management sub-systems” are the components that cover most of the mechanisms and functionalities during the service creation phases. Moreover the construction of this sub-system is very similar with the architecture of the Thesis’s proposed MDTV service creation solution. Thus this “content and service creation and management sub-system” is chosen as a reference and a theoretical support to the Thesis’ proposal.

Comparing this system with the process in Figure 3.6, the component correspondences are expressed in Table 3.2. A core difference between these two architectures is that in the INSTINCT project the content integration and UI generation are separated segments whilst in the Thesis’s proposal those parts of work are done in a single step. This is because in the INSTINCT project, the two segments are using different presentation technologies: XML is adopted in the content integration session for it is a default metadata format in DVB-H; during UI generation, some other visual presentation technology (for example: HTML) is applied since XML is not capable of presenting visual components of the service directly. In this case, some extra conversion work needs to be done between the XML formatted service content and the actual service page. In contrast, the Thesis’s proposal has chosen XHTML as the presentation technology throughout the entire service creation process. Therefore the service editors can use XHTML authoring tool to produce the UI components at the same time when the content is integrated. Thus based on the service creation methodology of the INSTINCT project, the Thesis has facilitated the original creation process and further contributed a novel semi-automatic service creation solution.

Correspondence		Functionality Description
INSTINCT	Creation aggregation	
Content Processing	MDTV content	Content collection and processing for MDTV provision
Content Integration & UI Generation	Content Integrator	Content integration and UI editing with reference to the application model
Application Generation	Service creation tool	UI modifying and Interactive application generation referring to the application model
Content Packaging	MDTV service	MDTV service packaging process

Table 3.2: Components correspondence

The rest of the components in the INSTINCT system are currently out of our scope and can be considered as the further implementation objectives of our project in the future.

3.3.1.1 XHTML and Java ME based MDTV service presentation

During the semi-automatic service creation, normal editing work like XHTML page manipulation or adding MDTV service related metadata into the page can be achieved by using basic functions of the proposed creation tool. This type of edition is based on basic XHTML knowledge. On the other hand, when there are requirements for adding interactive applications to the components, within the XHTML service page, the manipulations (of which the operation instruction will be presented in Section 3.3.2) are based on a new method called “ID Event” developed along with the proposed software.

ID Event

In order to arrange a more efficient interaction implementation method based on Java ME MIDP and JSR272, we propose the “ID Event” method as an alternative solution instead of ECMAScript. This model is enabled by MIDP and a special interactive event ID. By using MIDP as well as those assisting APIs, we have developed a novel interaction implementation mechanism based on the proposed terminal device service platform to handle most of the functionalities that scripting languages can provide (which will be discussed in details in Chapter 4). By using the special interactive event ID, we bridge the gap between MIDP and interactive application events, where the service browser can handle the events by passing them to the corresponding event handlers developed in MIDP according their ID value. The event ID is integrated into the XHTML code by adding a special ID attribute rather than requiring professionals to manually code it using scripting languages, which in turn increases the service creation and rendering efficiency.

An ID in the “ID Event” method is in the form of a string and its format is defined as follows:

Private ID@@cmd**Application serial No.n**@@**Application title**@@cmd**Parameter serial No.v**@@**Parameter value**@\$c**Application serial No.cmd**\$@

The “**Private ID**” is an identity for private purposes. For example, it can be “DVB-H” to indicate that this ID is designed for service over DVB-H network. Another example is that it can be “John” to indicate the author of the editor.

The “**Application serial No.**” is used to separate different applications when there are multiple applications in the ID string. It means that multiple applications along with their parameters will be placed in the numeric order in the ID string. Example:

```
xxxx@@cmd0n@@Application 1 title@@cmd0v@@Parameter  
value@$c0cmd$@@cmd1n@@Application 2 title@@cmd0v@@Parameter  
value@$c1cmd$@.....
```

The “**Application title**” is automatically generated by the proposed service creation tool during the “drag and drop” manner according to the name of the selected application in the application database.

The “**Parameter serial No.**” has a similar function with the “Application serial No.” that is to separate different parameters of an application. Multiple parameters are placed in the numeric order in the ID string. Example:

```
xxxx@@cmd0n@@Application 1 title@@cmd0v@@Parameter 1  
value@@cmd1v@@Parameter 2 value@@cmd2v@@Parameter 3  
value@$c0cmd$@@cmd1n@@Application 2 title@@cmd0v@@Parameter  
value@$c1cmd$@.....
```

The “Parameter value” is the parameter reference to the corresponding application

when the service platform on the terminal device is trying to run this application. The corresponding parameters of the supported interactive applications illustrated in Figure 3.12 are listed in Table 3.3. Besides an example of IDs in a XHTML page source code is illustrated in Figure 3.8.

Applications	ID Configuration	
	Application Title	Parameter value
Media Control	Playvideo	URI of the target video (e.g. "medias/clip1.mpg")
	Stopvideo	"null"
Live data feed	Realtimedata	URI of the target data source (e.g. "realtimedata.txt")
Live voting	Voteclient	voting target name __vote__ vote value (e.g. "medias/clip1.mpg __vote_good")
	Votereport	Order of request (e.g. "yes")

Table 3.3: Corresponding parameters of supported interactive applications

```

<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
  <title>testpage5.html</title>
</head>
<body>
  <p>
    
    
    
  </p>
  <p>
    
    
    
  </p>
</body>
</html>

```

Figure 3.8: Example of IDs for different interactive applications in an XHTML service page source code

From the definition of the ID, one of the advantages of such ID can be recognized that this type of ID supports multiple applications with multiple parameters. More precisely, the proposed assisting application model is able to allow multiple applications running within the lifecycle (press a component of the service – get reactions) of one interaction between the service and the user.

Therefore by using a XHTML and Java ME based “ID Event” method, the designers can successfully create the integrated MDTV service pages with the assistance of the

semi-automatic creation tool. All the MDTV service components are created and implemented as shown in Table 3.4, where the CSS will be involved also incorporated in future work.

MDTV service components		Presentation method	
MDTV content	Text	XHTML + CSS	
	Image	XHTML + CSS	
	Audio and Video	XHTML + MIDP MMAPI	
	Local interaction	MIDP	
MDTV interactive applications	Remote interaction	Hyperlink event	XHTML
		ID Event	ID Event method

Table 3.4: XHTML and Java ME based MDTV service presentation

3.3.1.2 Semi-automatic service creation software architecture

The proposed semi-automatic MDTV service creation tool is developed with Java 6 Standard Edition for the main tool’s GUI and functionality, and Java Micro Edition for the embedded MDTV service application database. The GUI is developed with Java Swing Component API. Java Swing is the lightweight UI developing toolkit in Java SE that its look-and-feel is independent from OS, which is different from Java SWT UI toolkit that the look-and-feel is different on different OS. Having considered that the creation tool may be used in different OS environment such as Windows, MAC OS and Linux, we chose Java Swing for GUI development to increase the portability of the software.

A) Software class components

As illustrated in the Class Diagram in Figure 3.9, the proposed semi-automatic service creation tool consists of eight classes: “MainFrame” (code contains 422 lines) is the main class; “LeftPane” (code contains 353 lines), “RightPane” (code contains 241 lines), “SouthPane” (code contains 70 lines) and “CenterPane” (code contains 446 lines) are its association classes; “IDHandler” (code contains 181 lines) and “CloseIcon” (code contains 80 lines) are the aggregation classes of “CenterPane”;

“TreeTableModel2” (code contains 200 lines) is the aggregation class of “SouthPane”.
 The detailed function description of all the classes refers to the corresponding software testing section in Chapter 5:



Figure 3.9: UML Class Diagram of proposed service creation tool

B) Software functional components

All the functional components in the service creation tool are listed in Table 3.5:

Functional Components	
GUI	Menu
	Hotkey Bar
	Project Explorer
	Workbench
	MDTV application list
	Parameter table
ID handler	
MDTV service application database	

Table 3.5: Functional components

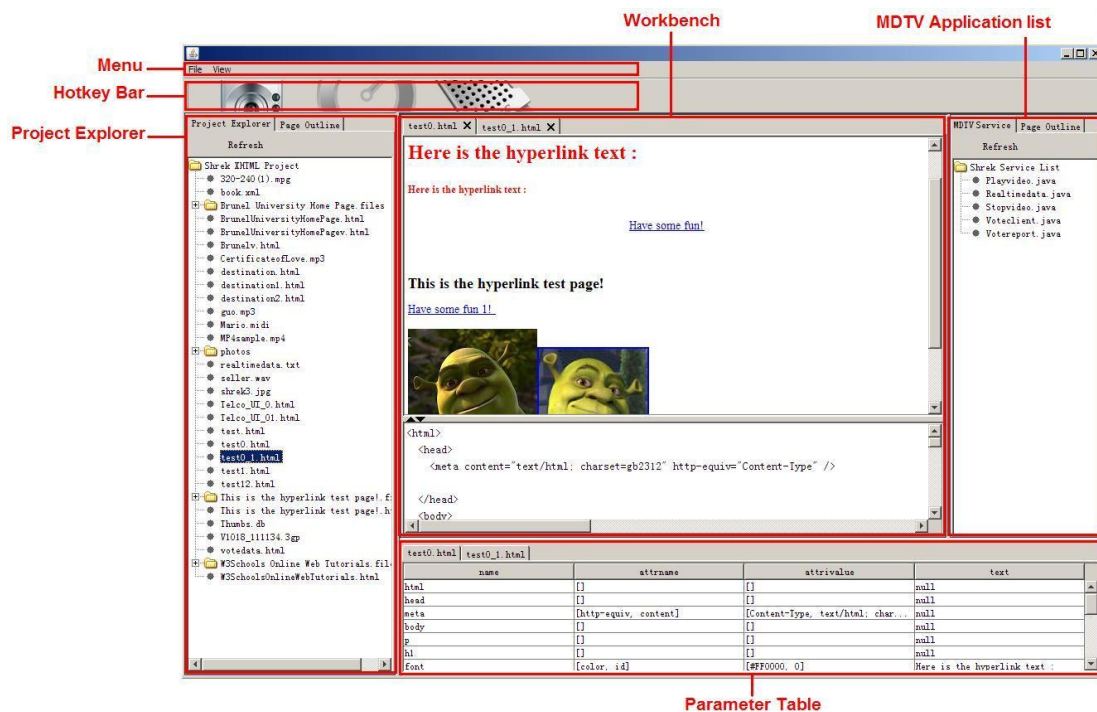


Figure 3.10: Semi-automatic service creation tool GUI overview

The Menu is the typical GUI functional component in the software design. The Menu currently contains two subsets: the File subset and the View subset. The File subset contains basic functions such as open file/project, save file/project and exit. The View subset contains the method of setting the layout style of the GUI. Other functional subset will be added into the menu along with new functions as further work. **Hotkey bar** is designed to quick-access to the frequent-used functions such as open/save file.

The Project Explorer is a viewpoint project directory. It lists all the XHTML

formatted MDTV service content and includes an editor that can select and open items from the list by double clicking on them. The Workbench is the main window for editing the service page. Once a service page is selected from the project explorer, the page will be opened and displayed in the workbench in a “split mode” as the default editing mode: page preview is displayed in the upper panel and the corresponding source code is shown in the lower panel. The lower source code panel supports plain text edit as well as drag-and-drop edit in order to add a MDTV URI or the interactive applications based on the “ID Event” method. The workbench also supports multi-editing mode where the service editor can use the tab bar to swap between different pages to achieve concurrent editing.



Figure 3.11: Workbench and MDTV application list

The MDTV application list provides a list of all the available MDTV interactive applications. It links with the application database embedded at the back-end of the service creation tool. When there is a need of adding an interactive application to a component in an XHTML service page, the editor can perform this by simply dragging the required application from the list and dropping it onto the selected component of the page in the workbench. In the meanwhile the rest of relevant configuration work (including adding the corresponding application ID attribute to the source code of the target element, generating relative metadata for other MDTV service like ESG and generating relative functional class for the service platform on

the terminal device) will be done automatically.

The Parameter table displays all the components in the XHTML page under editing, in the form of element name, attribute name, attribute value, and element content. This parameter table also supports multi-display mode and the sub table gets focused and closed together with the corresponding page in the workbench. The contents in the table also change along with the XHTML service page. The parameter table currently acts as a reference when modifying the components or adding any event ID. Further work will optimize the display preference of the table and enable the direct edition to the parameters in the table to modify the service page.

The ID Handler stays at the back-end of the service creation tool providing the link between the Parameter table and the Workbench while adding an interactive application based on the “ID Event” model. It helps the embedding of the ID’s corresponding application program into the back-end of the MDTV terminal device service platform, in order to support the ID event when rendering the component that has that ID in the XHTML service page.

SUPPORTED INTERACTIVE APPLICATIONS		
Applications	Functions	Descriptions
Media Control	Play media	Start playing the media streamed from a remote broadcast server or local directory.
	Stop media	Stop playing the media.
Live data feed	Require data	Provides live data feed from a remote server and displays it on screen.
Live voting	Vote action	Allows the user to interact with a customized voting application that could be related to the main TV content through a server.
	Require result	Require overall vote result to a specified service component from remote server.

Figure 3.12: Supported interactive applications

The MDTV service application database is embedded in the service creation tool. It contains all the available interactive applications especially developed for the MDTV service based on MIDP and they will be picked up according to the ID and added to

the terminal service platform. This database can be enriched by adding new MDTV applications with regards to future service requirements.

Interactive service applications that are currently supported within the proposed solution of MDTV service creation and implementation (including service creation tool and service platform for the terminal device) are listed in Figure 3.12.

3.3.2 SEMI-AUTOMATIC MDTV SERVICE CREATION PROCESS

3.3.2.1 Data flow through the semi-automatic service creation tool

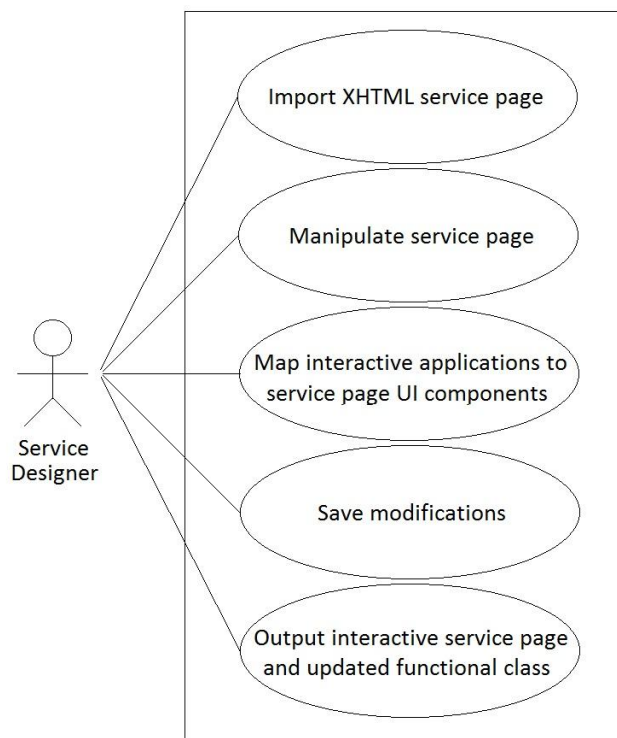


Figure 3.13: UML User Case Diagram for proposed service creation tool

In the UML User Case Diagram (see Figure 3.13), the proposed MDTV service creation tool can be broken down into five user cases: Import XHTML service page; manipulate the service page for normal design purpose; map interactive applications to service page UI components in order to create MDTV interactive service; save all the modifications done to the service page; finally output the integrated MDTV service page as well as the updated functional class for supporting new added applications on the client-side service platform.

After the MDTV content has been integrated in the XHTML format, all the service pages will be firstly imported to the service creation tool for further design (see Figure 3.14). The service designer can select those service pages through the Menu and Project Explorer components in the tool and double-click to open them. Both the page preview and the source code will then be displayed in the Workbench. The designer can carry out a series of design work such as general UI editing.

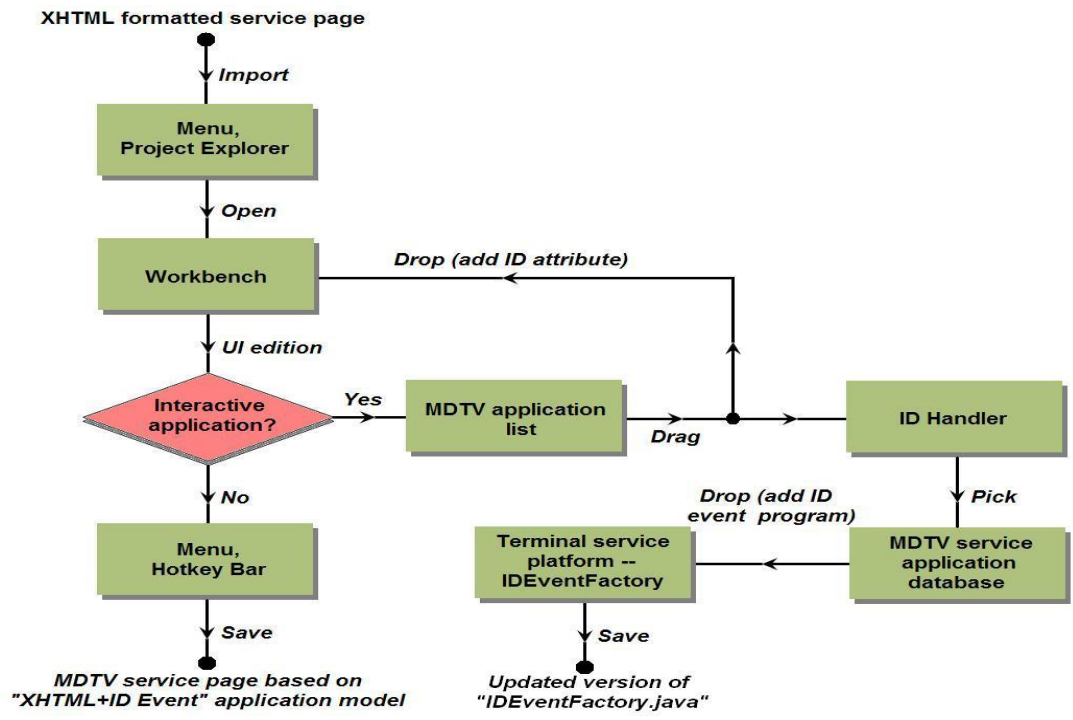


Figure 3.14: UML Data Flow Diagram: Data flow through the semi-automatic creation tool

If there is not any additional interactive application that need to be added in the service page, the designer can use the “save” function in the Menu or Hotkey bar to save the service page and finish editing. If some additional applications need to be added in the page, the designer can swap the Workbench to the “element mode” to start the “ID Event” edit process.

The designer first selects the required application from the MDTV application list, and then drags it towards to the target component displayed in the Workbench. As soon as the designer drops the selected application, the software will require some relevant parameter configuration according to the application needs (e.g. video

resolution, filename, etc). After setting the parameters, two actions are performed automatically: namely a special ID (for its definition refer to Section 3.3.1.1) is added into the target component as a new attribute (ID e.g.: shrek@@cmd0n@@Playvideo@@cmd0v@@root1/320-240(1).mpg@\$c0cmd\$@. The instruction refers to Figure 3.16). The workbench updates its scene to display the modification to the service page; The ID Handler is triggered and picks up the corresponding application program from the MDTV service application database, according to the name of the dragged application. Then the ID Handler adds this application program into the “IDEventFactory.java” in the terminal device service platform to generate a new version of “IDEventFactory.java” as the corresponding ID Event handler that supports the new added interactive applications.

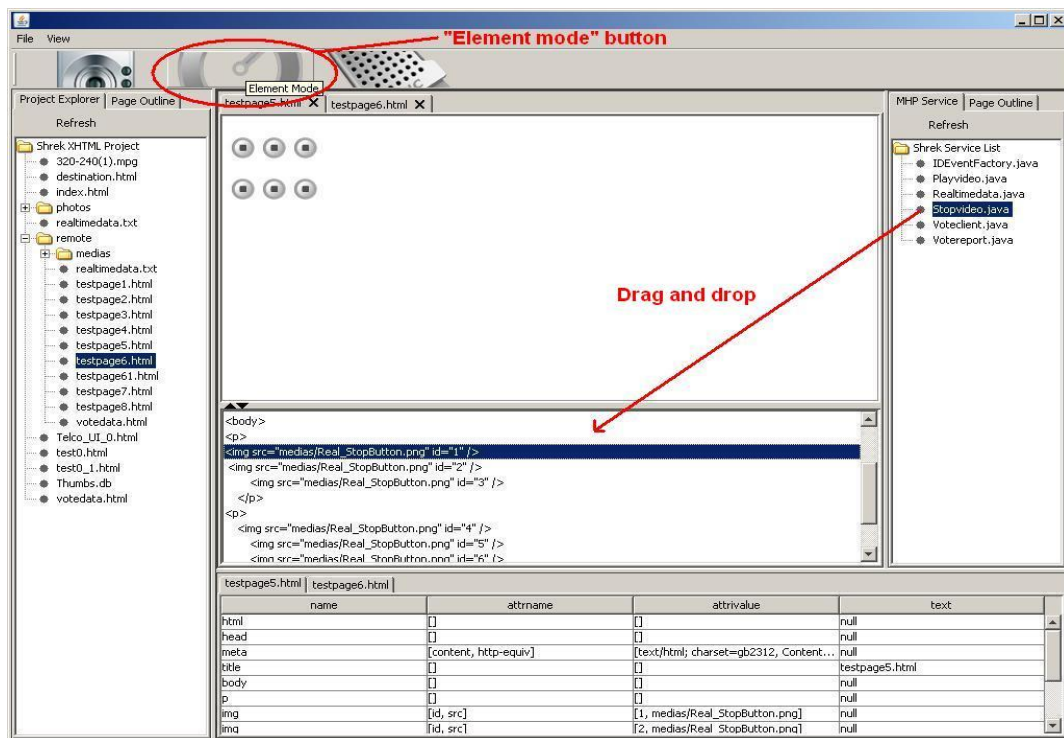


Figure 3.15: Sample XHTML service page opened in the proposed creation tool (before the addition of the ID event)

xxxxx@||@cmd0n@||@Playvideo@||@cmd0v@||@root1/320-240(1).mpg||@\$c0cmd\$@

Private ID **Application start tag and serial No.** **Application title** **Parameter start tag and parameter serial No.** **Parameter Value** **Application end tag**

Figure 3.16: Instruction of the ID example

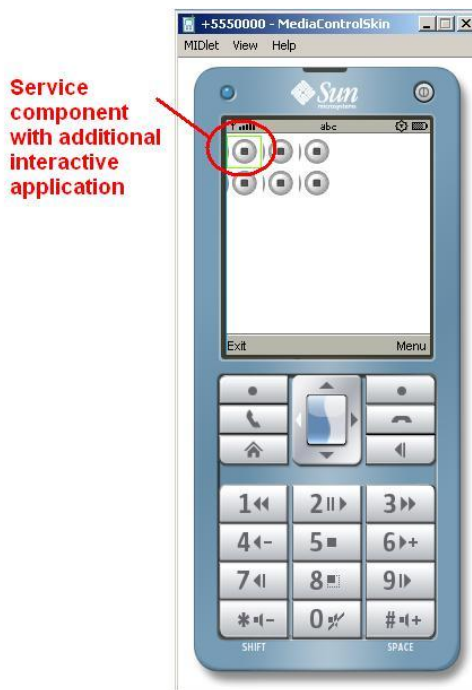


Figure 3.17: The screenshot of the service page outputted from the proposed creation tool

Finally, by saving the project, the integrated MDTV service page is outputted ready for further process such as service content packaging. An updated version of “IDEventFactory.java” is outputted as well.

3.3.2.2 Comparison and discussion

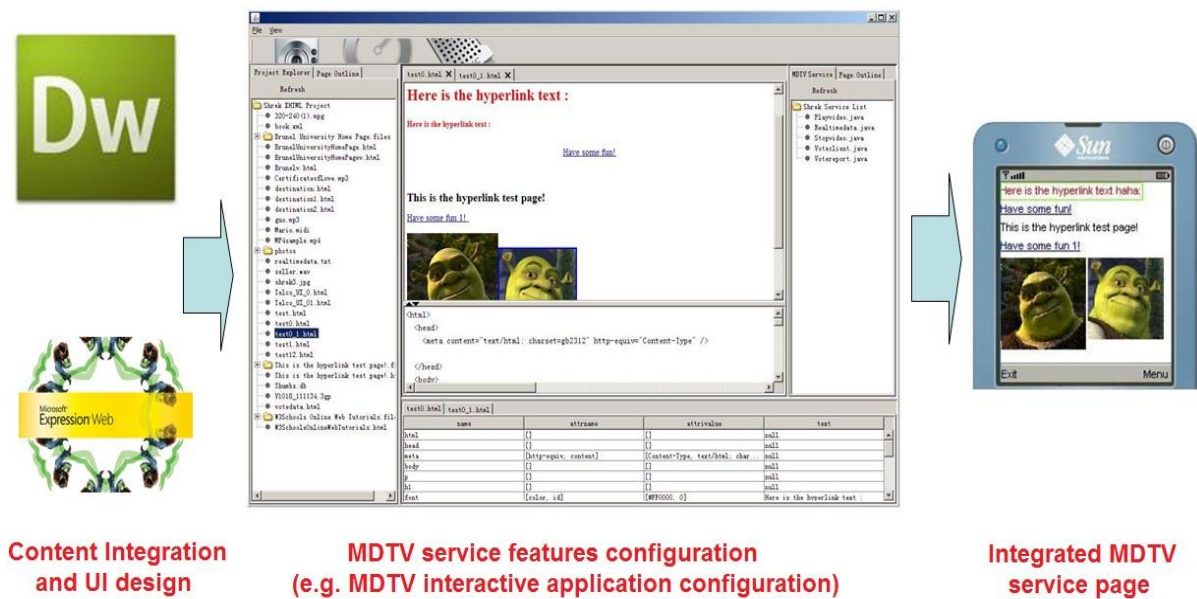


Figure 3.18: implementation of semi-automatic service creation

As discussed in previous sections of this chapter, the proposed semi-automatic service

creation mechanism has managed to cope with the most of the challenges of the Thesis. Moreover when compared to the current commercial solutions and some related research proposals, the proposed semi-automatic service creation mechanism also shows advantages.

As mentioned in Chapter 2, many of the commercial MDTV service creation solutions (such as ACCESS NetFront, EXPWAY and Streamezzo) have got a heavily proprietary character throughout their solutions. In contrast, the Thesis's proposed service creation process manages to avoid those proprietary issues by utilizing open-standard technologies including XHTML and Java ME. Just like Apple Co. opens their iPhone OS SDK to the public, the proposed service creation process manages to demystify the MDTV service implementation process and open the service creation to the public. This will encourage more creative designers to get involved and further encourages the development of new applications and services for MDTV in the future.

In the academic field, some of the related research works also realize the same problems and have proposed several solutions. However, most of them are based on Rich Media technologies [104] [119]. The proposed solution in [104] chooses SVG Tiny for visual presentation and Thinlet XUL in cooperation with Java for interaction presentation. As discussed in previous sections, Rich Media technologies require advanced design skills and do not support different screen sizes well. Moreover the interaction presentation solution is relatively complicated since the solution tends to use XUL to create the link between SVG Tiny and Java for implementing the corresponding interactions. When this solution is compared with the Thesis's proposal, the proposed solution shows evident advantages on both of these two aspects. XHTML is much more adaptive than SVG Tiny when producing the visual presentation; the ID Event method that has a simple definition is much easier to adopt than a script language like the Thinlet XUL. The proposed solution in [119] tends to provide a convertor for MDTV service creation based on different Rich Media technologies such as MPEG LAsER and MPEG BIFS. Comparing it with our solution,

leads to the conclusion that this solution is also relatively weak when trying to resolving the incompatibility issue of Rich Media. Since the utilization of XHTML can be more efficient than adding a convertor in the Rich Media based service creation process that is already complicated.

As a conclusion, the proposed semi-automatic service creation mechanism has realised the following contributions:

It facilitates the MDTV service creation process by introducing a semi-automatic service creation tool in the process to assist the designers producing the interactive-rich MDTV service. The technical demand to the editors is reduced, so that editors do not have to become technical masters on the MDTV professional technologies or the Java ME software development skills. Thus the editors can pay more attention to the design works based on the XHTML visual presentation and more design oriented professionals in IT field can be involved into the MDTV service creation more easily.

The selection of XHTML as the service visual presentation technology not only brings more design-related resources, but also essentially ensures the inter-compatibility of XHTML based MDTV service among different standards. The ID Event method for implementing the interactive applications on the service page is much easier to learn and use than other scripting languages such as the ECMAScript. This further simplifies the service creation process and also manages to cooperate with the service platform in the terminal device (will be discussed in next chapter) that is also based on Java ME. Lastly, the utilization of XHTML and Java ME provides an open-source and open-standard service development environment and does not suffer from the proprietary issues that other solutions face, during the MDTV service creation process.

Summary:

This chapter further states the motivation and the methodology of the proposed semi-automatic MDTV service creation process. The semi-automatic service creation tool

along with the XHTML and Java ME based service presentation method has been introduced in detail. Moreover a relative comparison and discussion has been conducted through the chapter and lastly reached the conclusion that the proposed semi-automatic MDTV service creation process has met the Thesis's aim and also shows advantages when comparing it with other current and related solutions.

4. CHAPTER 4: PROPOSED MDTV CLIENT IMPLEMENTATION ENVIRONMENT BASED ON JAVA ME

4.1 MOTIVATION

In Chapter 2, Figure 2.21 we have illustrate the typical MDTV service implementation model, where the terminal device service platform and the MDTV service server form the service consumption environment. Thus as described in Section 2.7.3, when the proposed solution for MDTV service creation and implementation is sent to the client end, the MDTV client implementation environment (MDTV-CIE) that consists of the proposed terminal device service platform (Client side) and the prototype server (Server side) enables the retrieval of the MDTV service on the terminal device and the consumption of the MDTV service through the one-way broadcast provision and the handling of interactivity between the terminal device and the server. This is illustrated in Figure 4.1. In this environment, the terminal device service platform is proposed as another core component in the proposed solution apart from the semi-automatic service creation process and tool presented in Chapter 3. The server is currently developed on the prototype level only for testing purposed and will be further enhanced during future work.

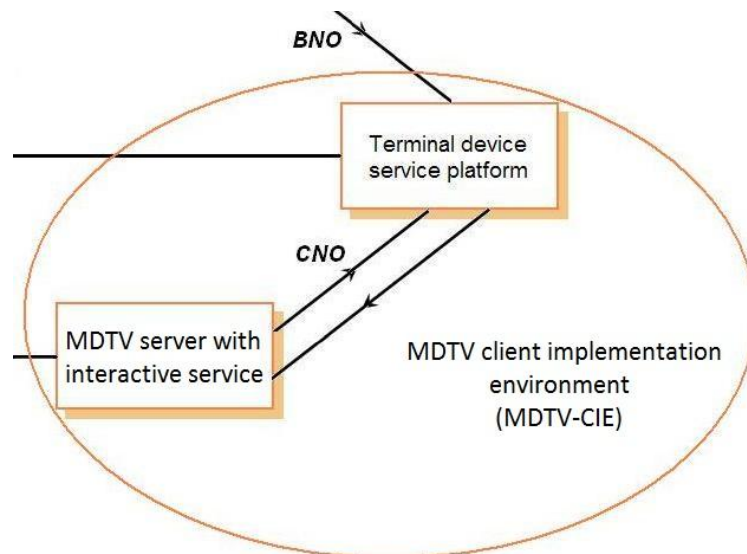


Figure 4.1: MDTV client implementation environment (MDTV-CIE)

The main motivation for developing the terminal device service platform is to assist the implementation of the proposed MDTV service solution. In the proposed Semi-automatic MDTV service creation process, the XHTML and Java ME have been chosen as the presentation technologies and the service is then created in the form of XHTML formatted pages by the designer using the semi-automatic creation tool. Thus when these service content and applications are received in the form of service streams and in turn decoded in the MDTV client side device (e.g. mobile phone), there is a requirement of a software “platform” on the client-side that is able to retrieve those service contents (audio, video, text, image and etc.) and handle any interaction between the service and the user during the consumption process.

4.2 MDTV TERMINAL SERVICE PLATFORM DISCUSSION

Research work on the current MDTV service platform solutions were then carried out after realizing the aforementioned requirement. Through this research the methodology for developing a service platform for the Thesis’s proposed MDTV service implementation solution has been realised: 1) select a suitable service platform structure type; 2) based on the selected structure, actualize the service platform either by modifying the existing software with the similar function and structure, or by developing a customized service platform according to the proposal’s requirement.

4.2.1 SELECT THE BROWSER-BASED SERVICE PLATFORM STRUCTURE

As discussed in the previous chapter, two types of MDTV service platforms are being researched and adopted in practice: the generic browser based and the rich media based, both of which have their pros and cons. The motivation here, in choosing the browser-based platform structure, mainly emanates from the facts as below:

On one hand a browser-based structure is a more common form of a human-computer interface (as it has been successfully applied in the Internet navigation paradigm). A

typical browser-based structure software achieves its functionality by rendering the received service contents and layout, retrieving those content audio-visually through the I/O unit of the hardware, and handling the interactions between the service and the consumer during the service consumption. It is more generic, and its supported contents are usually based on the group of sophisticated open-standard and open-source technologies (such as HTML, CSS and etc.) that have been already widely adopted in the IT field. It is also capable of supporting the newest technology (such as SVG Tiny, SMIL and other rich media components) through extending it. On the other hand, the service content based on this platform is independent from the platform (similar to the relationship between web services and a web browser), which means the creation and the maintenance of the service content can be individually and separately implemented. This is one of the key differences when compared with the rich media based service platform, where content has to be developed specifically and is often tied to these platforms.

A typical rich media based structure usually has a Rich Media parsing engine as the core unit with only a few of assistant technologies such as XML DOM and ECMAScript. Unlike, even a browser based platform with rich media enhancement, a rich media based service platform is more similar to a simple rich media client where there is only support for rendering the content with pre-specified rich media features. In other words, there are usually no extra features or technologies being supported within such rich media client and the extensibility of the rich media structure is relatively limited than a browser-based structure for further modification and update. This in turn restricts the service platform from being compatible to the content based on other presentation technologies and moreover the whole MDTV service implementation system (including service creation, server and rich media client) has to be proprietary.

Thus, a browser based platform solution has better flexibility. Its underlying supporting technologies are usually much more than a rich media client, which ensure

its compatibility to a wider range of service content formats; its extensible framework enable itself to be relatively easier and more open when it comes to the service creation and maintenance.

Regarding the Thesis's aims, if a browser-based structure is applied for the terminal service platform development, the proposed semi-automatic service creation process can be supported better:

1) A browser-based service platform is able to support many of the open standard technologies such as XHTML more easily. The XHTML based MDTV service contents outputted from the proposed semi-automatic service creation process can thus be recognized just as the conventional XHTML pages on the first stage before further retrieval and processing, without any additional development to the platform. Whilst some extra adjustment needs to be done first to a rich media based service platform before it is able to render the XHTML based service. This is because there is usually no XHTML parsing engine within a rich media platform structure. Moreover the extensibility of a browser-based service platform is better than the rich media structure does since the browser-based structure is relatively generic and open for any additional features, whilst the rich media structure is too specified for the Rich Media technologies to accept any reconstruction.

2) By using a browser-based service platform, the complexity during the MDTV service creation can be reduced. As discussed in Chapter 3 when comparing the XHTML and Rich Media, with the help of a browser-based MDTV terminal service platform, the XHTML based service content is more adaptive to the different terminal device screen sizes and resolutions than Rich Media is. The XHTML based service contents can thus be created regardless of the layout differences through various screen sizes.

Our proposed MDTV terminal device service platform therefore employs the browser-based structure for its basis and with a series of developments especially

incorporating MDTV service features (e.g. support to the proposed “ID Event” scripting), this proposal aims at producing an enhanced browser-based MDTV service platform.

4.2.2 SERVICE PLATFORM DEVELOPMENT AND DISCUSSION

Having selected the suitable structure, the actualization of the service platform is then the second step of the work. Basically, there are two main ways of implementing the service platform under the browser-based structure: firstly, by modifying existing commercial mobile web browsers to be proposal specific; secondly, by developing a customized XHTML browser-based service platform.

As mentioned in the Chapter 2, several commercial MDTV implementation solutions have been applied the first approach by modifying their proprietary mobile web browser solutions and extending the browser with the relevant MDTV features. ACCESS NetFront Browser DTV Profile is a good example where the company combines its existing mobile web browser solution NetFront Browser with EXPWAY FastESG and supports several MDTV transport standards such as DVB-IPDC and OMA-BCAST, so that the NetFront Browser is able to become a MDTV terminal service platform [90]. However in our case and research, this approach becomes very difficult: since on one hand, we would have to gain access and permission to proprietary commercial browser solutions, which is mainly available for commercial research projects. On the other hand, we would have to redesign the main runtime engine of those commercial browsers and structure to a great extent in order to render them more efficient for the proposal’s MDTV service specific requirements based on XHTML and “ID Event” scripting. Thus this approach is more suitable for a commercial development held by the browser’s authoring entity.

Having not selected the first approach as the solution, the actualization work moves onto the second approach that of developing a new service platform for supporting the proposed semi-automatic service creation process. The design requirements to such

service platform should be clarified as follow:

- The platform should be able to render the XHTML based MDTV service content outputted from the Thesis's proposed creation process;
- The platform should be able to render the MDTV interactive applications based on the "ID Event" method and also be able to handle the interactions between the user and any interactive applications (including XHTML hyperlink applications and "ID Event" based applications)

Firstly, a software development technology should be chosen. Among various popular software development technologies such as C, C++, C# and Java, the Java Micro Edition is the most suitable selection according to the aim of Thesis's proposal as well as the relative discussions in Chapter 2 and 3. Moreover the MIDP, the core technology of Java ME, has been adopted as the basis technology of the proposed "ID Event" interaction presentation method. It has thus formed another key component in our proposed system along with the other relevant international standards of the XHTML and JSR272. Its native functionalities allow not only the involvement of XML, XHTML, SVG and other popular elements, but also the design of any complex functional mechanism such as an interactive mechanism for a MDTV service. Its extensibility by plugging additional functional JSRs or customizing packages in the existing system can enhance our proposed system with the newest features and technologies.

4.2.2.1 XHTML based MDTV service rendering discussion

The first requirement to the service platform development is that the platform should be able to render XHTML service pages. An XHTML rendering mechanism should then be developed to achieve this functionality. Based on the relative research work, we learnt that the XHTML parsing engine (XHTML parser) is the core unit within the rendering mechanism that is in charge of parsing the XHTML source code into various MDTV service components such as text, images, video and interactive applications. To develop a XHTML parser by using Java ME should therefore be the

next step.

In fact, the XHTML was not the first choice as the visual presentation technology in the proposed work at the beginning but HTML. Since XML is mostly employed for data storing and transporting, we decided to implement HTML at the early stage of our project for our service visual presentation solution due to its popularity. Therefore the following work was to parse the HTML contents and the choice then was either to program an HTML parser from scratch or to find an open-source parser from a relative field/project and modify it according to our needs. However the first approach required considerable effort on just the basic function of the service browser trying to reinvent the wheel. Moreover based on the research on HTML parser in Java ME, we concluded that there is no HTML parser in native Java ME API like it exists in Java SE Swing. There are a few third-party HTML parsers out but some of them are proprietary and most of the other open-source ones are not very reliable and not compatible with different version of Java ME, thus are not suitable for customization as our project requires. Finally, having realized that there are native and open-source third-party XML parsers available for Java ME, it was considered to develop a HTML parser based on a XML parser. But another fact that needs considering is that there is a gap between XML and HTML because the syntax of HTML is not as strict as XML and a XML parser may have difficulty to read the HTML code in a loose syntax. Thus a bridge ought to be found between XML and HTML under this circumstance. Eventually, a new version of HTML, the XHTML was found with excellent presentation ability as HTML, as well as a strict and clean syntax as XML. By aggregating both advantages of XML and HTML, the XHTML became our choice.

Then the following work was led to develop the XHTML parser by extending the XML parser and a methodology was then proposed. As illustrated in Figure 4.2, the XHTML stream firstly arrives at the XML parser. Since XHTML has the same format as XML, the XML parser is then able to process the stream into XHTML code in string format. The XHTML code is then forwarded into a XHTML parser. A “string

marker” subsystem in the XHTML parser captures the code and marks the elements and their attributes to be recognizable to the “element sorter” subsystem. The “element sorter” then traverses the marked XHTML code, pulls out all the elements and the attributes and eventually saves them for further parsing operations.

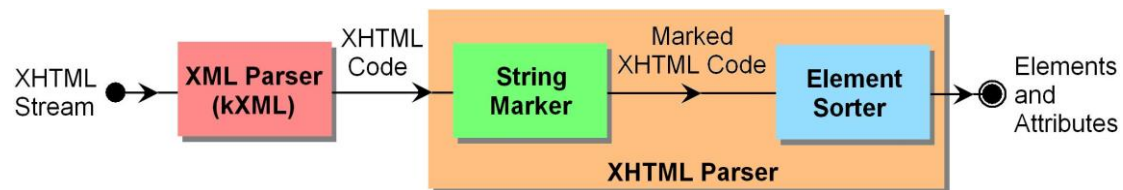


Figure 4.2: The integration mechanism between XML parser and XHTML parser

With this methodology, we firstly tried the native XML parser in Java ME, the SAX Parser as the basis but the negative result led us to a giving-up on it, since on several instances the tags and their attributes in the XHTML code couldn’t be matched properly when the SAX Parser was working on them. Eventually after a series of experiments, we implemented the kXML parser as the basis of the XHTML parser. The kXML parser is a small pull parser, specially designed for constrained environments such as Applets, Personal Java or MIDP devices. The kXML parser is licensed under the BSD license [126].

There are three fundamental XML parser types: the model, push and pull [125].

- A model parser reads an entire document and creates a representation of the document in memory. Model parsers use significantly more memory intensive than other types of parsers;
- A push parser reads through an entire document. As it encounters various parts of the document, it notifies a listener object. (For example the SAX Parser);
- A pull parser reads a little bit of a document at once. The application drives the parser through the document by repeatedly requesting the next piece. (For example the kXML parser).

In contrast to push parsers like the SAX Parser, pull parsers like the kXML parser make it possible to model the XML processing routines after the structure of the

processed XML document. The events processing in kXML are similar to an InputStream in Java ME. If a part of the stream requires special handling, the kXML parser can simply be delegated to a specialized method by handing over the parser [126]. Besides, a pull parser like kXML requires less memory and process resources for a parsing work, which is suitable basis for a XHTML browser on a mobile device such as a MDTV featured mobile phone.

Based on the kXML parser, the XHTML parser is eventually built up by developing a mechanism that classifies and organizes the different element tags and attributes defined in the XHTML specification. A slight modification is done to the original version of kXML source code due to the cache limitation in each pull parsing action. With the XHTML parser, the XHTML rendering mechanism was further developed, of which the mechanism structure will be stated in the system instruction section.

4.2.2.2 Interactive application rendering and handling

As to the second requirement, the service platform should be able to render and handle MDTV interactive applications. As presented in Chapter 3, the proposed semi-automatic MDTV service creation process has defined the XHTML and “ID Event” method as the service presentation technologies and Table 3.4 further illustrates the details of the various service components presentation solutions. Regarding to this definition, the interactive applications can be divided into two types: the Hyperlink Event and the “ID Event”.

Since all the ordinary MDTV service components including audio/video, text, images, and the Hyperlink Event are formatted in the form of XHTML code and are finally received, parsed and rendered as the XHTML based service pages. The interactive event at this stage which is the Hyperlink Event is also handled as a normal XHTML element. For the rest of the service components, which is the “ID Event” according to the context, it can be pre-embedded to the XHTML based service page by adding the corresponding ID attribute to the target element using the proposed service creation

tool. Therefore during the service page being parsed in the service platform, the ID Events are also pulled out and registered to their target XHTML components. When any ID Event is triggered, it is handled by a special functional manager (will be discussed in the next section) in the service platform and the client-server interactivity implementation mechanism that consists of the service platform and the server is also involved. All the implementation process details will be presented along in the data flow in the system instruction section.

4.3 MDTV SERVER IN THE MDTV-CIE

A client-server mechanism is a sub-system in a MDTV-SIM model for the implementation of the MDTV service. The MDTV server supports the application model defined in its underlying middleware/software standards and is responsible for monitoring events from the client, catching event requests and processing the requests and responding back to the client as requested. The signalling and data exchange between the client and the server is what enable the interactive MDTV services. As mentioned in the previous chapter, this mechanism is commonly used in MDTV standards and other relevant commercial service implementations and we would also employ it as the MDTV interactive service implementation structure in our proposed system. More precisely, the MDTV-CIE forms the client-server mechanism for handling the interactive applications in the proposed system architecture (illustrated in Figure 2.23).

The MDTV server (server side of MDTV-CIE) is a part of the proposed solution architecture (refer to Figure 3.6 in section 3.31 Chapter 3) but its full implementation is current out of the scope of the Thesis's objective. Thus the implementation of the MDTV server in this chapter is only for complementary and testing purposes. The server will be developed with necessary functions only to assist the service platform to implement the proposed interactivity implementation mechanism in the MDTV-CIE. More precisely, the MDTV server will be implemented only as the **Interactive**

Service Managing Sever in the current circumstance.

4.4 MDTV-CIE SYSTEM ARCHITECTURE

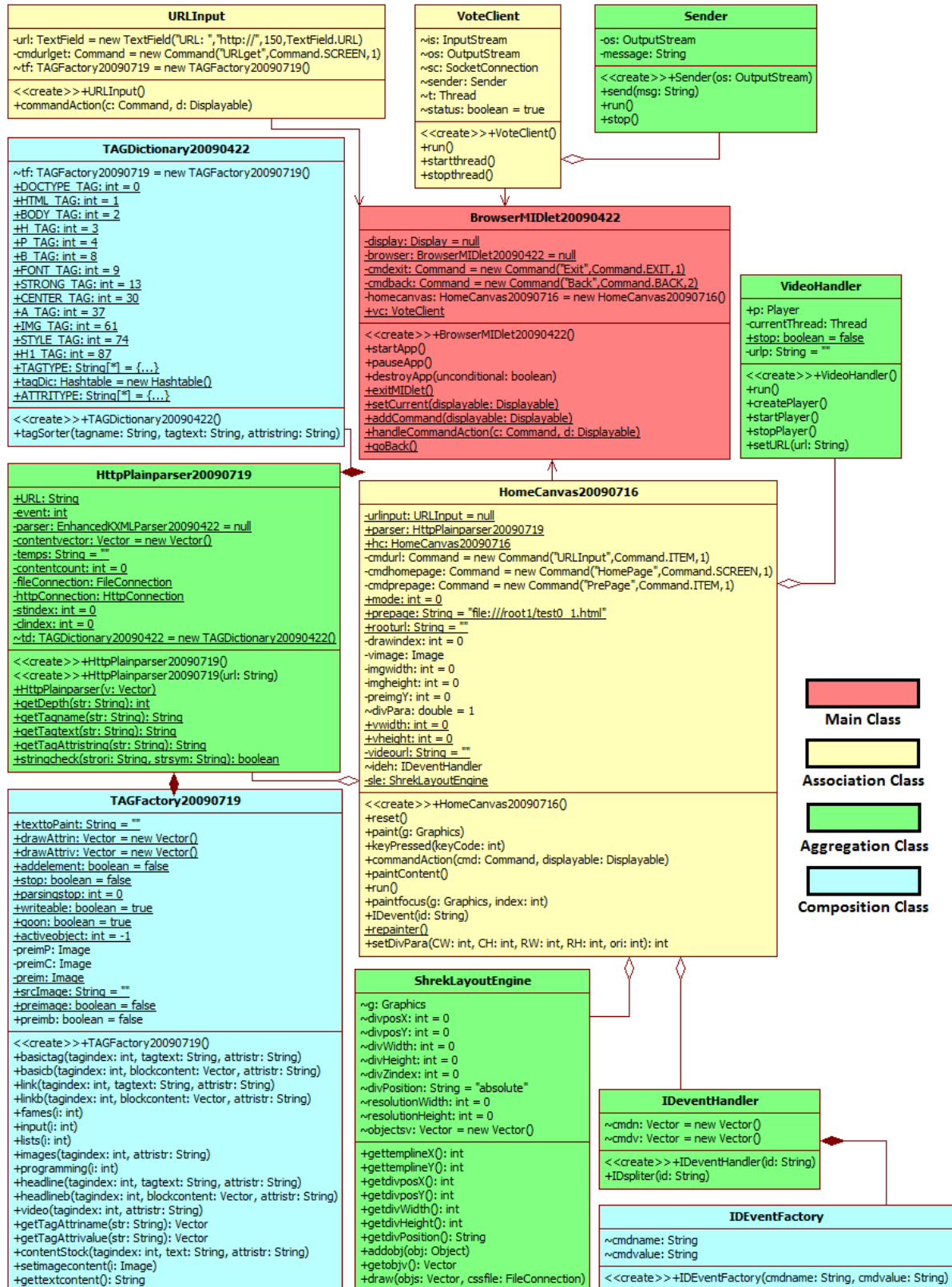


Figure 4.3: UML Class Diagram of the terminal device service platform

The MDTV-CIE system is constructed based on the terminal device service platform and the interactive service managing server. The terminal service platform consists of several functional classes as shown in Figure 4.3. The interactive service managing sever is developed in Java SE and its Class Diagram is illustrated in Figure 4.4.

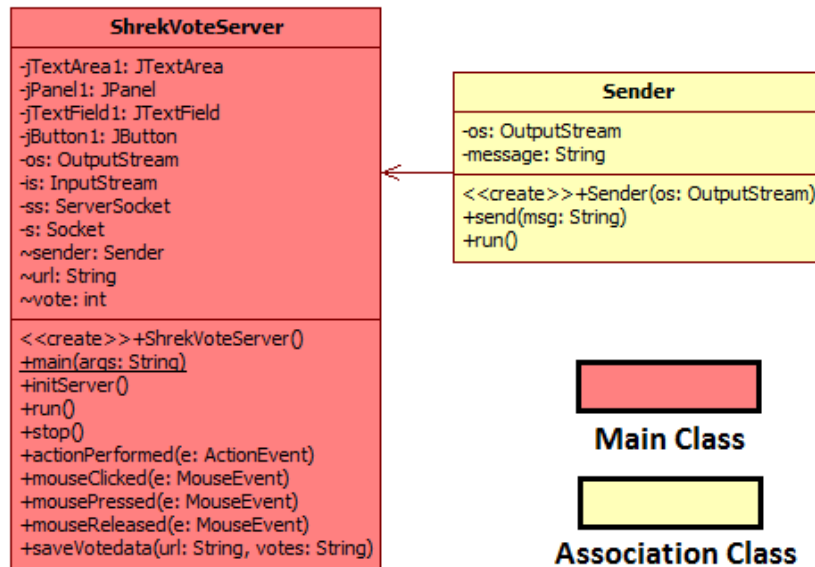


Figure 4.4: Class Diagram of the interactive service managing server

As illustrated in Figure 4.3, there are totally 12 classes in the client side service platform software system. The “BrowserMIDlet” is the main MIDlet class in charge of initializing all the components of the service platform; “HomeCanvas” is the main assistant class to the “BrowserMIDlet” for retrieving terminal platform’s UI and all of the visual elements in the service pages; the “URLInput” is a complementary UI page to the “HomeCanvas”; the “VoteClient” assists the “BrowserMIDlet” by making connection with interactive service managing server and enabling the data exchange between client and server; the “HomeCanvas” further has four aggregation classes, the “HttpPlainparser”, “LayoutEngine”, “VideoHandler” and “IDEventHandler”. The “VoteClient” has the “Sender” as the aggregation class. There are also three composition classes, the “TAGDictionary”, “TAGFactory” and “IDEventFactory”, assisting their corresponding supervising classes. On the server side, there are two classes in the interactive service managing server: namely the “VoteServer” and

“Sender” (see Figure 4.4). The “VoteServer” is the main class that provides server side access for interactive applications and data exchange in the client-server architecture. The “Sender” acts as the assistant class to the “VoteServer”. The code lengths of all the classes in the MDTV-CIE system are shown in Table 4.1.

Functional Class	Source code length (lines)
BrowserMIDlet	83
HomeCanvas	501
URLInput	42
VoteClient	108
Sender	48
HttpPlainparser	318
LayoutEngine	151
VideoHandler	122
IDEventHandler	52
IDEventFactory	160
TAGDictionary	387
TAGFactory	726
VoteServer	274
Sender	48
Total	3020

Table 4.1: Code length of all the classes in the MDTV-CIE

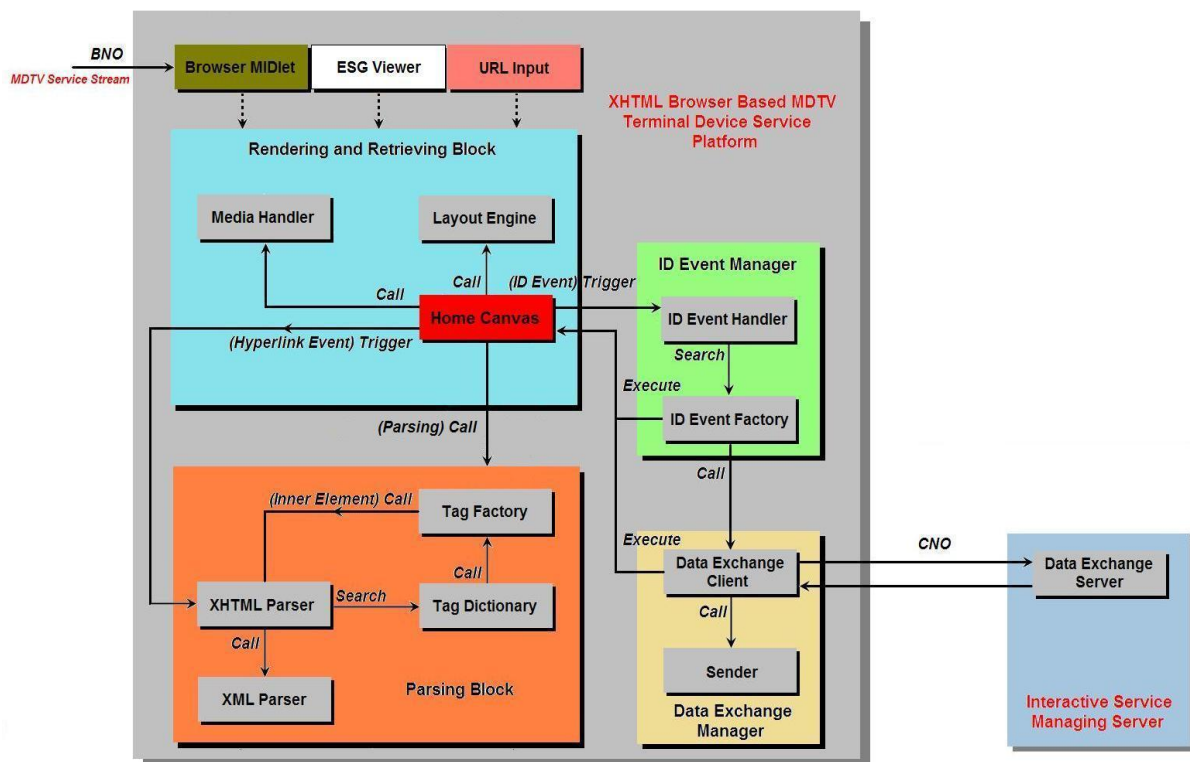


Figure 4.5: MDTV-CIE class functionality diagram

All the functionalities of the entire MDTV-CIE software system, as stated in its design

requirement, are achieved through these classes and the logic and relationship between them are depicted in Figure 4.5.

Command definition of the class functionality diagram in Figure 4.5 is as follow:

- **Call:** The command that requires assisting functions or following processing functions from other classes;
- **Search:** The command that searches in the reference class and classifies the temple results for further processing;
- **Trigger:** The command caused by interactive events that triggers a series of relevant processing threads;
- **Execute:** The command that achieves the requested processing and updates the service scene in cooperation with rendering and retrieving block.

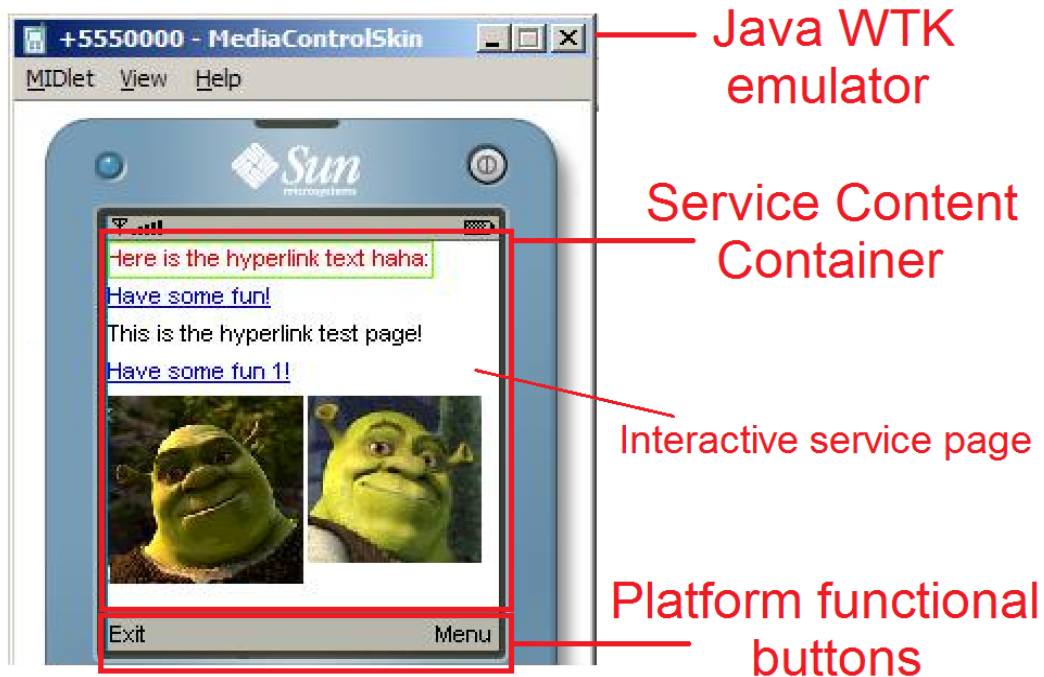
An arrow with solid line represents the direct logic flow and its position in a processing lifecycle whiles an arrow with a broken line represents the possible logic that could initialize the processing lifecycle at the first stage.

4.4.1 PROPOSED TERMINAL DEVICE SERVICE PLATFORM IN MDTV-CIE

As illustrated in Figure 4.6, the proposed service platform GUI has two functional components: the service content container and the platform functional buttons. The service content container includes all the multimedia components (such as text, image, video and interactive applications) on the MDTV service page, displays the components in a proper layout so as to be ready for further interactions. By selecting the relative platform functional buttons, a user can operate the terminal platform (such as navigating to the home page of the service or exiting the platform) in order to start/stop consuming a certain MDTV service. Thus with the assistant of these two GUI components, the user is able to consume MDTV services through the proposed terminal device service platform.

The software architecture of the service platform consists of six functional sections

(as illustrated in Figure 4.5): namely the Browser MIDlet, URL Input, Parsing Block, Rendering and Retrieving Block, ID Event Manager and Data Exchange Manager.



Service platform GUI components:

1. Service content container
2. Platform functional buttons

Figure 4.6: GUI of the proposed terminal device service platform

4.4.1.1 The Browser MIDlet



Figure 4.7: Three main portals of MDTV service in the MDTV-CIE

This is the only executable MIDlet of the service platform subsystem. Unlike DMB-

MATE that the service platform allows multiple application MIDlets running at the same time, each of which contains a certain group of MDTV contents including A/V and interactive auxiliary service data. Our proposed service platform allows only one MIDlet running in the JVM and all the MDTV services and interactive applications will be retrieved and consumed through this MIDlet. Having taken the browser-based feature, the required MDTV service will be browsed in a web-like form similar to that of Internet web services.

The Browser MIDlet defines the basic GUI including the basic navigating actions of the service platform, initializing the interactive mechanism for data exchange between the client and server, preparing the service platform to be ready for further MDTV service consumption and user interaction. By selecting the “Default service” or entering the Uniformed Resource Identifier (URI) of a certain MDTV service in URL Input class, the Browser MIDlet retrieves the required service content through either the broadcast network or the interactive network. Theoretically when the public version of JSR272 is released, the ESG Viewer, which can be selected in the Browser MIDlet menu, will be set up as the main portal for the MDTV service within our proposed MDTV service platform. Most of the services will be listed in the ESG for consumption. In short, the exiting gateways, “Default service” and the URL Input assisting class, and the future ESG Viewer are the main three portals of MDTV service in the MDTV-CIE.

4.4.1.2 The URL Input

This is an assisting class to the Browser MIDlet and is an optional portal of the MDTV service. As there is no ESG viewer currently, we designed this class as a temporary portal for the purposes of testing. Once entering the URI in the text field and pressing the “URLget” button in the menu, the service platform will send a request to the interactive server for a specific service.



Figure 4.8: URL Input class

4.4.1.3 The Rendering and Retrieving Block

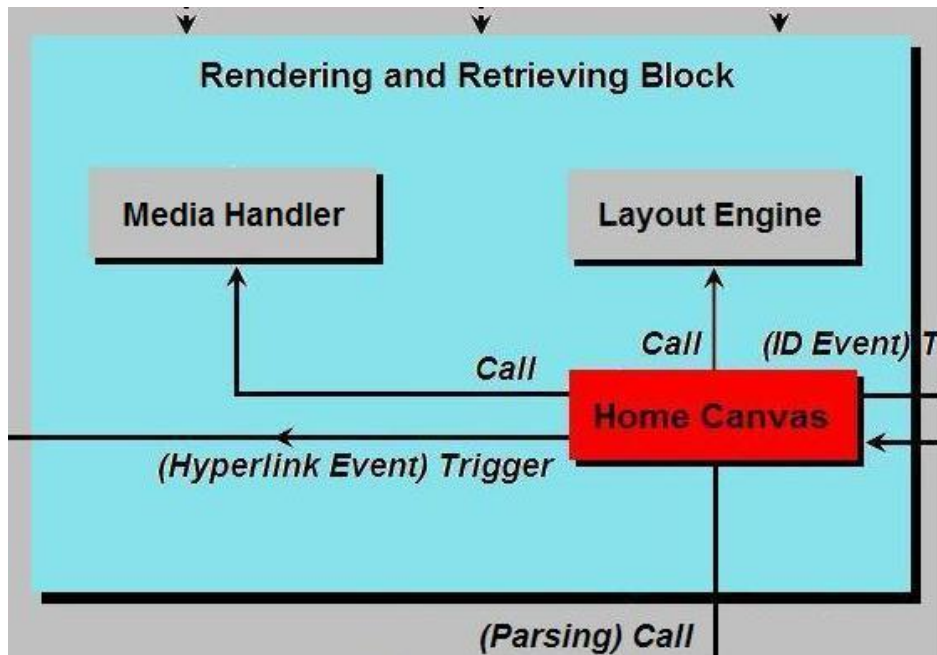


Figure 4.9: Rendering and Retrieving Block

The rendering and retrieving block acts as the processing command sender and the service retriever in the MDTV-CIE service platform. It receives the service access commands and relevant URIs from the three service portals, sends the service code, passing tasks to the Parsing Block. It also sends interactive service events and processing tasks to the Parsing Block and the ID Event Manager and receives the rendered results. There are three main classes in this block: the Home Canvas, Layout Engine and Media Handler. Among these, the Home Canvas is the leading class and

the other two are the assisting classes.

The Home Canvas

The Home Canvas is the core functional class within the service platform. It consists of the 2D Navigation Subsystem and Drawing Subsystem. The 2D Navigation Subsystem defines the default menu and layout of the service browsing canvas. It also defines the key functions, the navigation logic algorithm and incorporates an event detector that captures the service events, classifies and passes them to different event handlers according to their attributes. There are different types of service events which are the Hyperlink Event that belongs to the native XHTML code, and ID Event that is a pre-defined MDTV event for local and remote interactivity, such as voting or live data feeding. Due to their different attributes, the Hyperlink Event is passed to the Parsing Block for further processing and the ID Event is passed to the ID Event Manager. The Drawing Subsystem is in charge of receiving the service content and rendering it. It also sorts the content elements including their attributes and paints these elements on the service browsing canvas with different pre-defined methods; in cooperation with the 2D Navigation Subsystem, it repaints the active component as the navigating focus is moved. After the navigation subsystem has detected any service events, it passes them over to relevant classes for handling. The Drawing Subsystem can also receive the service event rendered results and update the scene in the canvas as requested.

The Layout Engine

The Layout Engine has been treated as a typical component in a general displaying mechanism in various kinds of software such as web-browsers and e-mail clients that are concerned with marked-up languages (XML, XHTML, etc.) coded content. Classic layout engines for web-browsers are: Trident for Microsoft Internet Explorer, Presto for Opera, WebKit for Apple Safari and Google Chrome, and Gecko for Mozilla [121] [122] [123] [124]. The main difference between these commercial

layout engines and the proposed one, is that commercial layout engines normally combine the parsing section and actual layout section together, whilst in our proposed solution, these are separated into two sections in terms of their functionality, namely the Layout Engine and the Parsing Block. The main reason for this separated form is that both of layout function and the parsing function still need further development and improvement and it is relatively easier to do modifications to each function when they are separated. In future work, these two will be merged together to form a functional block or class if there is any technical or update requirement.

The Layout Engine, Media Handler, Home Canvas, and Parsing Block are the functional classes for parsing and displaying XHTML based MDTV services in the MDTV-CIE, whereas the Layout Engine defines the formatting rules of the element displaying layout in the service browsing canvas such as positioning, margin, etc. The Layout Engine in our proposed MDTV service platform is a customized, extensible layout engine specially developed and optimized for XHTML based MDTV service retrieving in the terminal devices with different screen sizes and resolutions. It assists the Drawing Subsystem in the Home Canvas to paint the MDTV service elements including text, images, A/V, hyperlink and other interactive events in their corresponding place and with regards to the different screen features of the MDTV terminal devices. Besides, when there is a CSS file available with the XHTML based service code, the Layout Engine has also incorporated an interface for an enhanced extension to support a CSS template layout mechanism.

The Media Handler

Media Handler is developed based on the JSR135 Mobile Media API. It provides access to the media codec in the MDTV terminal device and retrieves the Audio/Video event elements in the MDTV service. It also defines the media control actions including initialize, play, pause and stop that assist the Home Canvas to implement the relevant and required interactivities.

4.4.1.4 The Parsing Block

The Parsing Block is responsible for parsing all the MDTV service codes in XHTML and the initial parsing results are sent to the Rendering and Retrieving Block to be displayed. Like as the Layout Engine, the Parsing Block is also a customized parsing system that is specially developed following our proposed service presentation method based on XHTML and “ID Event” method. This system consists of an XML and XHTML Parser, Tag Dictionary and Tag Factory.

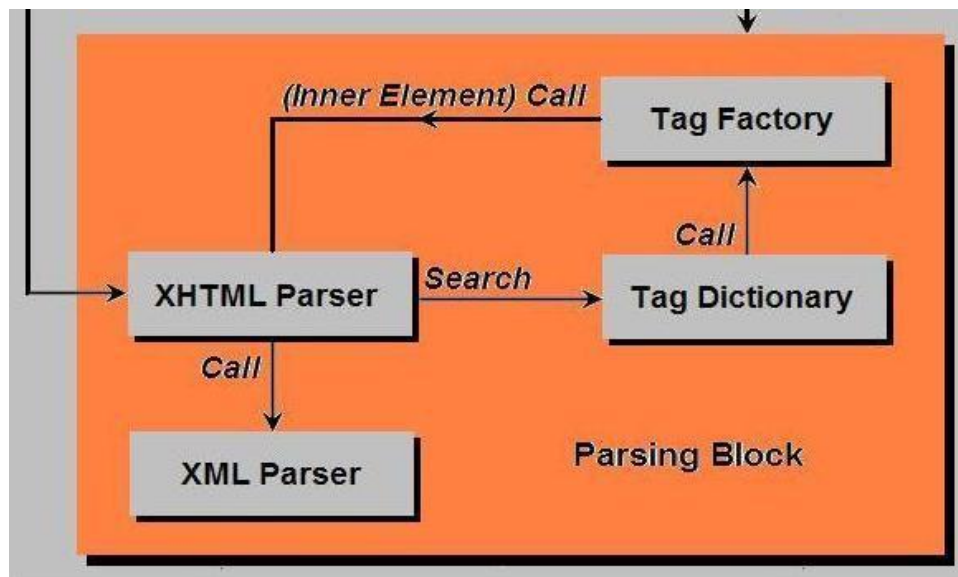


Figure 4.10: Parsing Block class diagram

The XML and XHTML Parser

As discussed in the previous section, this XHTML parser is developed based on a XML parser called kXML. The XHTML parser is in charge of receiving the XHTML code stream through an IP network, parsing the code with the help of the kXML parser, sorting and rendering the XHTML code in cooperation with the Tag Dictionary and Tag Factory, and finally sending the rendered result to the Home Canvas for displaying. The whole process of parsing is explained in the following section.

The Tag Dictionary

The Tag Dictionary defines all the element tags and attributes that are defined in the

XHTML specification as well as the ID Event element tags and attributes. It also defines the element sorting algorithm for triggering different parsing methods in the Tag Factory. The Tag Dictionary is an assisting class to the XHTML parser.

The Tag Factory

The Tag Factory is an aggregation of the parsing method for different types of elements in the XHTML service code including plain text, hyperlink text and image. It also contains a database for storing parsing results in order to provide processing references to the Rendering and Retrieving Block. The Tag Factory is also an assisting class to the XHTML parser.

Data Flow in the Parsing Block

As illustrated in Figure 4.11, after receiving the XHTML based MDTV service code, the Parsing Block starts parsing them.

1. The XHTML parser calls the XML parser to initially parse the XHTML code referring to the syntax of XML. Since the XHTML has the same format as XML, the XHTML code is therefore recognizable to the XML parser. All the elements in the XHTML code are pulled out and saved as the draft parsing result along with their names and attributes;
2. The XHTML parser then retrieves this draft parsing result and starts the XHTML parsing loop. The draft parsing result is firstly traversed for the first round by the XHTML parser and the corresponding tag sorters in the Tag Dictionary are called according to the element names during the traversal;
3. The tag sorter in the Tag Dictionary then refers to its tag name definition and calls the relevant parsing methods in the Tag Factory for final parsing;
4. In the Tag Factory, different methods are called and the attributes are pulled out and

matched to their corresponding elements;

5. Every time when an element is parsed, a logic judgment is performed: if the element has child elements, all the children are sent back to the XHTML parser and the parsing loop starts again from step 2; if there is no child, the element is saved in the element database for rendering process.

6. The XHTML parsing loop repeats until all the child elements are traversed. The element database is updated along with the parsing loop.

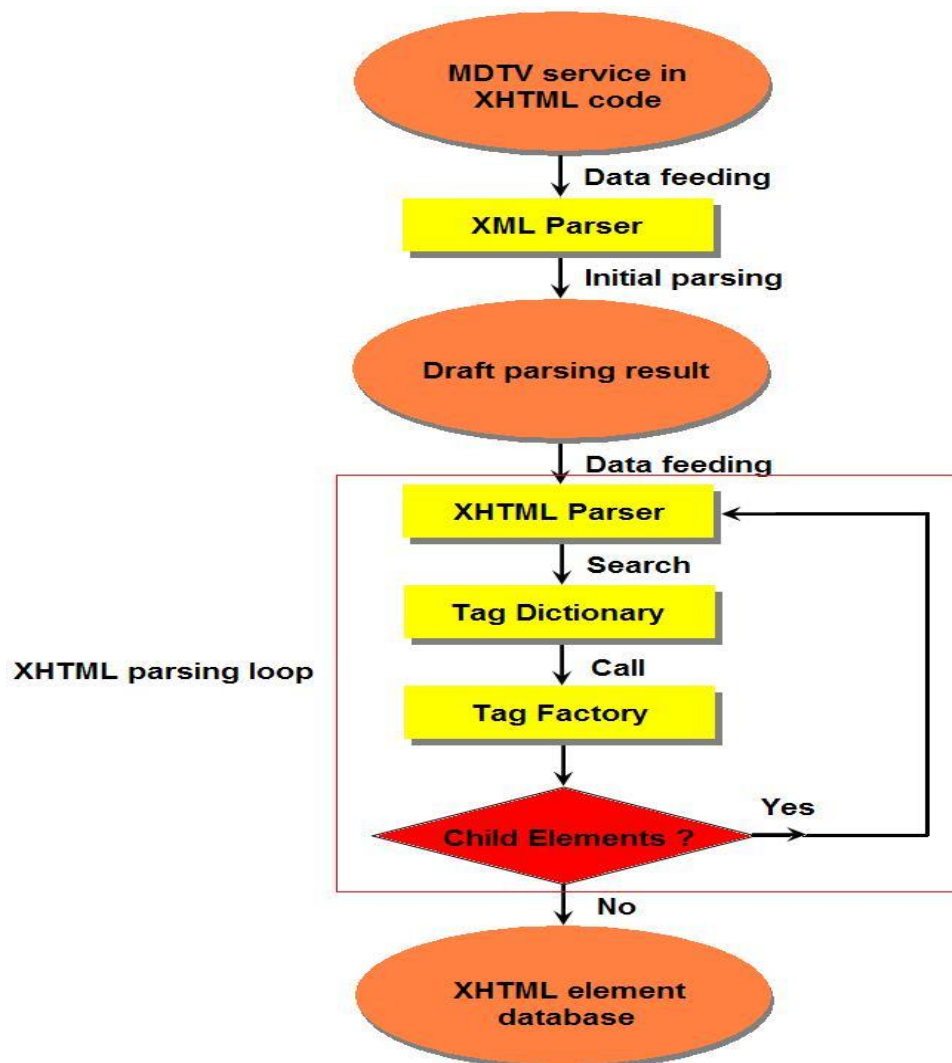


Figure 4.11: Data flow diagram in the Parsing Block

4.4.1.5 The ID Event Manager

The ID Event Manager handles all the ID Events that are pre-defined MDTV events

for local and remote interactivity such as voting or live data feeding. When a component with an ID Event in the Home Canvas is selected, the event detector passes the ID Event element to the ID Event Manager along with its name and attributes for further process. The ID Event Manager consists of two classes: the ID Event Handler and the ID Event Factory. The **ID Event Handler** receives ID Event elements and calls the event handling method in the ID Event Factory. The **ID Event Factory** defines all the MDTV interactive application execution methods that are supported by the service platform. Different application execution methods are called according to the ID Event element name and the event execution result are passed back to the Home Canvas for relevant scene update. In case there is a need of data exchange between the service platform and the interactive service managing server in an interactive application, the ID Event Factory calls the Data Exchange Manager for assistance.

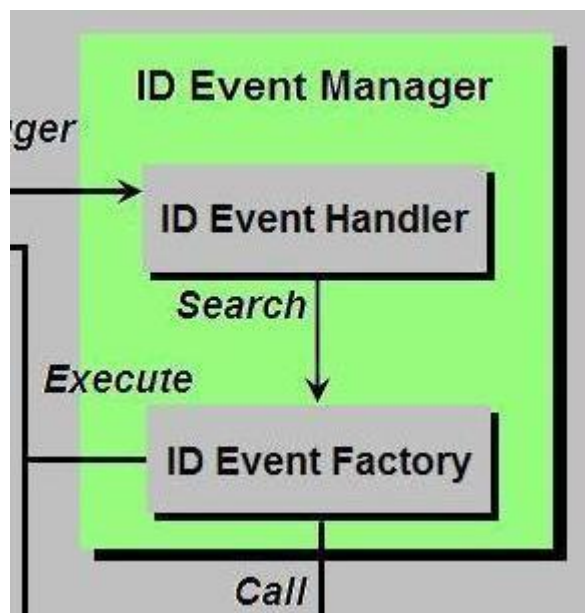


Figure 4.12: ID Event Manager

4.4.1.6 The Data Exchange Manager

The Data Exchange Manager handles the data exchange of an ID Event such as voting and live data feeding. It consists of the Data Exchange Client and Sender. When the data exchange is required, the **Data Exchange Client** sets up a connection between the interactive service managing server and itself through the MDTV service return

channel. The **Sender** is called for sending the interactive request from the client to the server. The client passes the replying data back to the Home Canvas for scene update like any other ID Events.

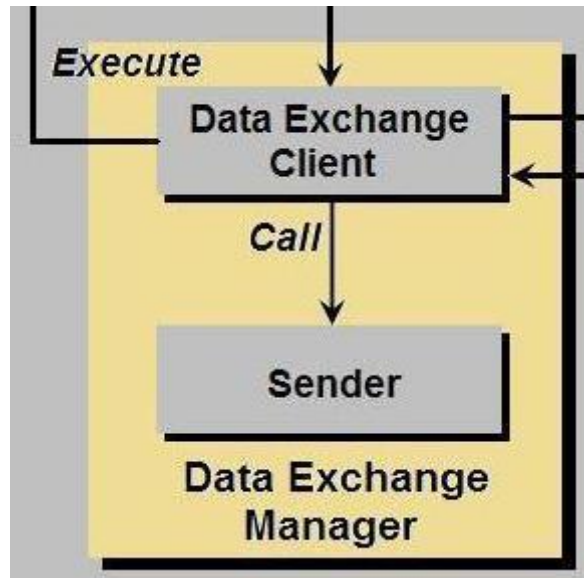


Figure 4.13: Data Exchange Manager

4.4.2 THE INTERACTIVE SERVICE MANAGING SERVER IN MDTV-CIE

A fully implemented MDTV server is a typical element in the MDTV service as a subset provided by the MDTV Service Provider (MSP) through the return channel. This server provides the server platform for most of the MDTV interactive services such as voting, online gaming, live data feeding, online shopping, user customized personal profile, service content download or user-generated content uploading. Moreover, the server usually has the customer account administration system that processes all customer requests, manages the customer profiles, and provides storage for customized contents or preference as well as the billing system. Besides it also in charge of transmitting updated data to the client service platform for service maintenance and update issues like the ESG update, service platform update. Our proposed Interactive Service Managing Server is an initial version of such MDTV server with fundamental functions including connection setup, and data exchange and a storage mechanism. It is designed as an extensible platform that allows further enhancement for different service applications in the future. Currently, the Data

Exchange Server is the core functional class in the managing server.

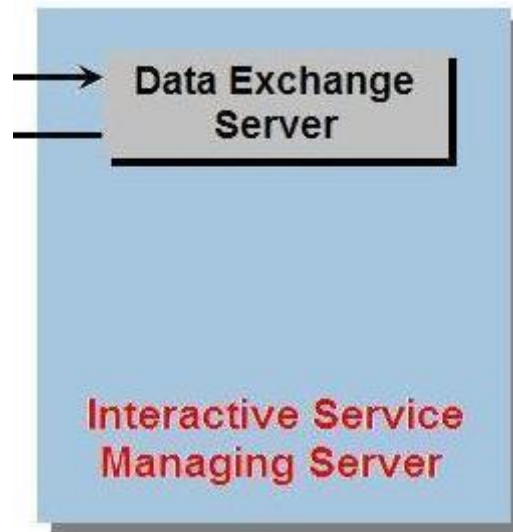


Figure 4.14: Interactive service managing server

4.4.2.1 The Data Exchange Server

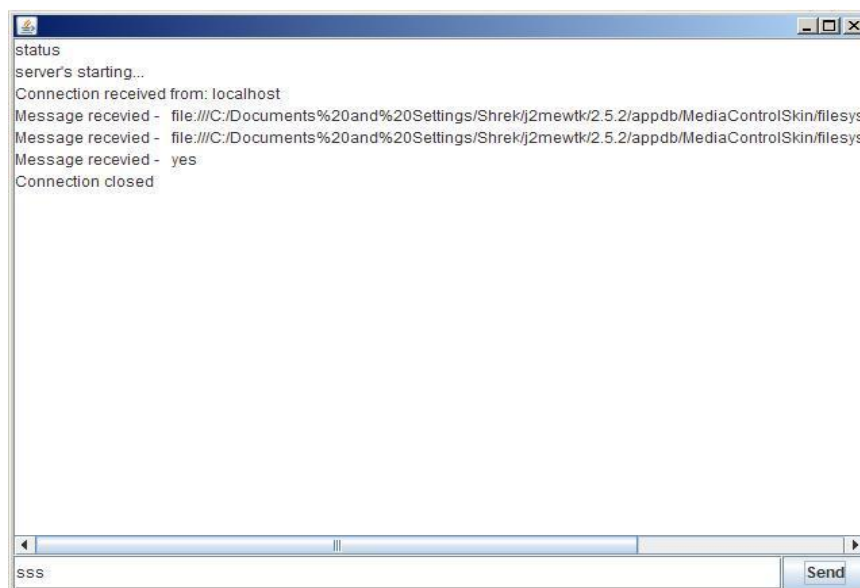


Figure 4.15: The Data Exchange Server

The Data Exchange Server defines the methods for connection setup and data exchange between the MDTV service server and client. It is developed in Java SE and designed in the form of a console platform as illustrated in Figure 4.15. When the server starts, it monitors any connection requirements from the client service platform through the MDTV service return channel and once a requirement is received, an IP connection is setup. Through the IP channel, the Data Exchange Server is able to

communicate with the Data Exchange Manager in the client service platform for handling any ID Event and in turn achieving the MDTV interactive service. Moreover, there is a database at the back-end of the Data Exchange Server for temporary or long-term data storage such as voting statistic results and user personal profiles.

4.4.3 SERVICE IMPLEMENTATION DATA FLOW IN MDTV-CIE

Regarding the service implementation, there are mainly seven user cases during the implementation process (as illustrated in Figure 4.16). Moreover within this seven user cases, the MDTV service application experiences three statuses: reception and storage, running, and updating.

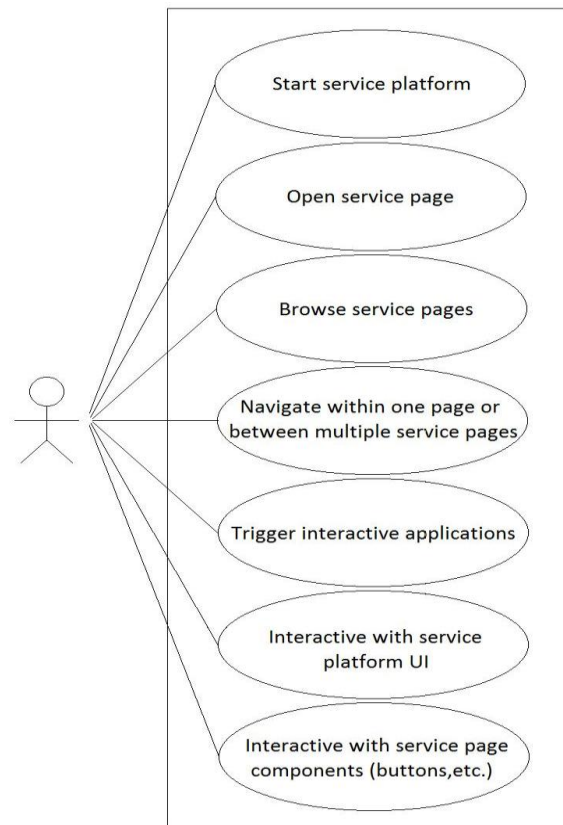


Figure 4.16: UML User Case Diagram of service implementation in MDTV-CIE

Service application reception and storage in MDTV-CIE

Once a service URI requirement is sent from the service portal such as the Browser MIDlet or URL Input to the MDTV server, or a service option is selected in the ESG,

the required service application is received in the form of XHTML source code from the MDTV server and stored temporarily in the terminal device memory. The service application XHTML source code can either be synchronized with the MDTV broadcasting A/V stream or can be downloaded from the MDTV server such as the Interactive service managing server.

Service application running in MDTV-CIE

Then the service portal calls the Parsing Block to parse the XHTML source code and the service XHTML source code is parsed into XHTML elements and saved in the element database for presentation on the screen. The Rendering and Retrieving Block then traverses the element database and paints all the XHTML components including text, image, and interactive events along with A/V on the screen scene.

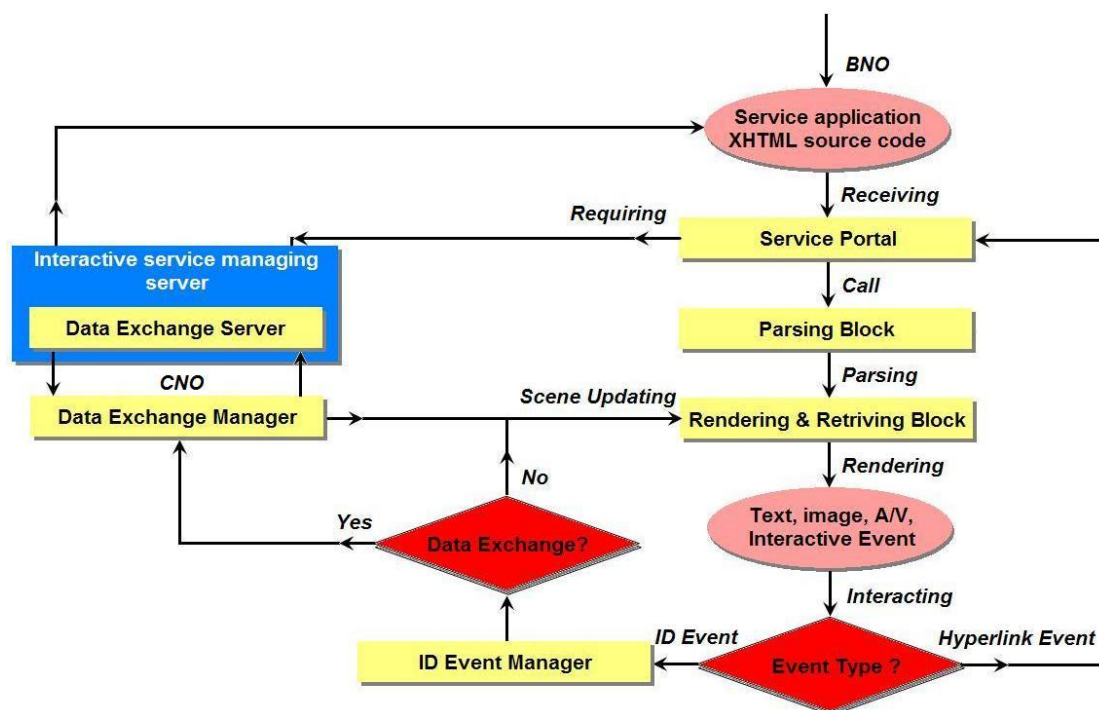


Figure 4.17: UML Data Flow Diagram in the MDTV-CIE

Through the I/O units such as monitor, keyboard or pointer in the terminal device, users can consume the MDTV service by watching the A/V content, reading the auxiliary text and image information and interacting with the interactive components or applications provided within the service platform. The local interactivity is handled

by the 2D Navigation subsystem of the service platform whilst once a remote interactive component or application is selected or activated, a logic judgment is performed by the event detector in the Home Canvas. The plain Hyperlink Event is passed back to the service portal and the service portal sends a relevant URI requirement to the Interactive service managing server for further service contents; ID Event will be passed to the ID Event Manager where another logic judgment is performed to decide whether the event needs a data exchange process. If the event is a local application, the ID Event Manager will call the relative program to implement the event; if the event requires a data exchange process, the ID Event Manager will call the Data Exchange Manager for assistance, which is achieved through the communication between the Data Exchange Server subset in the Interactive service managing server and the Data Exchange Manager on the service platform. All the ID Event rendering results are then passed back to the Rendering and Retrieving Block for relevant service scene update such as a voting confirmation message and a live data feeding update.

When the ID Events are passed to the ID Event Manager, the manager will search in itself for the event programs corresponding to the event ID. These event programs are the applications developed in Java ME especially for the MDTV interactive service, and by using our proposed service creation tool, those programs are then either pre-embedded in the terminal device or downloaded to service platform according to the actual MDTV service requirements.

Service application updating in MDTV-CIE

The MDTV service application based on the XHTML and ID Event model can be updated by sending regular requirements to the Interactive service managing server for the latest version. The ID Event Manager in the terminal device service platform is updated at the same time if necessary for any additional interactive application functionality. The update of the ID Event Manager may require an update process to the entire terminal device platform, which may be achieved through either the MDTV

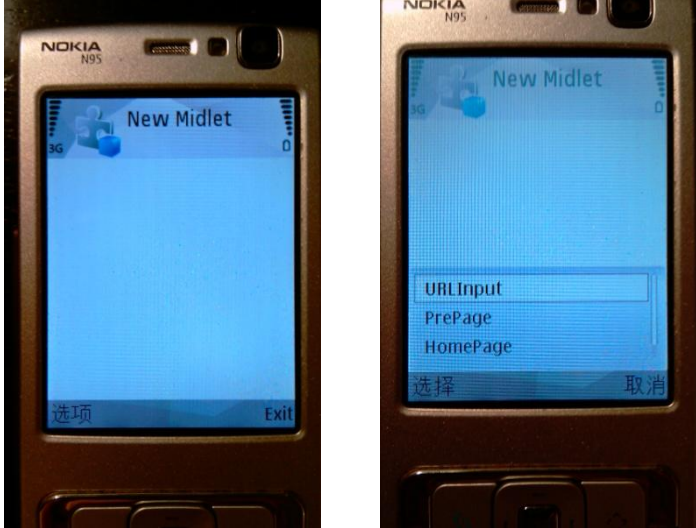
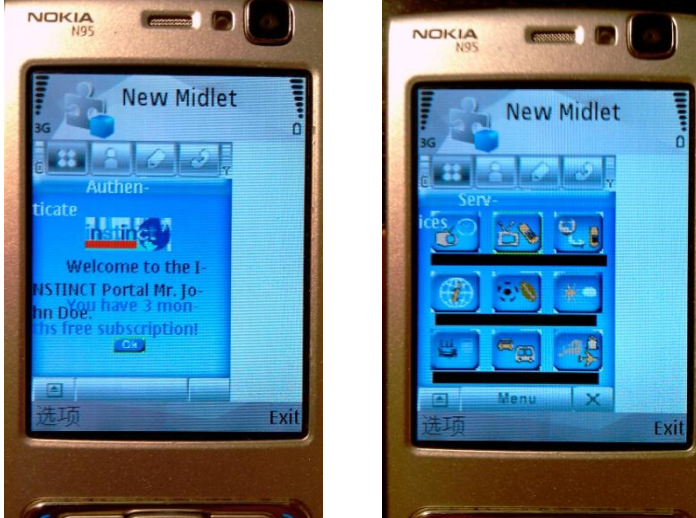

service broadcasting or the MDTV return channel.

4.4.4 TEST THE PROPOSED SERVICE PLATFORM IN NOKIA N95

Having developed the terminal device service platform through the Table PC based emulator, it is necessary to run the software in a real-device environment to assess its performance and functionality in practice.

As one of the biggest mobile phone manufacturers in the world, Nokia has been leading the development position of mobile phones with the latest technologies being deployed. As the Java ME environment has become popular, Nokia turned out to be one of the first few brands that started providing support to it. By now, this support has been well developed and deployed among in Nokia series mobile phones. Thus it is considered to be suitable and necessary for designers to test any software in Java ME within such hardware/software environment. Nokia N95 is a smartphone as part of their Nseries portable devices. It was released in 2007 and has been a successful product in the nowadays' mobile phones market with excellent performance and powerful functions. Its support to Java ME is also a default configuration and the supported Java ME version is MIDP 2.0 and CLDC 1.1. Regarding all these aforementioned advantages of Nokia N95, we thus chose it as the real-device test platform.

A validation test of the proposed MDTV terminal service platform has been conducted on the Nokia N95 handset. The testing has been conducted locally on the mobile phone with all services being accessed locally from the handset file system rather than accessing them from the server. The purpose of such test is to run the proposed service platform in a real hardware and software environment and evaluate the software in terms of performance. The test cases for the interactive applications of the service pages that require client/server connection were excluded in this stage. Table 4.2 shows the test case steps and the corresponding screenshots.

Test on Nokia N95		
Case Design	Screenshots	Testing result
Step1: Start up the service platform.	 <p>Figure 4.18, 4.19: Main UI of the software</p>	successful
Step2: browse the service pages by pressing the corresponding hyperlinks	 <p>Figure 4.20, 4.21: Service pages displayed in the platform</p>  <p>Figure 4.22: Service platform running after the screen is switched to landscape mode</p>	Succeeded in loading the pages but failed on arranging the page layout properly.

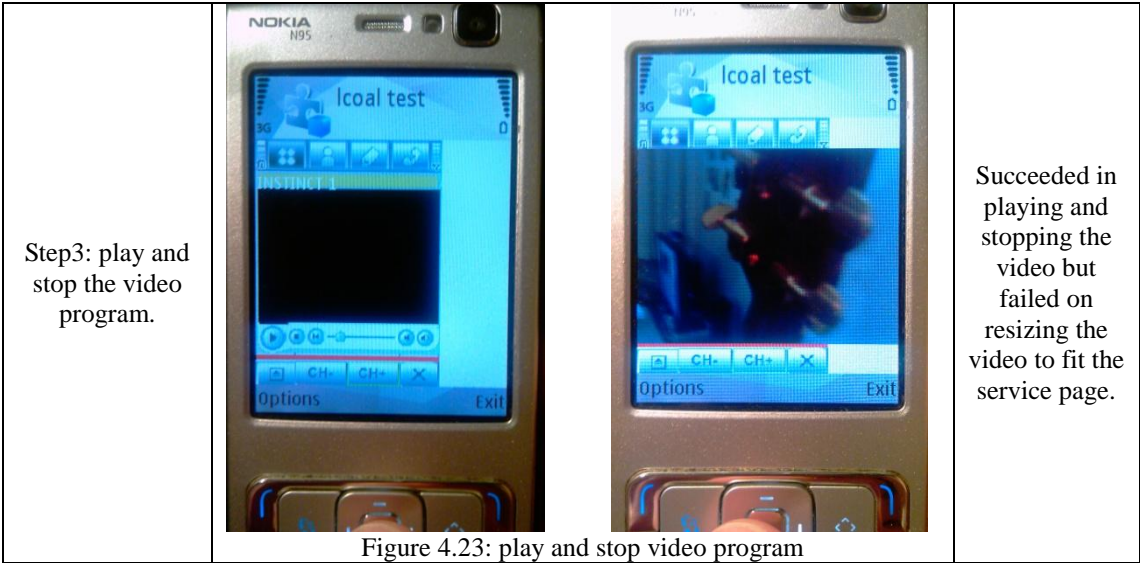


Figure 4.23: play and stop video program

Testing results evaluation

The test case has been successfully executed on the Nokia N95 handset. All the result outputs are as expected. Issues and errors occurred on two aspects:

1. The duration that the platform has spent on loading the service pages is much longer (about 2 minutes per page) than it spends in the Java ME emulator on the PC (about 1 second per page). The reason of this is considered to be the execution efficiency of the software source code in the N95 handset environment. Another important reason is because that the security permission of the software is low in the handset as the program is not signed and registered. This caused a confirmation message popped every time the terminal platform intended to access the local file system, which result in extra time spent during the parsing process.
2. Two errors occurred due to bugs in the layout engine of the service platform.

Table 4.2: Validation testing of the proposed service platform on the Nokia N95 handset

4.4.5 COMPARISON AND DISCUSSION

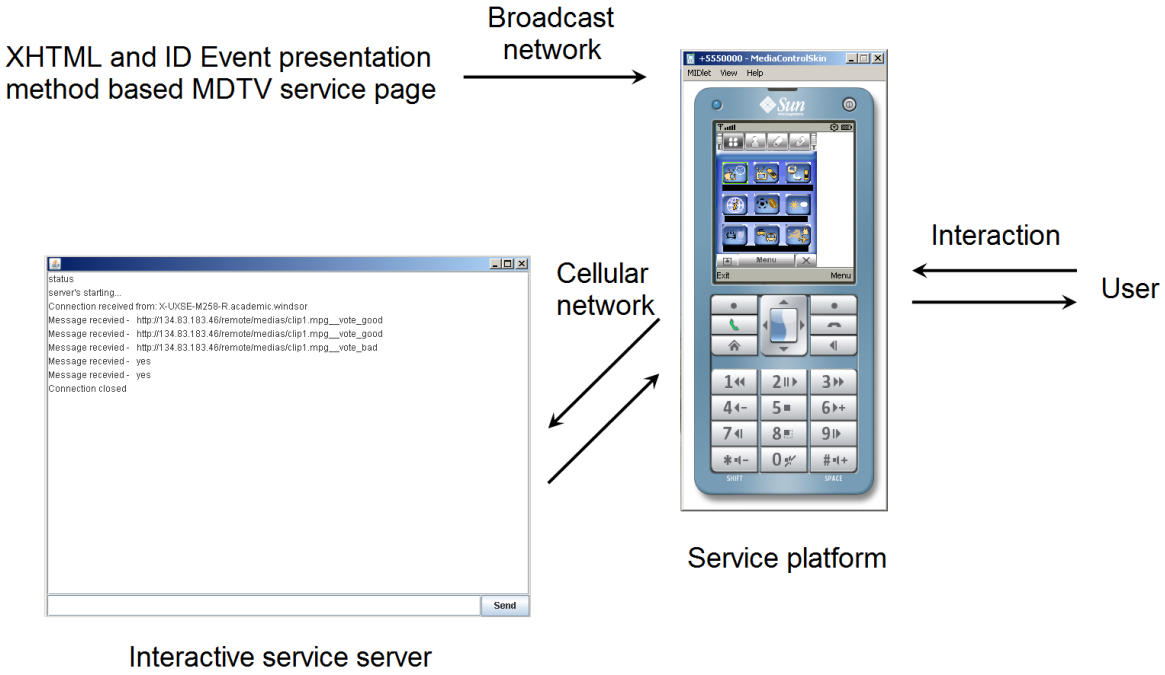


Figure 4.24: Functionality of MDTV-CIE

The implementation of the MDTV-CIE assists the proposed semi-automatic service creation process by rendering the MDTV interactive service created based on the XHTML and “ID Event” presentation method through the proposed creation tool; it offers the user a platform to interact and consume such MDTV service. Moreover by doing this, it further completes the proposed MDTV service creation and consumption system as a universal and improved MDTV implementation solution. Regarding to the functionality, although the MDTV-CIE components do not perform the functions as comprehensively as those relative commercial solutions mentioned in Chapter 2, there are still advantages and potential benefits when implementing it in the entire proposed service creation and consumption software system:

Most of the MDTV standards as well as commercial solutions have employed the ECMAScript as the basis of their interaction handling mechanism during the service creation and they use different technologies such as Rich Media or C during the terminal platform development. In contrast in the proposed solution, the service platform is developed by using the same technology, the Java ME, as the interaction handling method called “ID Event”. At the same time, the proposed solution can implement most of the MDTV services and applications as those commercial solutions do (will be discussed in Chapter 5). In this case, fewer technologies are used whilst the functionality remains the same. This can further result in the relative easier service creation and the architecture of the service platform can be relatively simpler to design and implement.

Moreover, the proposed service platform is specially design according to the MDTV service implementation requirement and its architecture is simpler than those commercial service platforms based on their web browser products. The maintenance and update can thus be easier. Moreover, the functionality of such customized service platform focuses mostly on what the MDTV service concerns and thus the execution of MDTV service/application can be more efficient than aforementioned browser-

based commercial solutions.

Summary:

This chapter further presents the client implementation environment of the proposed MDTV service creation and consumption system. As another core component in the proposed system, the terminal device service platform has been developed with a fully-customized browser-based structure according to the proposal's requirement. A test of such service platform in a real-device environment is conducted after the completion of the development work. A few advantages and potential benefits of the proposed service platform are briefly discussed in the end when compared with other current solutions.

5. CHAPTER 5: TEST AND EVALUATION

5.1 INTRODUCTION AND MOTIVATION

Having presented and discussed the implementation of the proposed solution for MDTV service creation and implementation in chapter 3 and 4, this chapter tends to follow the MDTV-SPM model (refer to Fig 2.22) and a software testing procedure to conduct a functionality and validation evaluation on our proposed components including the semi-automatic service creation tool and the terminal device service platform. The motivation for this testing procedure is to evaluate whether each of the proposed components is able to work well as integrated software system, whether each of the proposed software components is able to achieve their functions, and whether these functionalities meet the aim and requirement of the proposed solution. Also by conducting this testing procedure, we intend to gain useful test results as the reference for further modification and enhancement to the target software in the future work.

However, since we have proposed these two software components only at a prototype stage, which mainly focus on the implementation of the basic functionalities according to the design requirements, a series of testing cases will be designed and implemented only for the qualitative evaluation of the software performance on their corresponding functions rather than applying it for every aspect of the software testing technologies. The test procedures to be conducted in this chapter aim mostly at verifying and validating the target software on the initial level and evaluating them to be usable regarding to the requirement of the propose solution. Moreover in the future work, when modifications and improvements are done to the proposed components according to the testing results gained from this chapter, the comparison testing between the proposed components and the relative commercial solutions, the in-depth quantitative testing and other necessary testing procedures will then be further conducted.

Therefore, the aims of this testing section can be further stated as follow:

- Test the semi-automatic service creation tool to evaluate: if the target software is able to work properly as a integrated software without major errors; if the target software is able to achieve its functionality to meet its design requirement and eventually enable the proposed semi-automatic MDTV service creation process.
- Testing the terminal device service platform to evaluate: if the target software is able to work properly as a integer without major error; if the target software is able to achieve its functionality to meet its design requirement and eventually render the MDTV interactive service page based on XHTML and “ID Event” presentation method.
- Testing the semi-automatic service creation tool and the terminal service platform in the proposed MDTV service creation and consumption system (illustrated in Figure 2.23) to evaluate: if these two software components are able to work together properly in the system without major error; if these two software components can further enable the proposed system to achieve the aim of the Thesis.

5.2 TESTING METHODOLOGY

According to the software testing principles presented in Chapter 2 and the reference software testing level flow diagram illustrated in Figure 2.20, the testing process for each software target will start from the component testing level and will be completed with the acceptance testing. On each level, the testing case will be designed according to the recommended Black-box – White-box testing strategy. Besides, the integration testing and the system testing will be conducted manually rather than automatically due to the complexity of the proposed system prototype. In short, three test sub-sections are included according to the test aim: 1) software testing to semi-automatic service creation tool; 2) software testing to terminal device service platform; 3) integration testing to both of the software targets in the proposed system.

5.2.1 TEST AND EMULATION ENVIRONMENT

All the testing processes are to be implemented within a Desktop PC environment, where the hardware and the software configurations are listed in Table 5.1. The Eclipse Integrated Development Environment (IDE) v3.30 is the main development environment for both of the proposed software. By importing necessary Software Development Kit (SDK) such as Java SE and Java ME SDK in our case, developers can use this IDE to develop the target software applications visually and conveniently. The Eclipse IDE v3.30 is thus chosen as the test execution environment throughout the entire test process of this chapter.

Within the Eclipse IDE, the semi-automatic service creation tool is developed with Java SE Development Kit and thus it will be run in Java Virtual Machine (JVM) included in jdk 1.6.0 during the testing procedure. Whilst for the terminal device service platform that is developed with Java ME Development Kit, an emulator is required besides the runtime environment. As a Java ME program usually runs within a portable device environment that has different hardware and software configuration from a Desktop PC. Thus in order to develop and test a Java ME software application in a Desktop PC, the emulator must be used to simulate the hardware and software environment of a portable device so as to able to run the application in it and evaluate how well the application has been designed. Therefore the emulator included in the Java WTK V2.5.2 is utilized to run the terminal device service platform during its relevant testing procedures.

		Test sections	
		Test on semi-automatic service creation tool	Test on terminal device service platform
Hardware	CPU	Intel Pentium Dual-core E5200 2.5GHz	
	Memory	2GB	
Software	Operating System	Windows XP sp3; Windows 7	
	Runtime Environment	Java SE Runtime Environment jre v6	
	Development Kit	Java SE Development Kit jdk 1.6.0 update 20	Java ME platform Software Development Kit v3.0; Java Wireless Toolkit for CLDC v2.5.2_01
	Integrated Development	Eclipse IDE v3.30	

	Environment	
	Assistant software	Windows Internet Explorer 7; Windows Wordpad

Table 5.1: Testing and emulation environment

5.2.2 TEST CASE DESIGN OF SEMI-AUTOMATIC SERVICE CREATION TOOL

The aims of the test to the semi-automatic service creation tool are:

- To evaluate if the target software is able to work properly as a integer without major error;
- To evaluate if the target software is able to achieve its functionality that the designer can use it to manipulate and create the MDTV interactive service page based on XHTML and “ID Event” presentation method semi-automatically;
- To evaluate if the target software can enable the proposed semi-automatic MDTV service creation process by outputting the XHTML and “ID Event” based MDTV interactive service page as well as a new version of the corresponding functional class for the proposed terminal device service platform.

To implement these aims, a series of test cases along with the testing data are then designed for different testing levels from component testing to system testing and different testing methods including Black-box testing, White-box testing and GUI testing.

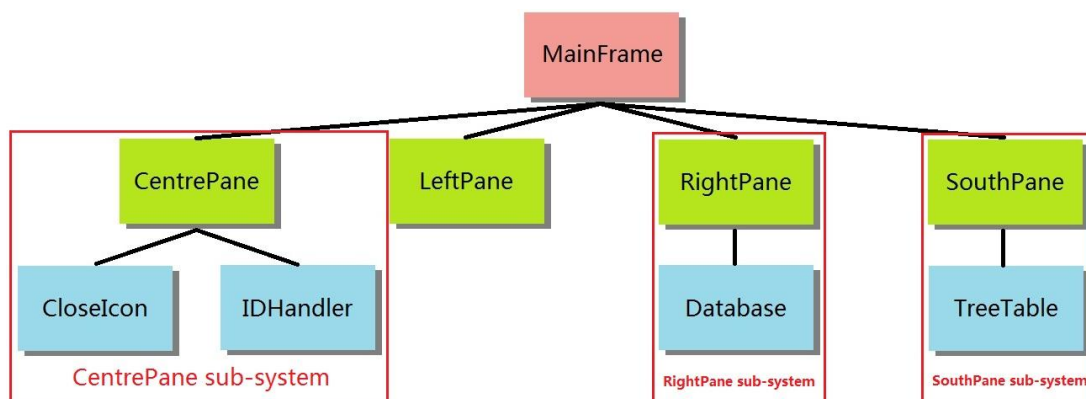


Figure 5.1: Class Components of the semi-automatic service creation tool

Testing data

Here are all the service pages, which will be used during the testing cases on different

testing levels (All the figures are the screenshots of the testing service pages viewed through Microsoft Internet Explorer 7):

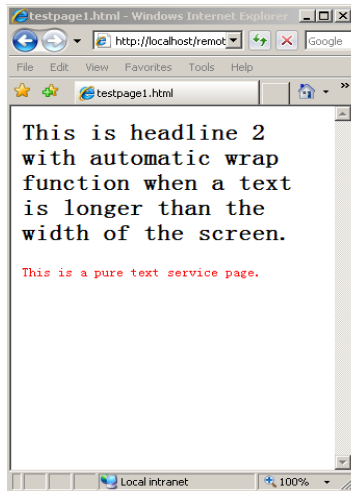


Figure 5.2

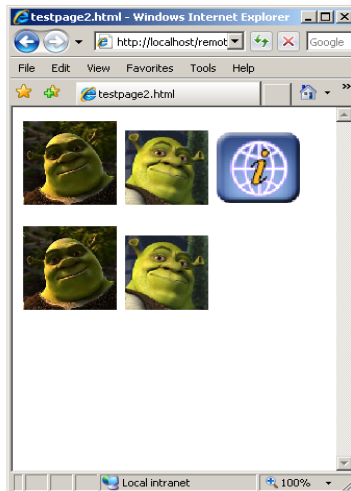


Figure 5.3

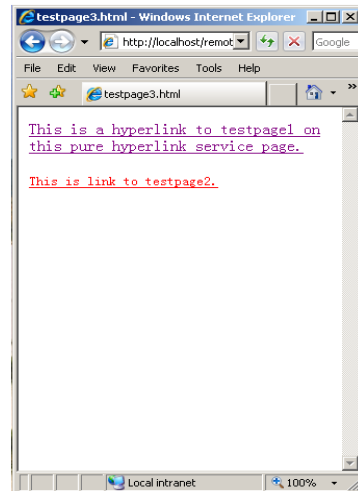


Figure 5.4

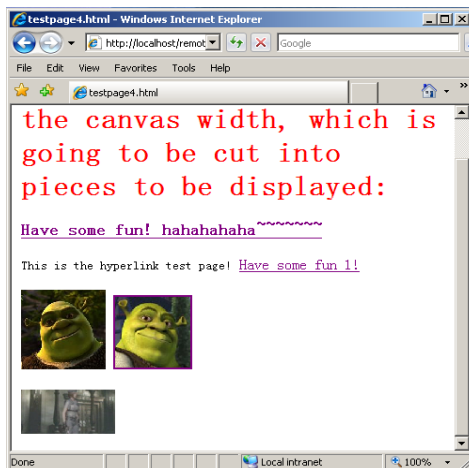


Figure 5.5

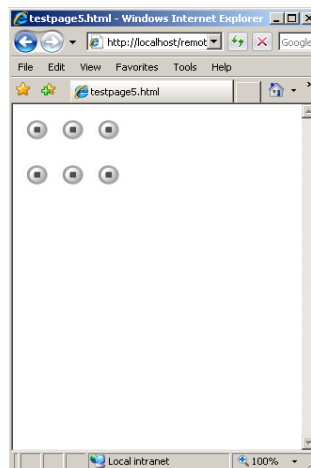


Figure 5.6



Figure 5.7



Figure 5.8



Figure 5.9



Figure 5.10

Figure 5.2: Testpage1.html: pure plain text page.

Figure 5.3: Testpage2.html: pure image page.

Figure 5.4: Testpage3.html: pure hyperlink page.

Figure 5.5: Testpage4.html: mixed components page (text, image and hyperlink).

Figure 5.6: Testpage5.html: test page for testing interactive applications.

Figure 5.7: Testpage6.html: service homepage

Figure 5.8: Testpage61.html: service portal.

Figure 5.9: Testpage7.html: multimedia service page with video

Figure 5.10: Testpage8.html: program guide

“Testpage5.html” is a plain XHTML service page without any interactive application added on in this testing section.

Component testing

Since the proposed semi-automatic service creation tool was designed mainly for the visual editing of content with most of its functions being implemented through a GUI, this software system was thus developed according to its GUI structure. As shown in Figure 5.2, the service creation tool system consists of 9 class components: the MainFrame, CentrePane, LeftPane, RightPane and the SouthPane as the GUI elements and the CloseIcon, IDHandler, Database and TreeTable as the assisting components. Since the component testing work on these classes has been done while they were being programmed, it can be concluded that the design requirement specification (shown in Table 5.2) of the proposed software system has been met as expected.

Class Components	Design Requirement
MainFrame (GUI component)	This is the main software frame that holds all the other GUI components; has a menu and includes hotkeys to open and save the target file.
CentrePane (GUI component)	This is the main editing area that displays a service page as well as its source code; it allows a multiple-page display mode; it allows conventional page editing and interactive MDTV service configuration;

	it allows a “drag-and-drop” function from the RightPane.
CloseIcon	It assists the CentrePane by providing visual feedback of the “close tab” button.
IDHandler	It assists the CentrePane by associating with the RightPane to help achieve the interactive MDTV service configuration at the back-end; it allows the necessary parameter to be input through a pop-up dialogue window.
LeftPane (GUI component)	This is the main project file system that displays all the project files and folders in a list format; it allows the use of double-click to open the target file; also the file system allows conventional navigation (select, go up and down on the list, go into sub directory, etc.) and real-time/manual update.
RightPane (GUI component)	This is the main interactive application database that displays all the available MDTV interactive applications in a list format; All the applications in the list are selectable for further operations; it also supports display synchronization with Database when there is any change in the Database; the list allows “drag-and-drop” to the CentrePane.
Database	It assists the RightPane by storing all the available interactive application source code. It has an extension function to allow more new applications to be stored.
SouthPane (GUI component)	This is the main service page parameter list that displays all the elements and their corresponding attributes in the service page source code; it supports display synchronization with the CentrePane when switching between different pages as well as when making any modification to the page.
TreeTable	It assists the SouthPane by providing a modelling mechanism for the element display in the list; it provides connection with LeftPane and the CentrePane for file synchronization.

Table 5.2: Design requirement specification for the semi-automatic service creation tool

Integration testing

In this section, we chose “bottom-up integration”, which starts integration and testing from the lowest level of the system to the upper level, as the integration method. In our test case, all the lowest level class components, depicted in Figure 5.1, were then integrated with the upper level classes into functional sub-systems and the relevant testing procedure was also conducted during the development process. As illustrated in Figure 5.1, all the functional sub-systems are: the CentrePane sub-system, the LeftPane, the RightPane sub-system, and the SouthPane sub-system. All these sub-systems will be integrated with the MainFrame to form the complete service creation tool system. On this integration level, we design the test cases mainly focusing on the interfacing between the different sub systems (shown in Figure 5.11) to ensure they can work together properly. The testing cases are designed and listed in Table 5.3.

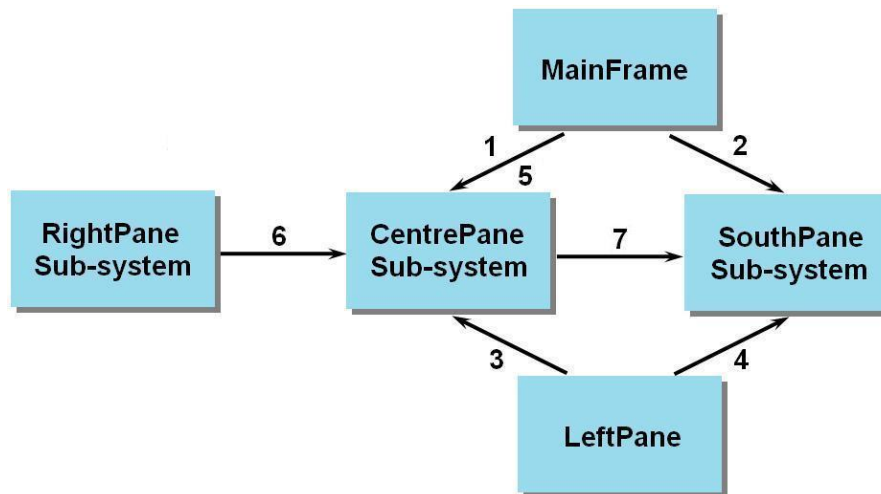


Figure 5.11: Interfaces between sub systems

Testing cases	Input	Expected output
Interface 1: MainFrame – CentrePane	1. Select “File – Open XHTML File...” to get a pop-up window, select the target XHTML (.html) file in the window to open the target service page; 2. Repeat the above operation to open another service page.	1. Service page appears in the CentrePane upper frame with all the elements (text, images and hyperlinks) displayed; the corresponding source code appears in the lower frame of CentrePane; A title tab is integrated on the top of editing field with the target page name on; 2. When more than one page is displayed, they overlap into a tab list format.
Interface 2: MainFrame – SouthPane	1. Select “File – Open XHTML File...” to get a pop-up window, select the target XHTML (.html) file in the window to open the target service page; 2. Repeat the above operation to open another service page.	1. The elements along with their attributes’ names and values of the target service page source code appear in the SouthPane in form of table; A title tab is integrated on the top of the editing field with the target page name on; 2. When more than one page is displayed, they overlap into a tab list format.
Interface 3: LeftPane – CentrePane	1. Double click on the target XHTML file name in the project file list within LeftPane to open the target service page; 2. Repeat the above operation to open another service page.	1. The service page appears in the CentrePane upper frame with all the elements (text, images and hyperlinks) displayed; the corresponding source code appear in the lower frame of CentrePane; 2. A title tab is integrated on the top of editing field with the target page name on; When more than one page is displayed, they overlap into a tab list format.
Interface 4: LeftPane – SouthPane	1. Double click on the target XHTML file name in the project file list within LeftPane to open the target service page; Repeat the above operation to open another service page.	2. The elements along with their attributes’ names and values of the target service page source code appear in the SouthPane in form of table; A title tab is integrated on the top of editing field with the target page name on; When more than one page is displayed, they overlap into a tab list format.
Interface 5: MainFrame -	1. Press “Element Mode” button on the hotkey bar in	1. The source code display frame in the CentrePane changes from “text mode”

CentrePane	the MainFrame to enable the “element modification mode”; 2. Press “Element Mode” button again to disable the “element modification mode”.	that displays source code in plain text to “list mode” that displays source code in an element list format so as to be prepared for interfacing with the RightPane; 2. The source code display frame in the CentrePane changes back from “list mode” to “text mode” for normal text editing.
Interface 6: RightPane – CentrePane	1. Switch the CentrePane to “list mode”; 2. Select “Stopvideo.java” in the interactive application list in RightPane; drag the selected application and move the mouse into the source code display frame in CentrePane; release the mouse left key to drop the application on the target element in the source code element list; 3. Switch the CentrePane back to “text mode”;	1. The source code display frame of the CentrePane changes from “text mode” to “list mode”; 2. A dialogue window pops up and asks for inputting relevant parameter; After the parameter is input, there is a new “id” attribute adding to the target element with a special interactive application id; 3. The new “id” attribute remains the same in text when the CentrePane changes back from “list mode” to “text mode”.
Interface 7: CentrePane – SouthPane	Switch between different service pages by selecting corresponding tab in the tab list on the top of CentrePane	The corresponding parameter table is shown on the front in the SouthPane while different target pages are selected in the CentrePane.

Table 5.3: Integration testing level test case design

System testing

After the sub components have passed the integration testing, system testing is necessary to ensure the software design requirements and functionality requirements are met. Therefore according to the Black-box – White-box testing case design strategy, the system testing has been designed to be conducted in two sections:

a) Black-box testing

We choose Equivalence partitioning as the black-box testing method. The functionality requirement of the proposed service creation tool is that the tool is able to configure and semi-automatically create the MDTV interactive service pages that follow our proposed XHTML and “ID Event” presentation method. Therefore there are mainly three elements on an XHTML service page that are available for adding interactive applications to. These are: plain text, images, and hyperlinks. Thus we divide the input into four partitions: pure plain text XHTML page, pure image

XHTML page, pure hyperlink XHTML page and XHTML page with all three types of elements mixed. The testing case design is shown in Table 5.4, with the purpose of testing whether the functionality requirement is met.

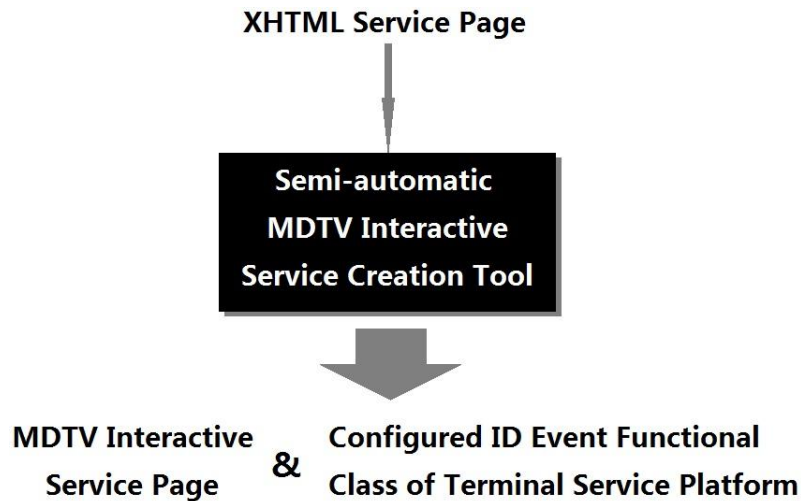


Figure 5.12: Black-box testing

Case No.	Equivalence partitioning input	Expected output
1	Pure plain text XHTML page (testpage1.html)	The pure plain text MDTV interactive service page is based on the XHTML and “ID Event” presentation method; Reconfigured Functional class of the terminal service platform as the second output.
2	Pure image XHTML page (testpage2.html)	The pure image MDTV interactive service page is based on the XHTML and “ID Event” presentation method; Reconfigured Functional class of the terminal service platform as the second output.
3	Pure hyperlink XHTML page (testpage3.html)	The pure hyperlink MDTV interactive service page is based on the XHTML and “ID Event” presentation method; Reconfigured Functional class of the terminal service platform as the second output.
4	XHTML page with all three types of elements mixed (testpage4.html)	The integrated MDTV interactive service page is based on the XHTML and “ID Event” presentation method; Reconfigured Functional class of the terminal service platform as the second output.

Table 5.4: System testing case design by using a Black-box method

b) White-box testing

As the design requirements to the components of the proposed service creation tool are all listed in Table 5.2, we can thus use the White-box testing method to test whether every component or sub-system has met the corresponding design

requirement. Here we choose Path testing as main white-boxing testing method and Loop testing as the secondary testing methods. Since the tool is mainly utilised for service creation, we thus choose the service creation process as the main logic path for the path testing. A MDTV interactive service page creation requirement has been defined firstly (see Table 5.5 and Figure 5.14) and the test case design can be also found in Table 5.6.

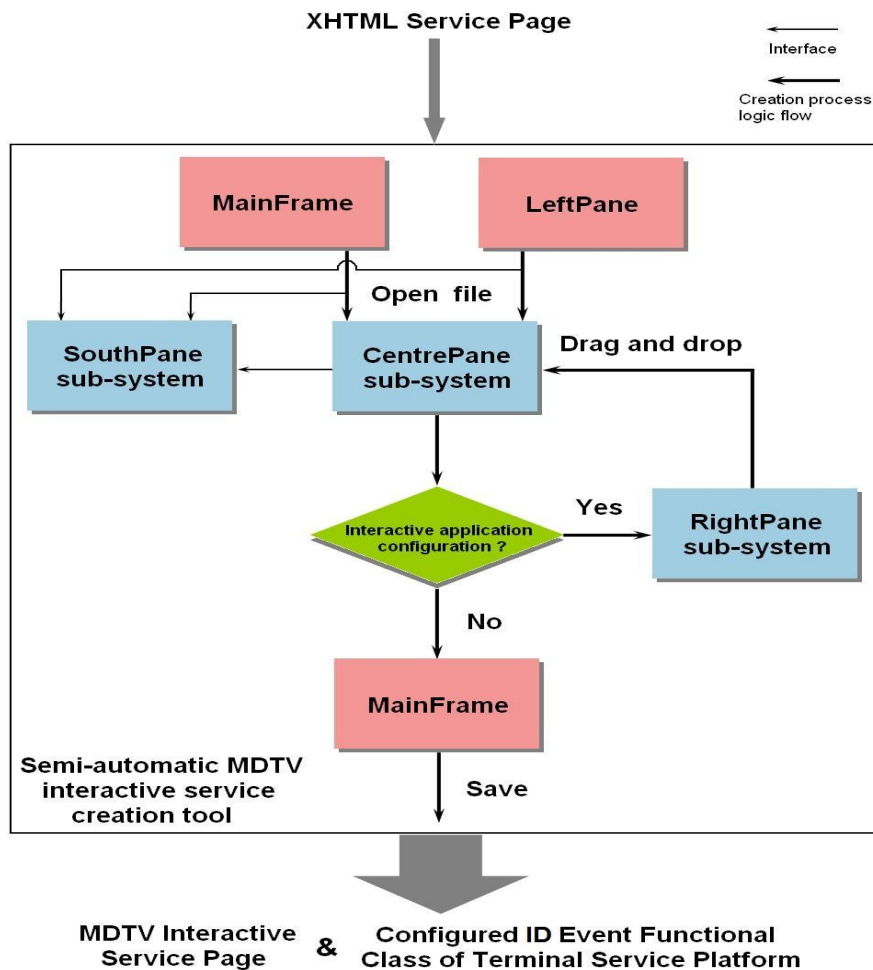


Figure 5.13: White-box testing

MDTV interactive service page creation requirement (for testing purpose)
1. Add “play video” interactive application to Element 1 on “testpage5.html” service page;
2. Add “stop video” interactive application to Element 2 on “testpage5.html” service page;
3. Add “request for real time data feed” interactive application to Element 3 on “testpage5.html” service page
4. Add “vote good” interactive application to Element 4 on “testpage5.html” service page
5. Add “vote bad” interactive application to Element 5 on “testpage5.html” service page
6. Add “request for vote report” interactive application to Element 6 on “testpage5.html” service page

Table 5.5: MDTV interactive service page creation requirements (for all the explanations to the supporting interactive applications refer to Figure 3.12 in Chapter 3)

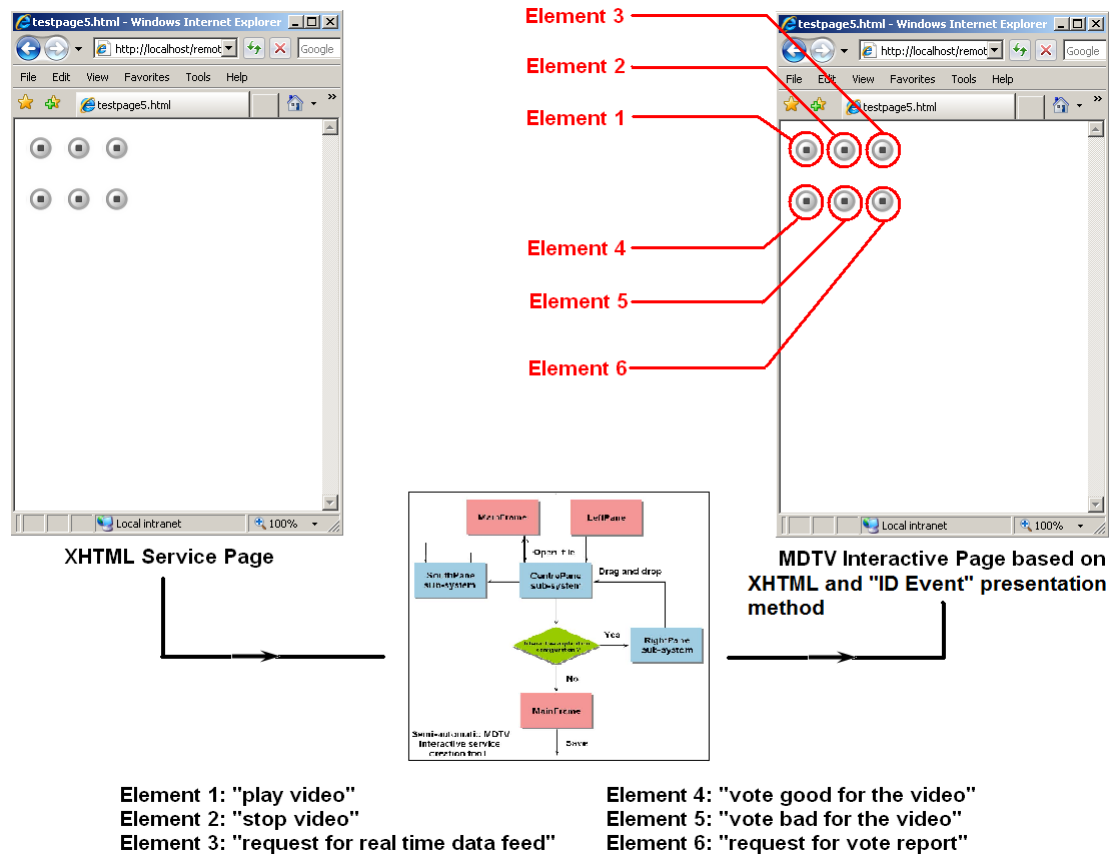


Figure 5.14: White-box testing requirement

Case No.	Testing Case	Design Requirement	Expected Implementation
1	1. Select "File – Open XHTML File..." to get a pop-up window, select the "testpage5.html" file in the window to open the test service page; 2. Double click on the "testpage6.html" in the project file list within LeftPane to open the target service page;	Import XHTML service page file.	1. When selecting "File – Open XHTML File...", a pop-up window will appear in the top middle within the MainFrame; The tester can navigate in the file system within the window; when selecting the "testpage5.html" file in the window, all the relevant content will appear in the CentrePane and SouthPane. 2. The tester can perform conventional navigation (up, down, select, go into deep folder and etc.) in the project file list within the LeftPane; when the left mouse button is double clicked on "testpage6.html" in the project file list, all the relevant content will appear in the CentrePane and SouthPane.
2	Monitor CentrePane and SouthPane while the tester is executing the Testing Case No.1	1. Display XHTML service page with the layout; 2. Display the service page source code; 3. The CentrePane	1. "testpage5.html" or "testpage6.html" XHTML service pages appear in the form of web-like page with the layout shown in the upper frame of CentrePane; 2. "testpage5.html" or "testpage6.html" XHTML service

		<p>supports multiple tab displays that each service page's content is displayed in a tab pane and the tab panes overlap when there is more than one;</p> <p>4. Display Service page elements and attributes in a table;</p> <p>5. The SouthPane supports multiple tab displays that each service page's content is displayed in a tab pane and the tab panes overlap when there is more than one.</p>	<p>pages' source code appear in the form of plain text in the lower frame of CentrePane;</p> <p>3. When "testpage5.html" and "testpage6.html" are both opened, the contents of each page appear in one individual tab pane (upper frame for the page and lower frame for the code); each tab pane has a corresponding title and a "close icon" and these two tab panes overlap and can be switched over;</p> <p>4. Elements and attributes of "testpage5.html" XHTML service pages are displayed in form of table list in the SouthPane and so is "testpage6.html" service page;</p> <p>5. When "testpage5.html" and "testpage6.html" are both opened, the contents of each page appear in one individual tab pane (elements and attributes are formed in a table list in the tab pane); each tab pane has a corresponding title and a "close icon" and these two tab panes overlap and can be switched over;</p>
3	<p>1. Add a new text element "new text" with attributes to "testpage5.html" from its source code in the CentrePane;</p> <p>2. Modify an existing hyperlink element in "testpage5.html" by changing the name and "href" attribute value in its source code in the CentrePane;</p> <p>3. Delete an existing image element in "testpage6.html" through its source code in the CentrePane</p>	<p>Manipulate XHTML page: conventional web-like page editing.</p>	<p>1. In "testpage5.html" tab pane, the screen updates with "new text" elements appearing under the default layout in the upper frame;</p> <p>2. In "testpage5.html" tab pane, the screen updates with the target hyperlink's name modified in the upper frame; When loading the page with Internet Explorer, the hyperlink's name is different from the original version and leads to a different link page from the original version when clicking on the hyperlink;</p> <p>3. In "testpage6.html" tab pane, the screen updates with the target image object disappearing in the upper frame.</p>
4	<p>1. Switch the CentrePane to "list mode" by pressing the "element mode" button on the hotkey bar of the MainFrame;</p> <p>2. Select "Stopvideo.java" in the interactive application list in the RightPane; drag the selected application and move the mouse into the source code display frame of "testpage5.html" tab pane in the CentrePane; release the mouse left key to drop the application on Element 1 in the source code element list;</p> <p>3. Repeat Step 2 to add "PlayVideo.java" interactive</p>	<p>Advanced manipulation to XHTML service page: Semi-automatic configuration of interactive service application on UI components.</p>	<p>1. The source code display frame in the CentrePane changes from "text mode" that displays source code in plain text to "list mode" that displays source code in an element list format so as to be prepared for interactive application configuration;</p> <p>2. A dialogue window pops up and asks for inputting the relevant parameter; After the parameter is input, there is a new "id" attribute added to the Element 1 with a special interactive application id;</p> <p>3. Same process is repeated for Element 2;</p> <p>4. Same process is repeated for Element 3;</p> <p>5. Same process is repeated for</p>

	<p>application to Element 2 in “testpage5.html” source code (loop testing);</p> <p>4. Repeat Step 3 to add “Realtimedata.java” interactive application to Element 2 in “testpage5.html” source code (loop testing);</p> <p>5. Repeat Step 2 to add “voteclient.java” interactive application to Element 4, 5 in “testpage5.html” source code (loop testing);</p> <p>6. Repeat Step 2 to add “votereport.java” interactive application to Element 6 in “testpage5.html” source code (loop testing);</p> <p>7. Switch the CentrePane back to “text mode” by pressing “element mode” button on the hotkey bar of MainFrame again;</p>		<p>Element 4, 5;</p> <p>6. Same process is repeated for Element 6;</p> <p>7. The source code display frame in the CentrePane changes back from “list mode” to “text mode” for normal text editing; all the newly-added “id” attributes of the Element 1- 6 remain the same.</p>
5	<p>1. Press the “save” button on the hotkey bar in the MainFrame when “testpage5.html” tab pane is on top;</p> <p>2. Press the “save” button on the hotkey bar in the MainFrame when “testpage6.html” tab pane is on top;</p>	<p>Save any modifications to the XHTML service page file</p>	<p>1. All the modifications to “testpage5.html” service page are saved. Confirmation on saved files refers to testing case No.6 and 7;</p> <p>2. All the modifications to “testpage6.html” service page are saved. Confirmation on saved files refers to testing case No.6 and 7;</p>
6	<p>1. Open file “testpage5.html” using Windows Notepad and check the code according to the proposed XHTML and “ID Event” presentation method and “MDTV interactive service page creation requirement” in Table 5.5;</p> <p>2. Load “testpage5.html” with the proposed terminal service platform and check if it can be rendered and if all the interactive applications work as well as the creation requirement;</p> <p>3. Open file “testpage6.html” with Windows Notepad and check the code according to the proposed XHTML and “ID Event” presentation method;</p> <p>4. Load “testpage6.html” with the proposed terminal</p>	<p>Output MDTV interactive service page based on the XHTML and “ID Event” presentation method.</p>	<p>1. The source code of “testpage5.html” has been modified according to the “MDTV interactive service page creation requirement” and the format of the source code matches XHTML and “ID Event” presentation method;</p> <p>2. Functionality testing to all the interactive applications refers to the testing procedure of the proposed terminal service platform in the next section;</p> <p>3. The source code of “testpage6.html” matches XHTML and “ID Event” presentation method;</p> <p>4. Functionality testing on “testpage6.html” service page refers to the testing procedure of the proposed terminal service platform in the next section;</p>

	service platform and check if it can be rendered.		
7	<ol style="list-style-type: none"> 1. Search for a file named “IDEventFactory.java” in the pre-defined output folder; 2. Open the file with Windows Notepad and check the source code according to the XHTML and “ID Event” presentation method and “MDTV interactive service page creation requirement” shown in Table 5.5 3. Replace the old version of “IDEventFactory.java” in the proposed terminal service platform project folder with this new one; 4. Run the service platform in the default Java ME emulator to check if the platform starts to support all the newly added interactive applications (associating it with “testpage5.html”). 	Semi-automatically output the reconfigured functional class of the terminal service platform based on the XHTML and “ID Event” presentation method that supports newly added interactive applications.	<ol style="list-style-type: none"> 1. By completing testing case No. 5, the new version of “IDEventFactory.java”, which is the database of all the supporting interactive applications in the terminal service platform, has been generated automatically; 2. The source code of “IDEventFactory.java” matches the XHTML and “ID Event” presentation method and “MDTV interactive service page creation requirement”; 3. Functionality testing to the revised terminal service platform refers to the testing procedure in the next section.

Table 5.6: System testing case design by using White-box method

GUI testing

Since the purpose of the software’s GUI is to assist the user to achieve the functionality of the software, and in order to define the event coverage of the GUI testing, which requires all reachable events in a GUI to execute at least once during a complete cycle of the test case, we thus set the GUI testing events by choosing the events that achieve the functionalities of the software. Therefore we have chosen white-box testing cases as the GUI testing cases and the record-playback technique (as explained in Chapter 2) is implemented as follow:

GUI Group	GUI Component	Event Record (Expected Output)
Menu	“Open XHTML File...” option	When selecting “File – Open XHTML File...” a pop-up window will appear in the top middle within the MainFrame; The tester can navigate in the file system within the window; when selecting the target file in the window, all the relevant content will appear in the CentrePane and SouthPane.
	“Save XHTML File...” option	All the modifications to the target service page are saved.
Hotkeys	“Save XHTML File” hotkey	All the modifications to the target service page are saved.
	“Element Mode” hotkey	The source code display frame in the CentrePane changes from “text mode” that displays source code in plain text to “list

		mode” that displays source code in the form of an element list so as to be prepared for interactive application configuration; The tester can press it again to change the mode back to “text mode”.
LeftPane	“Select and open file” operation	The tester can perform conventional navigation (up, down, select, go into deep folder and etc.) in the project file list within the LeftPane; When the left button of the mouse is double clicked on “testpage6.html” in the project file list, all the relevant content will appear in the CentrePane and SouthPane.
RightPane	“Select and drag” operation	The tester can perform conventional navigation (up, down, select, go into deep folder and etc.) in the interactive application list within RightPane; When the tester selects a application by pressing the mouse left button and then holds it while he is trying to move the mouse pointer from RightPane to CentrePane source code display frame, the whole process is allowed to be performed by the software without any error alert.
CentrePane	“Drop to the element” operation	By following the “select and drag” option from the RightPane, the tester can release the mouse left button when the mouse pointer is moved to the target element in the source code frame under the “element mode”.
	“Source code” manipulations	Plain editing modes (add and delete) are allowed in the source code frame of the CentrePane when it is under the “text mode”.
	“Scroll the editing area” button	The tester can press the “up arrow” to minimize the page display frame while maximizing the source code display frame within the CentrePane; The tester can press the “down arrow” to minimize the source code display frame while maximizing the page display frame within the CentrePane.
SouthPane	“Data update” manipulation	The tab switches together with the corresponding tab in CentrePane; Parameters update immediately if there are any modifications being done to the service page displayed in the corresponding tab in the CentrePane.
MainFrame	“Scroll the sizes of different pane area” manipulations	All the edges between the different panes in the MainFrame are drag-able to make other panes more visible.

Table 5.7: Event records of the GUI

Testing cases	GUI testing targets (playback targets)
Case No.1	“Scroll the size of different pane areas” of the MainFrame
	“Open the XHTML File...” option of the Menu
	“Select and drag” operation of the LeftPane
Case No.2	“Data update” manipulation of the SouthPane
	“Scroll the editing area” button of the CentrePane
Case No.3	“Source code” manipulations of the CentrePane
Case No.4	“Element Mode” hotkey of Hotkeys
	“Select and drag” operation of the RightPane
	“Drop to the element” operation of the CentrePane
Case No.5	“Save XHTML File” hotkey of Hotkeys
	“Save XHTML File...” option of the Menu
Case No.6	Out of the scope of GUI testing
Case No.7	Out of the scope of GUI testing

Table 5.8: Event playbacks of GUI testing (Case No. refers to Table 5.6)

5.2.3 MDTV SERVICE TERMINAL SOFTWARE PLATFORM TESTING

The aims of the testing to the terminal device service platform are:

- To evaluate if the target software is able to work properly as a integrated software without major errors;
- To evaluate if the target software is able to achieve its functionality by rendering the MDTV interactive service page based on XHTML and “ID Event” presentation method;
- To evaluate if the target software is able to achieve its functionality by handling the interactions between the user and the interactive applications in the service page.

To implement these aims, the test cases along with the testing data are then designed for different testing levels and different testing methods.

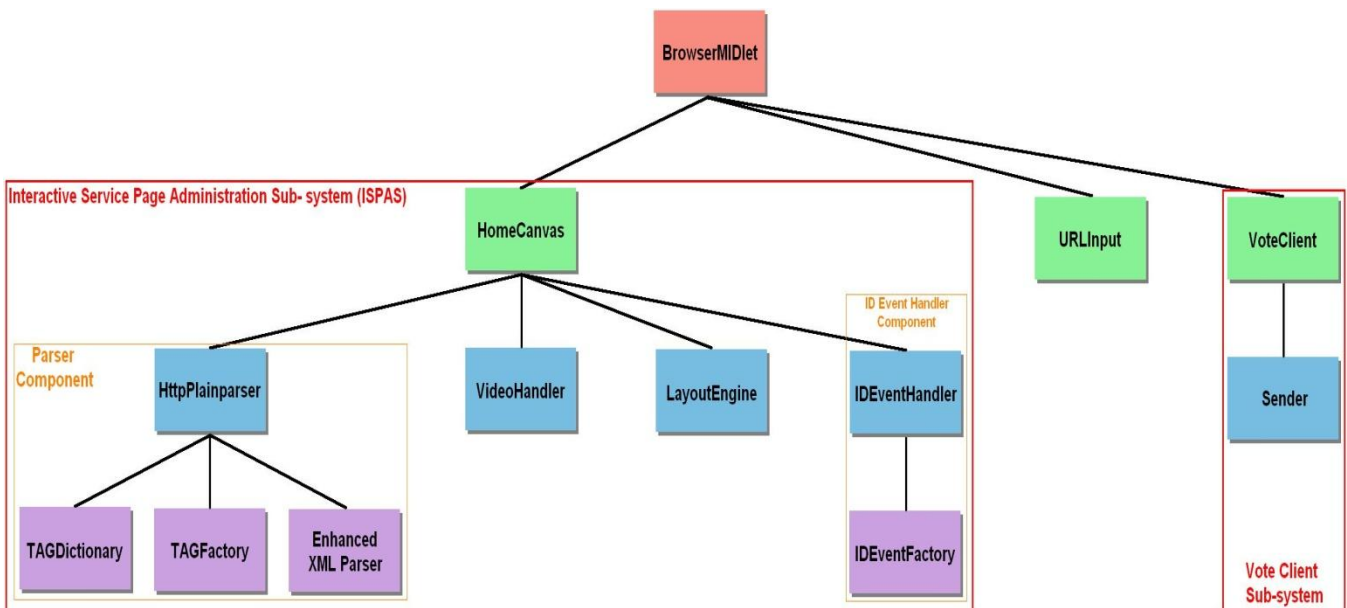


Figure 5.15: Class components of the MDTV service terminal software platform

Testing data

Here are all the service pages (as shown in Section 5.2.2 on page 140) for which will be used during the testing cases on different testing level:

Figure 5.2: Testpage1.html: pure plain text page.

Figure 5.3: Testpage2.html: pure image page.

Figure 5.4: Testpage3.html: pure hyperlink page.

Figure 5.5: Testpage4.html: mixed components page.

Figure 5.6: Testpage5.html: test page for testing interactive applications.

Figure 5.7: Testpage6.html: service homepage

Figure 5.8: Testpage61.html: service portal.

Figure 5.9: Testpage7.html: multimedia service page with video

Figure 5.10: Testpage8.html: program guide

“Testpage5.html” is a MDTV interactive service page with interactive application added by using the proposed semi-automatic service creation tool. Also, the IDEventFactory.java of the terminal platform is also the output of the creation tool, along with the “Testpage5.html”.

Component testing

As the functional requirement of the terminal service platform is to deliver interactive multimedia MDTV services to the end user, two aspects of design requirements are set. These are the rendering and retrieval of interactive multimedia MDTV service page, and the handling of the interactivities between the MDTV services (locally and remotely) and the end user. The thirteen classes that compose the proposed terminal service platform are listed in Table 5.9 further with their corresponding design requirement descriptions. Since the component testing work on these classes has also been done while they were being programmed, it can be concluded that the design requirement specification (the details refer to relevant section in Chapter 4) of the proposed software system has been met as expected.

Class Components	Design Requirement
BrowserMIDlet	MIDlet class for initializing the terminal service platform.

URLInput	GUI component and input interface that accepts MDTV service local/remote URI inputted by end user and calls HomeCanvas for actual implementation.
VoteClient	Client side for remote interactive applications (such as voting) that is in charge of setting up a connection with the remote interactive service server.
Sender	Assistant class to VoteClient; it is employed for sending interactive data from the client side to the server side to help achieve the remote interactions.
HomeCanvas	Functional class that administrates main processes within the service platform including GUI navigation, service page drawing and event detecting.
HttpPlainparser	Primary service page parser to receive and parse the XHTML and "ID Event" based service page.
TAGDictionary	Assists HttpPlainparser by providing a tag dictionary.
TAGFactory	Assists HttpPlainparser by providing tag sorting methods
EnhancedXMLParser	Assists HttpPlainparser by providing an element sorting algorithm.
VideoHandler	Assists HomeCanvas by handling operations concerned with video media.
LayoutEngine	Assists HomeCanvas by providing a page layout processing mechanism (supports CSS for further work).
IDeventHandler	Assists HomeCanvas by handling all the events on the MDTV interactive service page based on "ID Event" method.
IDEventFactory	Assists IDeventHandler by providing ID events for implementing methods.

Table 5.9: Design requirement specification for the MDTV interactive service terminal software platform

Integration testing

The integration testing of the proposed service platform also utilises a "bottom-up integration" method as illustrated in Figure 5.15. In the first integration loop, all the lowest functional classes were integrated to the related upper functional classes to form the functional components. These are the parsing component, Media Handler, Layout Engine and ID Event Handler. In the second integration loop, the functional components integrate with the upper functional classes to form several sub-systems. These are the interactive service page administration sub- system (ISPAS), the URLInput and Vote Client. All these sub-system assist the BrowserMIDlet to complete the service terminal platform. An interface testing (shown in Figure 5.16) is designed for the final integration loop to ensure that all the sub-systems are interfacing properly within the system. The test case design is listed in Table 5.10:

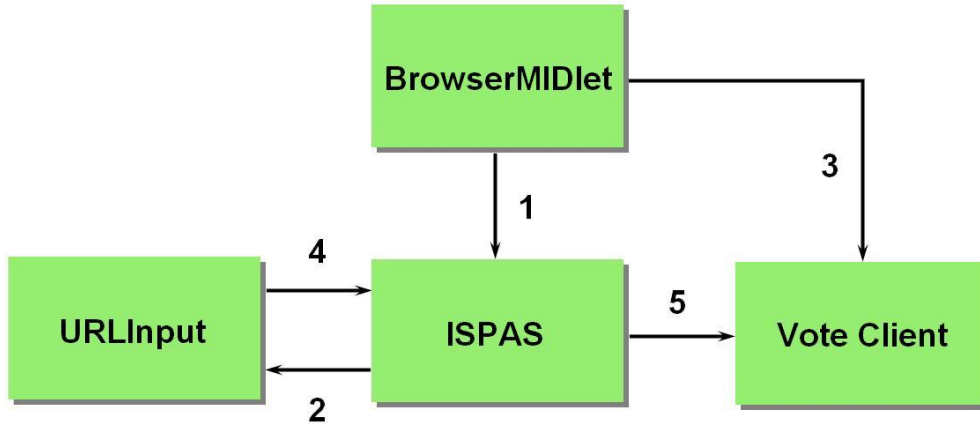


Figure 5.16: Interface testing logic between the different sub systems

Interface testing cases	Inputs	Expected output
Interface 1: BrowserMIDlet - ISPAS	Start up the main MIDlet “BrowserMIDlet.java” in Java ME WTK emulator through Eclipse IDE.	The HomeCanvas, consists of a “Menu” button at the right-bottom corner and a blank background, is shown on top of the screen and is set on focus;
Interface 2: ISPAS - URLInput	Press “Menu” button on the emulator’s keyboard, select “URLInput” option and press the “OK” button on the keyboard.	The screen refreshes and the URLInput is set on focus and on top of the screen; The URLInput page has a text field available for URI input.
Interface 3: BrowserIMDlet – Vote Client	1. Starts up the demo “interactive service server”; 2. Execute the same input in the Interface 1 testing case.	1. An alert message page pops up and asks for the confirmation of accessing the interaction network; 2. When “Yes” is selected, the server console detects and affirms the connection; the terminal platform initializes successfully with a data exchange port connected to the interaction server; 3. When “No”, is selected another alert message page pops up showing the error message “client run error”; the terminal platform initializes successfully without a remote connection to the interactive service server.
Interface 4: URLInput - ISPAS	1. Press the “Back” button at the left bottom corner of URLInput page; 2. Go to the URLInput page; 3. Type “http://localhost/testpage5.html” in the test field of URLInput page; Press the “Menu” button at the right bottom corner, select the “URLget” and press the “OK” button on the keyboard. Repeat step 2.	1. The screen refreshes and the HomeCanvas is set on focus and on top of screen; 2. A pop up window comes up confirming the use of the interaction network if this is the first time of trying to access a service page remotely; When pressing “Yes”, the screen refreshes and the “testpage5.html” is displayed on the screen with full features and proper layout that fits

		the screen size; When pressing “No”, the screen refreshes and a blank HomeCanvas is set on top of screen available for operation; No more further network confirmation notice pops up again.
Interface 5: ISPAS – Vote Client	1. Repeat the Interface 3 testing case input; 2. Repeat step 2 in the Interface 4 testing case; 3. Press “vote for good” button on the service page	1. The terminal platform is initialized successfully; 2. “testpage5.html” is loaded successfully; 3. The vote input is sent to the interactive service server and recorded in a database on the server side; the validation of this operation refers to a related testing case in White-box system testing section.

Table 5.10: Testing case design on Integration testing level

System testing

a) Black-box testing

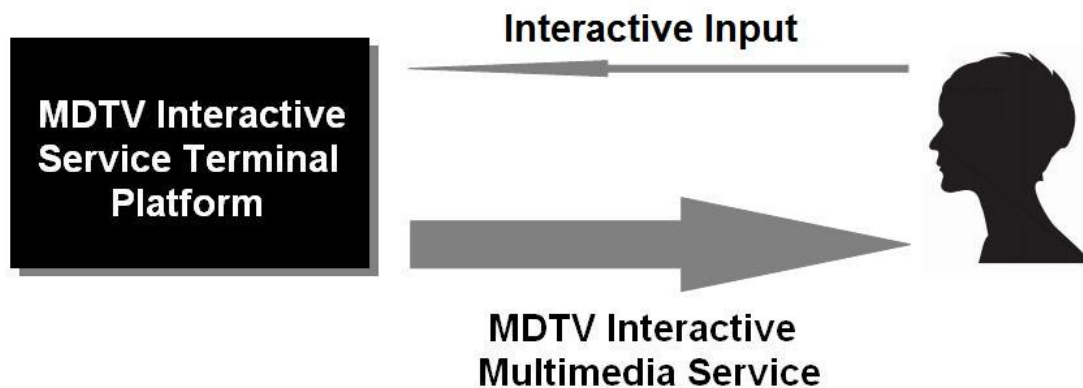


Figure 5.17: Black-box testing

We have also applied Equivalence Partitioning here as the black-box testing method. Here, all the inputs are handled through the GUI of the interactive service terminal platform. There are three main types of inputs: the Page Link Input, the Interactive Event Input and the Navigation Input. Thus the input partitioning definitions are designed as listed in Table 5.11.

Case No.	Equivalence partitioning input		Expected output
1	Page Link Input	Local	Testpage1.html
2			Testpage2.html
3			Testpage3.html
			Testpage1 loads successfully and is available for further operation.
			Testpage2 loads successfully and is available for further operation
			Testpage3 loads successfully and is available

			for further operation
4	Remote	Testpage5.html	Testpage5 loads successfully and is available for further operation
5		Testpage61.html	Testpage61 loads successfully and is available for further operation
6	Interactive Event Input	“Play video” application	A pre-ordered video appears playing on the service page with a proper layout that fits the screen size;
7		“Stop video” application	The video stops playing;
8		“Request for real time data” application	A ticker appears on the top position of screen with live data refreshing every 10 seconds;
9		“Vote for good” application	An alert message pops up on the screen to inform the data has been received; The message page disappears after 2 seconds and the scene goes back to the previous page. validation of this operation refers to next input case;
10		“Request for vote result” application	An alert message pops up on the screen to inform the request has been received; The message page disappears after 2 seconds and screen refreshes, then a “vote result” service page appears on top of screen with the vote record received from the last input case;
11	Navigation Input	Press “up” on the keyboard	The green pointing rectangle moves to the upper next active component; if the rectangle was already on the top first active component, the rectangle jumps to the bottom last active component;
12		Press “Down” on the keyboard	The pointing rectangle moves to the lower next active component; if the rectangle was already on the bottom last active component, the rectangle jumps to the top first active component;
13		Press “Left” on the keyboard	The pointing rectangle moves to the left next active component; if the rectangle was already on the top first active component, the rectangle jumps to the bottom last active component; if the rectangle was already on the left-most active component in a line, the rectangle will jump to the right-most active component on the upper line;
14		Press “Right” on the keyboard	The pointing rectangle moves to the right next active component; if the rectangle was already on the bottom last active component, the rectangle jumps to the top first active component; if the rectangle was already on the right-most active component in a line, the rectangle will jump to the left-most active component on the lower line;
15		Select and press “OK” on Hyperlink 1 on “testpage7.html”	The screen refreshes and a new service page that the hyperlink 1 is linked to is displayed on top of screen and is available for further operation.

Table 5.11: System testing case by using Black-box method

b) White-box testing

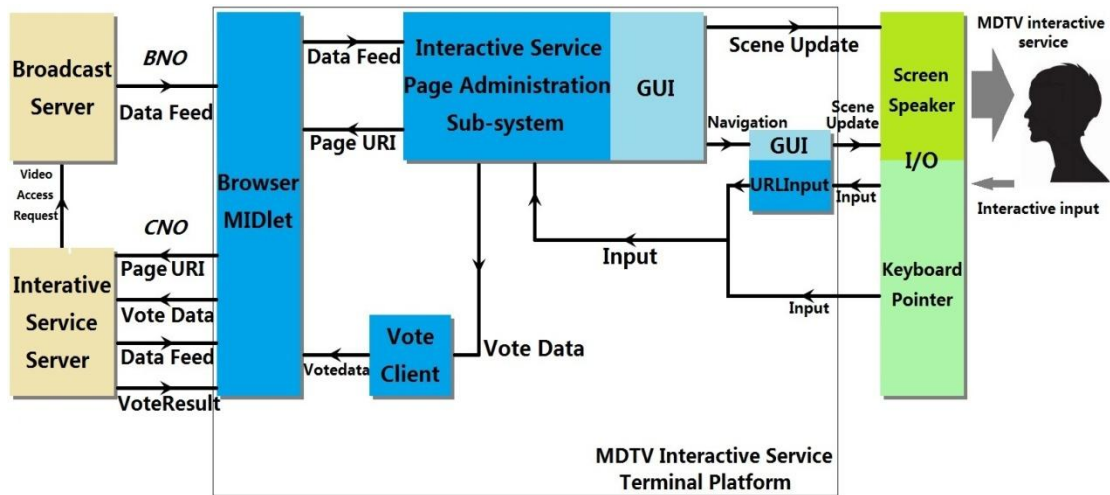


Figure 5.18: White-box testing

In white-box testing, we choose path testing and loop testing as the case design methods. As mentioned at the beginning of this section, the two aspects of the functional requirement of the terminal service platform are the rendering and retrieval of interactive multimedia MDTV service pages, and the handling of the interactivities between the MDTV services (locally and remotely) and the end user. Therefore we designed the path testing cases according to the implementation logic of these two functions. The user requirements are defined for testing purposes in Table 5.12 and the testing case design is included in Table 5.13. Here we choose several test page sets including the “testpage5.html”, which is an interactive application test page linked to the MDTV interactive service sample home page “testpage6.html” and “testpage7.html” which is a multimedia service page with video also linked to the home page as the test data. All of these pages are MDTV interactive service pages outputted from our proposed semi-automatic service creation tool that follows the XHTML and “ID Event” presentation method.

Logic path	User requirement
Logic path 1: MDTV interactive service page rendering	To be able to render interactive service pages from a local directory.
	To be able to render interactive service pages from a remote directory.
Logic path 2: Interactivity between the MDTV service and the end user	To be able to support conventional service page browsing (navigation on a page or between different pages).
	To be able to handle any interactive service applications that are based on the proposed XHTML and “ID Event” presentation method.

Table 5.12: User requirement to MDTV interactive service terminal platform

Case No.	Testing Cases	User Requirement	Expected Output
Logic Path 1 possibility 1: BrowserMIDlet – ISPAS – URLInput – ISPAS – BrowserMIDlet - ISPAS			
1	<p>1. Prepare the testing service page set by copying these pages to the local directory: "C:\WTK25\appdb\MediaControlSkin\filesystem\root1";</p> <p>2. Start up the Browser MIDlet;</p> <p>3. select the "URLInput" option in the menu on the right bottom of the screen;</p> <p>4. Type "file:///root1/local/testpage6.html" into the text field and then select the "URLget" option in the menu and press the "ok" button on the keyboard.</p>	To be able to render the interactive service page from a local directory.	<p>1. This test case happens when the service pages are firstly received through the broadcast network/interaction network and are ready for end-user to access. Therefore, here we assume that all the service pages are already in the local storage unit and the default data storage directory of the service platform is as mentioned in testing step 1.</p> <p>2. A demo sever application is started up and waits for connection from the service platform; The WTK emulator appears on the screen and the Browser MIDlet starts running in the emulator and connects to the demo server automatically; the service platform is initialized with a blank background and with two menu options at the emulator screen bottom.</p> <p>3. A sub menu list shows up on the screen when selecting the menu option; Navigate up and down to find the "URLInput" option and the emulator scene refreshes by putting the URLInput window on top and the components on the URLInput window become available for further operation.</p> <p>4. The URL appears in the text field while it is being input through the keyboard of the emulator; When selecting the "URLget" option in the menu, the emulator screen refreshes and the target page "testpage6.html" is displayed on top of the screen with all the components (text, image, hyperlink and interactive applications) displayed in a proper layout that fits to the emulator screen size; The active elements on the page are available for further operations.</p>
Logic Path 1 possibility 2: BrowserMIDlet – ISPAS - BrowserMIDlet – ISPAS			
2	<p>1. Prepare testing service page set by copying these pages to local directory: "C:\Program Files\Apache Software Foundation\Apache2.2\htdocs";</p> <p>2. Start up the demo server; Start up Browser MIDlet;</p> <p>3. Select the "homepage" option in the menu on the right bottom of the emulator screen. This "homepage" option is a predefined home page</p>	To be able to render the interactive service page from a remote directory.	<p>1. This test case occurs when the service pages needs to be requested through the interaction network. Therefore, here we set up a demo interaction server by using Apache 2.2 and utilize the Apache's default root path as the remote storage directory as mentioned in testing step 1;</p> <p>2. Same output as in step 2 in case No.1;</p> <p>3. The emulator screen refreshes and the target page "testpage6.html" is displayed on top of the screen with all the components (text, image, hyperlink and interactive applications) are displayed in a proper layout that fits to the emulator screen size; the active elements on the page are available for further operations.</p>

	<p>short cut in our proposed service platform. By selecting this option, a remote URL “http://server IP/remote/testpage6.html” of the home page is sent to the ISPAS.</p>		
<p>Logic Path 2 possibility 1: BrowserMIDlet – ISPAS - BrowserMIDlet – ISPAS – BrowserMIDlet – ISPAS</p>			
<p>3</p>	<ol style="list-style-type: none"> 1. Start the BrowserMIDlet and the demo server; Select “URLInput” option in the menu on the right bottom of screen; Type “http://server IP/remote/testpage6.html” into the text field and then select “URLget” option in the menu and press “ok” button on the keyboard; 2. Press the “ok” button on the page; 3. Press the “up” navigation buttons on the emulator keyboard ten times; 4. Press the “down” navigation buttons on the emulator keyboard ten times; 5. Press the “left” navigation buttons on the emulator keyboard ten times; 6. Press the “right” navigation buttons on the emulator keyboard ten times; 7. Navigate to the Hyperlink No.1 and press the “ok” button; 8. if a new page displayed on top of the screen, navigate to any hyperlinks on the service page and press “ok”; 9. Repeat steps 7, 8 for three times by selecting different hyperlinks on the service pages. 	<p>To be able to support conventional service page browsing (navigation on a page or between different pages).</p>	<ol style="list-style-type: none"> 1. Service platform is started and connected to the demo server; After entering the URL and pressing “URLget” in the menu, the screen refreshes and the target page “testpage6.html” is displayed on top of the screen with all the components (text, image, hyperlink and interactive applications) displayed in a proper layout that fits to the emulator screen size; 2. The screen refreshes and the “testpage61.html” is displayed on top of the screen with all the components in a proper layout; 3. The green pointing rectangle moves to the upper next active component; if the rectangle was already on the top first active component, the rectangle jumps to the bottom last active component; 4. The pointing rectangle moves to the lower next active component; if the rectangle was already on the bottom last active component, the rectangle jumps to the top first active component; 5. The pointing rectangle moves to the left next active component; if the rectangle was already on the top first active component, the rectangle jumps to the bottom last active component; if the rectangle was already on the left-most active component in a line, the rectangle will jump to the right-most active component on the upper line; 6. The pointing rectangle moves to the right next active component; if the rectangle was already on the bottom last active component, the rectangle jumps to the top first active component; if the rectangle was already on the right-most active component in a line, the rectangle will jump to the left-most active component on the lower line; 7. The emulator screen refreshes and the target page “testpage6.html” is displayed on top of screen with all the components (text, image, hyperlink and interactive applications) displayed in a proper layout that fits to the emulator screen size; 8. The screen refreshes and either “testpage6.html” or “testpage7.html” is displayed; 9. The screen refreshes when different hyperlinks are selected and the service pages related to the hyperlinks are displayed.

Logic Path 2 possibility 2: BrowserMIDlet – ISPAS – BrowserMIDlet – ISPAS – VoteClient - BrowserMIDlet – ISPAS			
4	<ol style="list-style-type: none"> 1. Start the BrowserMIDlet and the demo server; Select “URLInput” option in the menu on the right bottom of screen; Type “http://server IP/remote/testpage5.html” into the text field and then select “URLget” option in the menu and press “ok” button on the keyboard; 2. if the page is displayed properly, press Button 2 to play video; 3. Press Button 1 to stop playing video; 4. Press Button 3 to request for real time data; 5. Press Button 4 twice to vote “good”; 6. Press Button 5 once to vote “bad”; 7. Press Button 6 to request for the vote result. 	<p>To be able to handle any interactive service applications that is based on the proposed XHTML and “ID Event” presentation method.</p>	<ol style="list-style-type: none"> 1. The service platform is started and connected to demo server; After entering the URL and pressing the “URLget” in the menu, the emulator screen refreshes and the target page “testpage5.html” is displayed on top of the screen with all the components (text, image, hyperlink and interactive applications) displayed in a proper layout that fits to the emulator screen size; 2. A pre-ordered video appears playing on the service page with a proper layout that fits the screen size; 3. The video stops playing; 4. A ticker appears on the top position of screen with live data refreshing every 10 seconds; 5. An alert message pops up on the screen every time the button is pressed to inform the request has been received; The message page disappears after 2 seconds; the screen refreshes and the scene goes back to the “testpage5.html”; the validation of this operation refers to the expected output of step 7; 6. An alert message pops up on the screen every time the button is pressed to inform the request has been received; The message page disappears after 2 seconds; the screen refreshes and the scene goes back to the “testpage5.html”; the validation of this operation refers to the expected output of step 7; 7. The screen refreshes and a “vote result” service page appears on top of the screen with the vote result (good: 2, bad: 1) received from the last two input cases;

Table 5.13: System testing case design by using White-box method

GUI testing

Here we have designed the GUI test case according to the white-box testing case presented in Table 5.13 and the record-playback technique has been implemented as shown in Table 5.14 and 6.15:

GUI Group	GUI Component	Event record (Expecting output)
MIDlet Menu	“Back” command	The tester can execute this command to go back to other optional windows (such as the URLInput) to the main platform window (HomeCanvas).
	“Exit” command	The tester can terminate the platform program and shut down the emulator by executing this menu command.
HomeCanvas Menu	“URLInput” command	Execute it to go to the “URLInput” window for entering the service page URL.
	“Homepage” command	Execute it to load the default service page.
	“Prepage” command	Execute it to go to the previous service page.
	“ESG Viewer” command	Execute it to start viewing the ESG and perform further operations through the ESG.

HomeCanvas	“General web-like navigation” operation	Allows the user to browse the service page visually (see the page components including text, image, hyperlink, and video) and interact with the page (navigate between hyperlinks on the page and select these to go to the target page).
	“Interactive application” operation	Allows the user to navigate between the interactive service components within the service page and select them to execute the corresponding interactive applications.
URLInput	“URLget” menu command	Execute it to load the service page corresponding to the URL entered in the text field of the URLInput window
	“Text field” manipulation	Allows the user to enter any service page URL through the keyboard.
Alert messages	“Confirmation for server connection” message	Select “yes” to allow server connection, select “no” to refuse.

Table 5.14: Event record of GUI components

Testing Case		GUI testing target (playback targets)	
Case No.1	Step 1	Out of the scope of GUI testing	
	Step 2	Start the demo server and the BrowserMIDlet	
	Step 4	Step 3	The “URLInput” command of the HomeCanvas
			The “Text field” manipulation of the URLInput
			The “Back” command of the MIDlet Menu
			The “URLget” menu command of the URLInput
Case No.2		The “Homepage” command of the HomeCanvas Menu	
Case No.3		The “General web-like navigation” operation of the HomeCanvas	
		The “Prepage” command of the HomeCanvas Menu	
Case No.4		The “Interactive application” operation of the HomeCanvas	
		The “Exit” command of the MIDlet Menu	

Table 5.15: Event playbacks of the GUI testing (Case No. refers to Table 5.13)

Client/server testing

Considering that the server we created is only for assisting purposes, thus the Client/Server testing can be merged into the terminal platform white-box testing section. More precisely, the Client/Server mechanism can be tested during several testing cases that are related to the execution of the remote interactive applications (such as require for live data feed, voting, and require for vote result), where the data exchange between client side and server side is required.

5.2.4 INTEGRATION TESTING OF THE SERVICE CREATION TOOL AND THE

TERMINAL PLATFORM

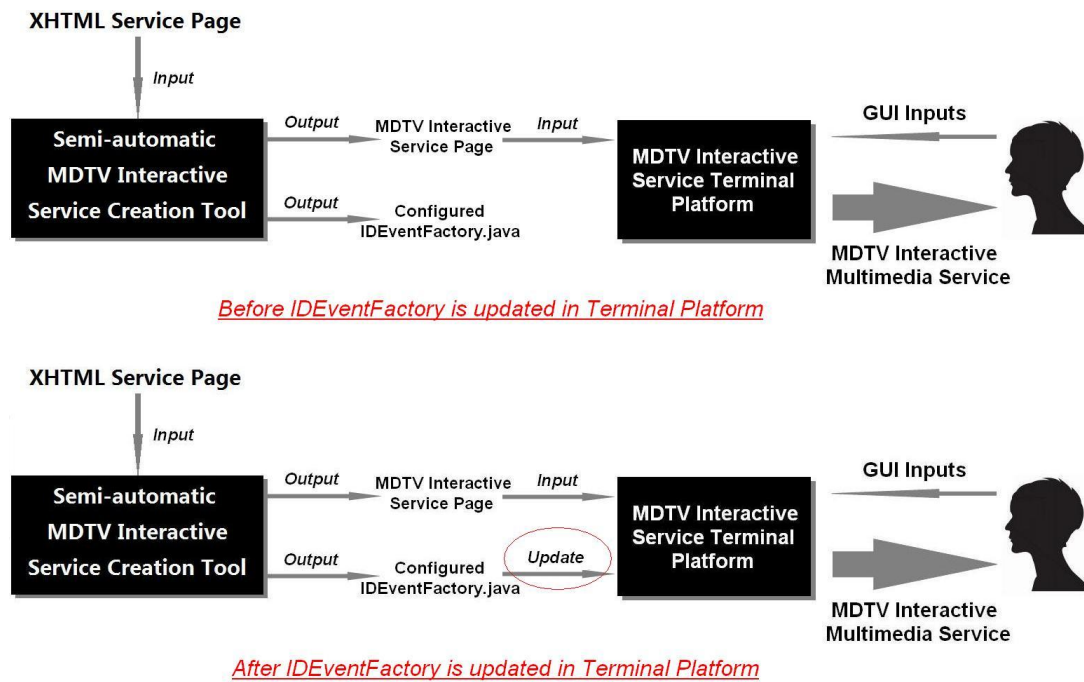


Figure 5.19: Interface testing and comparison testing

Based on the proposed solution requirement, we have developed two pieces of functional software during the MDTV service creation and consumption process; and two sets of test cases also have been designed separately so as to ensure their verification and validation. Moreover, when these two software components are integrated in the proposed MDTV service implementation system, there is a need of an Integration Testing procedure that would help to ensure they are able to work together properly. Regarding to the requirement of the proposed MDTV service creation and consumption system, the test aims can be further set as:

- To evaluate if the semi-automatic service creation and the terminal device service platform are able to work together properly within the proposed system without major error;
- To evaluate if these two software components can further enable the proposed MDTV service creation and consumption system (illustrated in Figure 2.23) and eventually achieve the aim of the Thesis.

So for the test case design, an interface testing between the semi-automatic service

creation tool and the service terminal platform will be conducted as part of the integration testing procedure. Besides a comparison testing will be applied as an assisting testing method between the terminal platform with the original IDEventFactory and terminal platform with the new IDEventFactory outputted from the service creation tool.

Here we take testpage6.html, testpage61.html, testpage7.html, testpage8.html as the testing pages. Before the testing cases begin, the default hyperlink relationships between these four pages are shown in Figure 5.20.



Figure 5.20: Hyperlink relationships

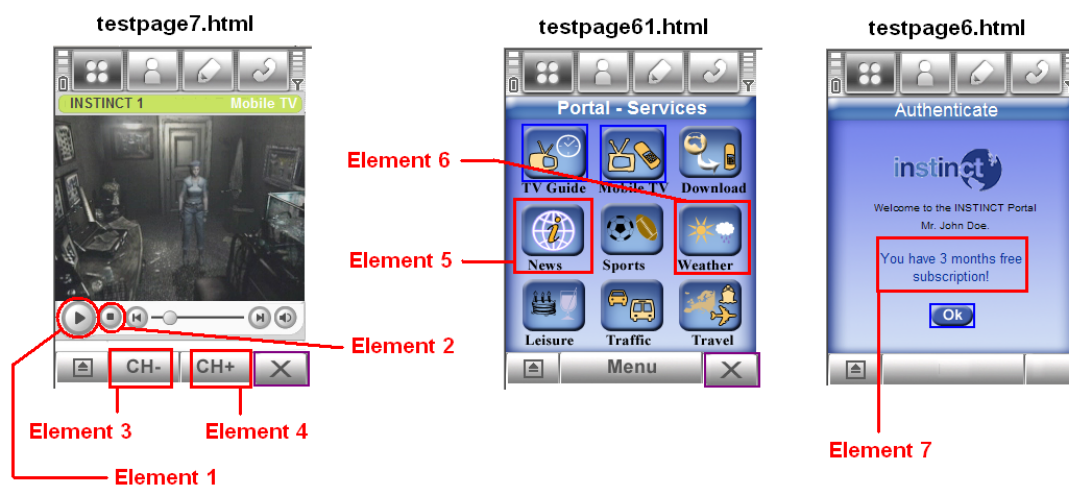
The testing case design is shown in Table 5.16:

Case No.	Testing Case Design	Expected Output
1	Prepare the testing input.	Select “testpage6.html”, “testpage61.html”, “testpage7.html”, and “testpage8.html” as the testing input.
2	Edit four testing service pages according to the design requirements presented in	All the requirements will be met and the four modified MDTV interactive service pages

	Figure 5.21 by using the proposed semi-automatic service creation tool.	(“testpage6.html”, “testpage61.html”, “testpage7.html”, and “testpage8.html”) based on XHTML and “ID Event” presentation method and a configured “IDEventFactory.java” will be outputted.
3	Set the interface input and prepare the terminal platform for testing.	Select four modified service pages as the interface input for the proposed terminal service platform; No change to the service platform will occur.
4	Set the modified “testpage6.html” as the input, repeat Case No.3 of the White-box testing case design illustrated in Table 5.13.	All the case steps will be passed without any errors.
5	Set the modified “testpage6.html” as the input; repeat Case No.4 of the White-box testing case design depicted in Table 5.13.	Case No.4 will fail: the terminal platform fails to handle any new interactive applications added to the page through the service creation tool.
6	Re-prepare the terminal platform and re-set the interface input	Update the terminal platform by replacing the original version of the “IDEventFactory.java” with the configured “IDEventFactory.java” outputted from Case No.2. Keeps the modified “testpage6.html” as the input.
7	Repeat Case No.5	All testing steps will be passed without any errors.

Table 5.16: Integration testing case design

The MDTV service creation requirements are shown in Figure 5.21:



Service creation requirement:

1. Add "play video" application on Element 1 on testpage7.html;
2. Add "stop video" applicaiton on Element 2 on testpage7.html;
3. Add "vote bad for the program" application on Element 3 on testpage7.html;
4. Add "vote good for the program" application on Element 4 on testpage7.html;
5. Add "request for vote report" applicaiton on Element 5 on testpage61.html;
6. Add "request for live data feed" application on Element 6 on testpage61.html;
7. Modify Element 7 on testpage6.html by change the text to "You have permanent free subscription";
8. No modification on testpage8.html.

Figure 5.21: Design requirement for the proposed service creation tool

5.3 TESTING EXECUTION OF THE SEMI-AUTOMATIC SERVICE CREATION TOOL

5.3.1 TEST PREPARATION

The test preparation information has already been presented in previous sections. For the testing environment refer to Table 5.1 and for the test data and test case design refer to the descriptions in Section 5.2.2.

5.3.2 TEST RESULT

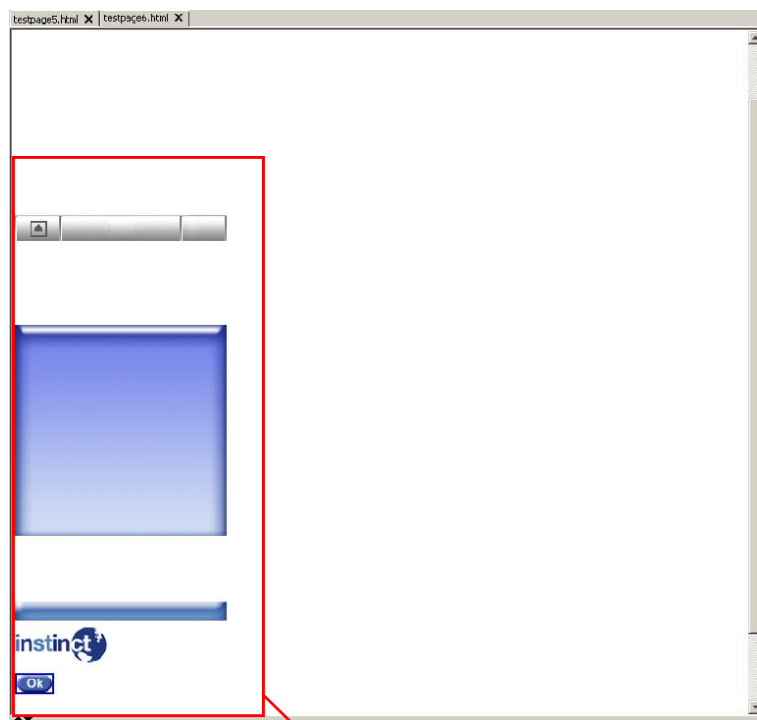
Integration Testing: The Interface testing cases have been successfully executed. Interface 3, 4, 5, 6, 7 have had the actual outputs matched with the corresponding expected outputs as listed in Table 5.3; Interface 1, 2 succeeded in getting the expected outputs when the testing cases were executed in a Windows XP environment. However when the testing cases were executed in a Windows 7 environment, Interface 1, 2 test case got an exception notice (error) through the Eclipse console window instead of the expected outputs. Overall, the software functional components of the target software that were tested, interface well with each other. The target software has thus passed the integration testing successfully except for the compatibility problem with the Windows 7 OS.

Black-box Testing: The equivalent partitioning input testing method has been successfully executed. All the inputs (including testpage1.html, testpage2.html, testpage3.html and testpage4.html) have had their actual result outputs matched to their corresponding expected outputs as listed in Table 5.4. More precisely, the output of each case includes an interactive MDTV service page and a reconfigured functional class: the interactive MDTV service page is based on the corresponding original XHTML page and presented in the proposed XHTML and ID Event presentation method; the reconfigured functional class is to update the proposed service platform

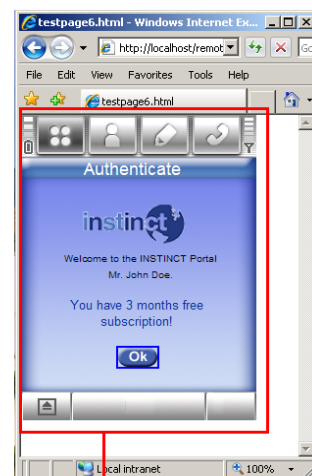
to support the newly-added interactive applications on the service page. Therefore, the target software has been able to pass the black-box testing successfully with out any errors.

White-box testing: All the testing cases (Case 1 to 7) have been successfully executed. All the test cases except Case 1 and Case 2 step 1 have got their actual outputs matched to their corresponding expected implementations as listed in Table 5.6. Through these cases, the proposed semi-automatic service creation tool is tested to be capable of assisting the tester achieving the MDTV service creation successfully. However the Two main errors that have occurred are:

1. When Case 1 was executed in a Windows 7 environment, there was an exception notice through the Eclipse console window instead of the expected output.
2. When Case 2 step 1 was executed, testpage5.html was displayed as expected but testpage6.html was displayed out of layout (refer to Figure 5.22). However this error was not a major problem that may cause crashing the software or the whole white-box testing process until completed.



testpage6.html viewed in the proposed service creation tool



testpage6.html viewed in Microsoft Internet Explorer 7

Figure 5.22: Failure in displaying testpage6.html in service creation tool

Besides, all the functionality validation in Case 6 and 7 will be completed in the testing procedure to the terminal platform presented in the next section. Overall, the target software has passed the white-box testing with two minor errors.

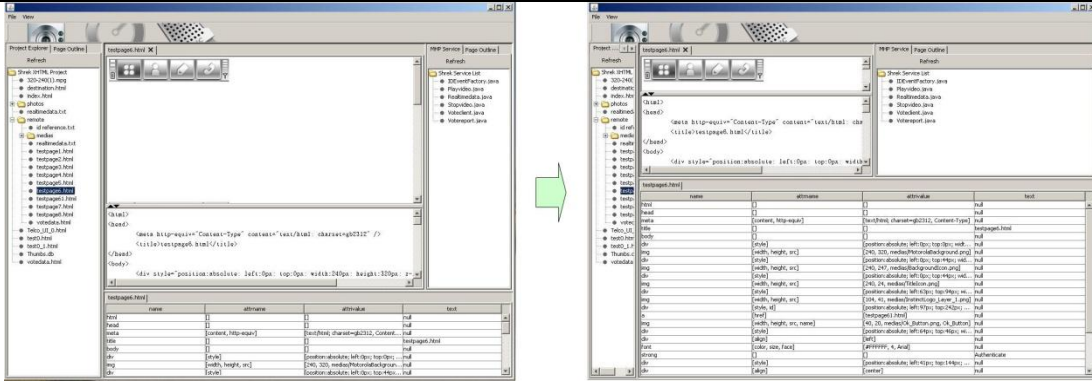
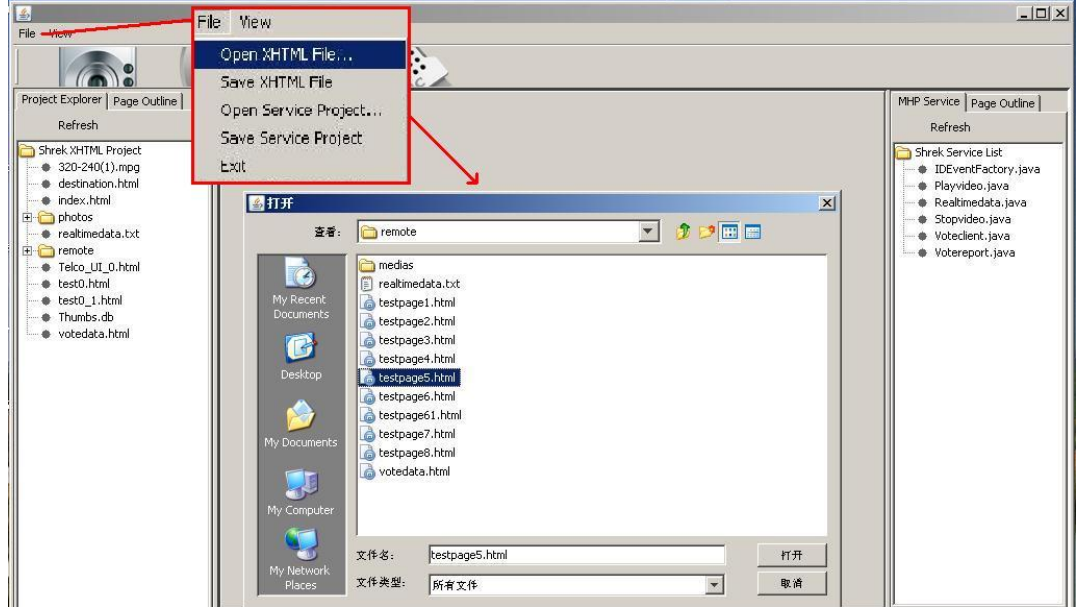
UI testing result	
Case No.	Target UI Component
	<p style="text-align: center;">“Manipulate the sizes of different pane areas” of the MainFrame</p>  <p style="text-align: center;">Figure 5.23</p>
	<p>Evaluation: Successful; the panes in the MainFrame can be easily resized.</p>
	<p style="text-align: center;">“Open the XHTML File...” option in the File Menu</p>
1	

Figure 5.24

Evaluation: Successful output; a dialog window pops up in the centre of the MainFrame area, ready for file selection and file opening.

“Select and open” operation of the LeftPane

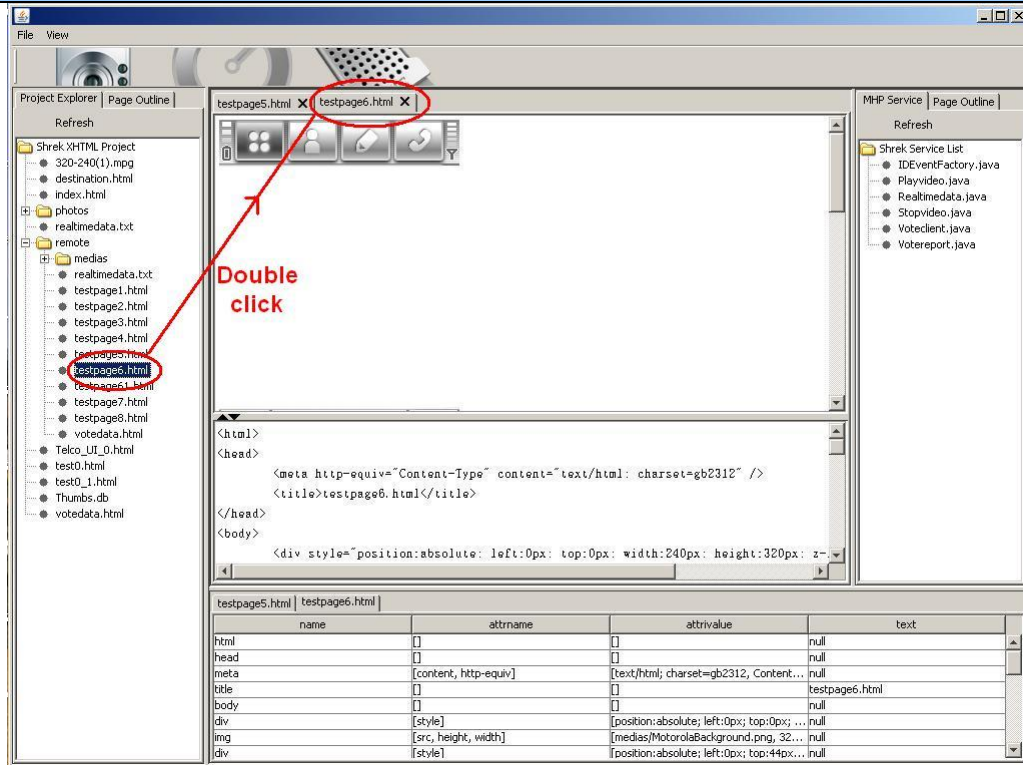
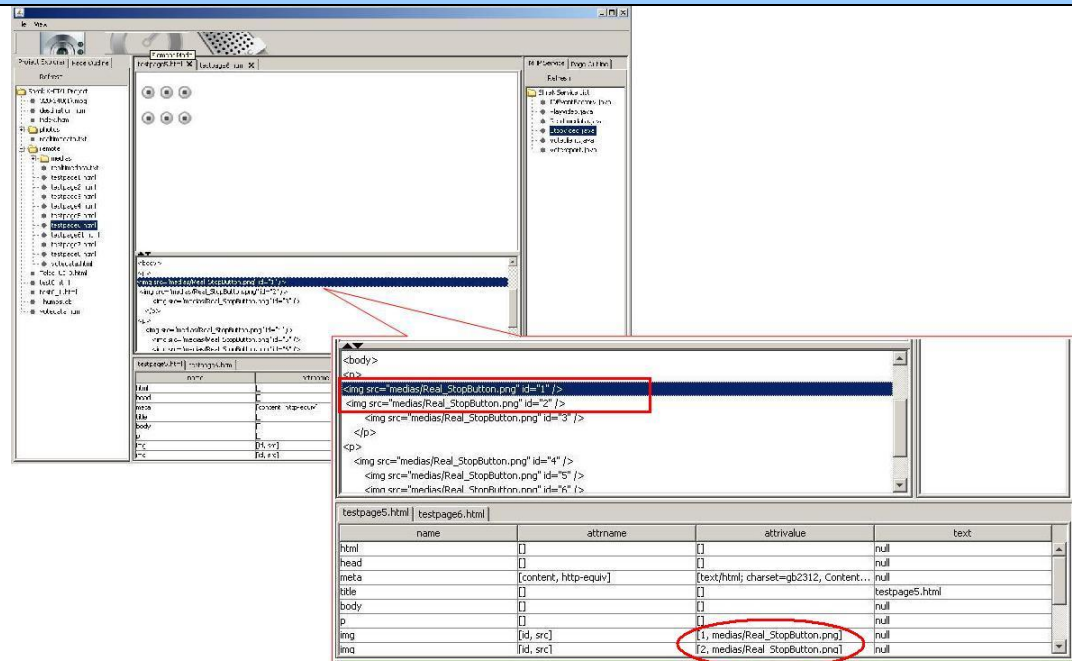


Figure 5.25

Evaluation: Successful output; a new tab appears in the CentrePane after selecting and double clicking an element name in the LeftPane file list.

“Data update” manipulation of the SouthPane



2

Figure 5.26: Before manipulation

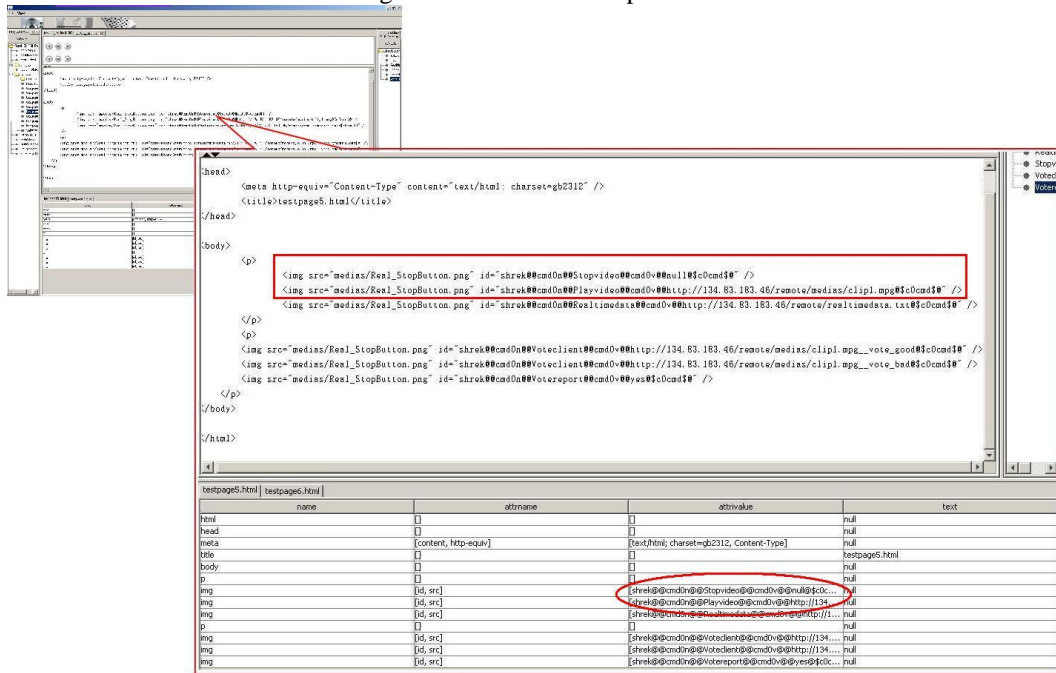


Figure 5.27: After manipulation

Evaluation: Successful output; the corresponding parameter updated together with the modifications in the CentrePane.

“Scroll the editing area” button of the CentrePane

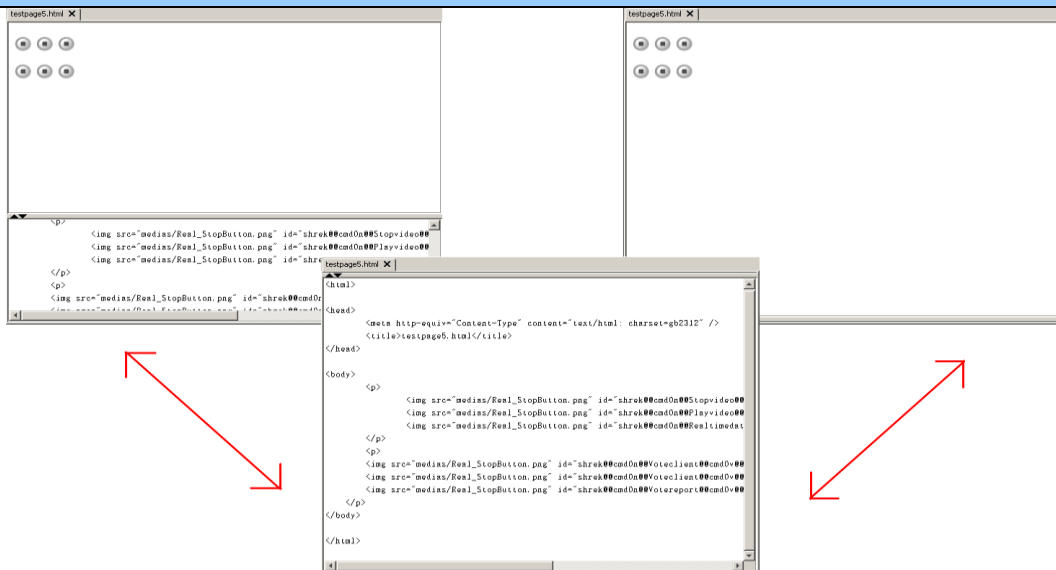


Figure 5.28

Evaluation: Successful output; the sizes of the upper frame and lower frame of the CentrePane is resizable

3

“Source code” manipulations of the CentrePane

testpage6.html X

instinct

Ok

Welcome to the INSTINCT Portal Mr. John Doe.

You have 3 months free subscription!

```


</div>
<div style="position:absolute; left:63px; top:94px; width:104px; height:41px; z-

</div>
<div id="Ok_Button" style="position:absolute; left:97px; top:242px; width:40px:
<a href="testpage61.html">

```

testpage6.html

name	attrname	attrvalue
img	[width, height, src]	[240, 24, medias/TitleIcon.png]
div	[style]	[position:absolute; left:0px; top:44px; width:240px; heig...
img	[width, height, src]	[240, 24, medias/TitleIcon.png]
div	[style]	[position:absolute; left:63px; top:94px; width:104px; hei...
img	[width, height, src]	[104, 41, medias/InstinctLogo_Layer_1.png]
div	[style, id]	[position:absolute; left:97px; top:242px; width:40px; hei...
a	[href]	[testpage61.html]



testpage6.html X

Ok

Welcome to the INSTINCT Portal Mr. John Doe.

You have 3 months free subscription!

```


</div>
<div style="position:absolute; left:63px; top:94px; width:104px; height:41px; z-
</div>
<div id="Ok_Button" style="position:absolute; left:97px; top:242px; width:40px:
<a href="testpage61.html">

```

testpage6.html

name	attrname	attrvalue
img	[width, height, src]	[240, 24, medias/BackgroundIcon.png]
div	[style]	[position:absolute; left:0px; top:44px; width:240px; heig...
img	[width, height, src]	[240, 24, medias/TitleIcon.png]
div	[style]	[position:absolute; left:63px; top:94px; width:104px; hei...
div	[style, id]	[position:absolute; left:97px; top:242px; width:40px; hei...
a	[href]	[testpage61.html]
img	[name, width, height, src]	[Ok_Button, 40, 20, medias/Ok_Butto...

Figure 5.29

Evaluation: Successful output; when the source code is verified, both the upper frame in the CentrePane and the parameter list in the SouthPane are updated together with it.

4 "Element Mode" hotkey of the Hotkeys

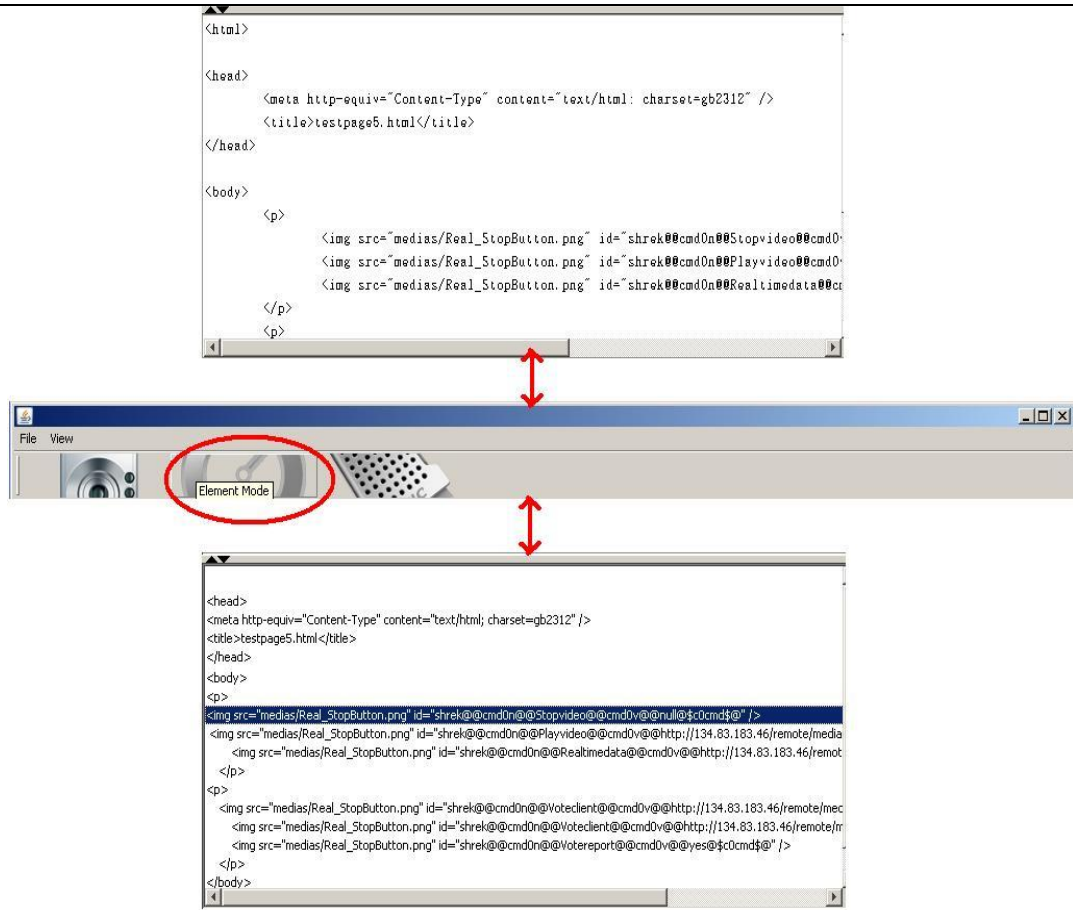


Figure 5.30

Evaluation: Successful output; by pressing the “element mode” key in the hotkey bar, the source code frame in the CentrePane switches between plain text display mode for regular text editing and list mode for interactive application editing. The source code remains the same during the switch.

“Select and drag” operation in the RightPane; “Drop of an element” operation in the CentrePane

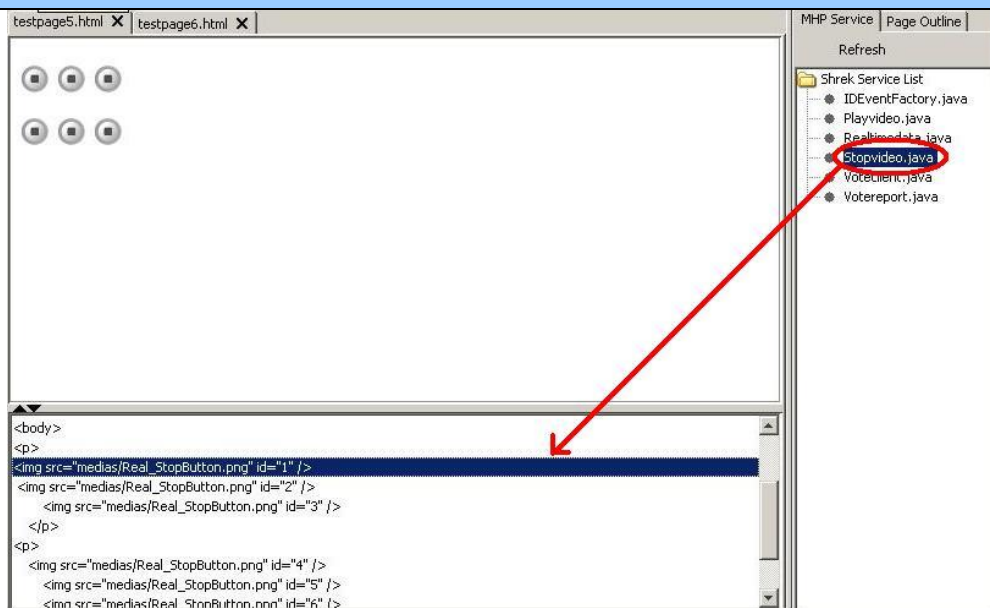


Figure 5.31: Drag the application from RightPane to CentrePane and drop on target element

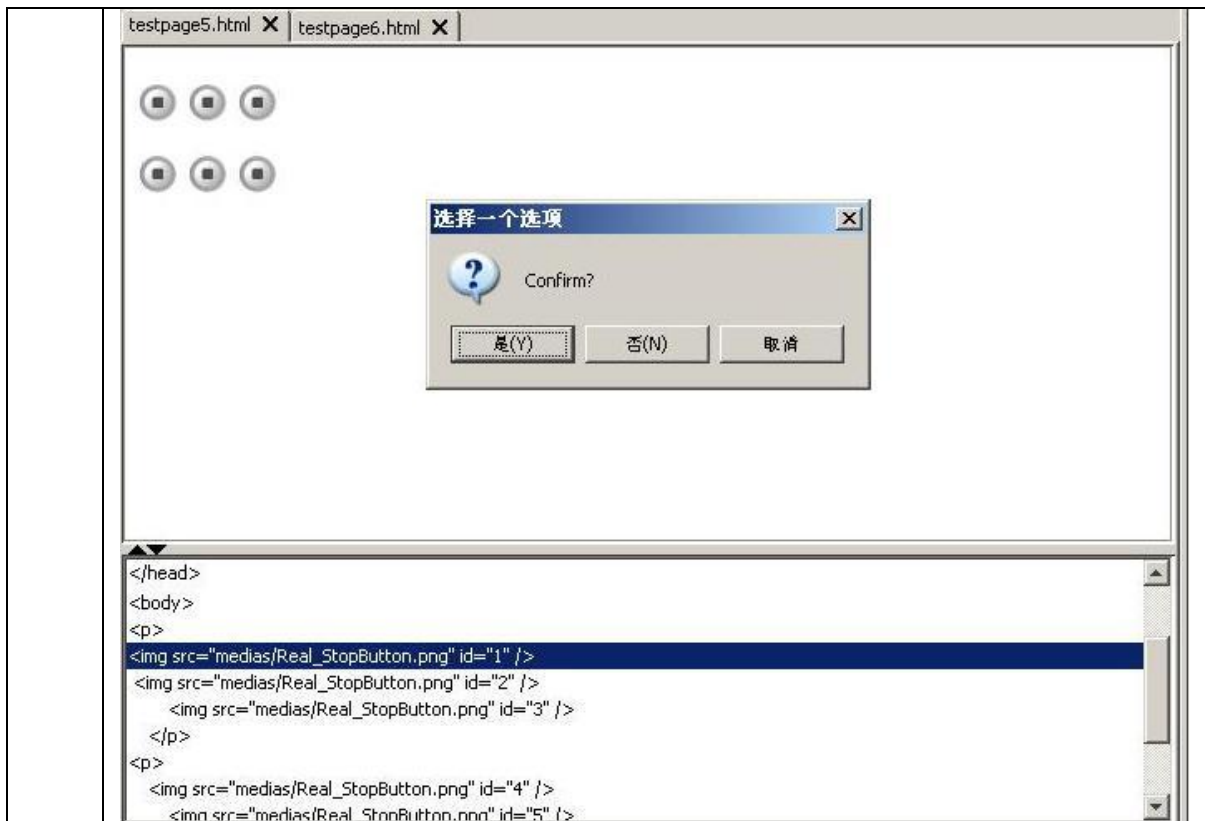


Figure 5.32: A dialog window pops up for confirmation

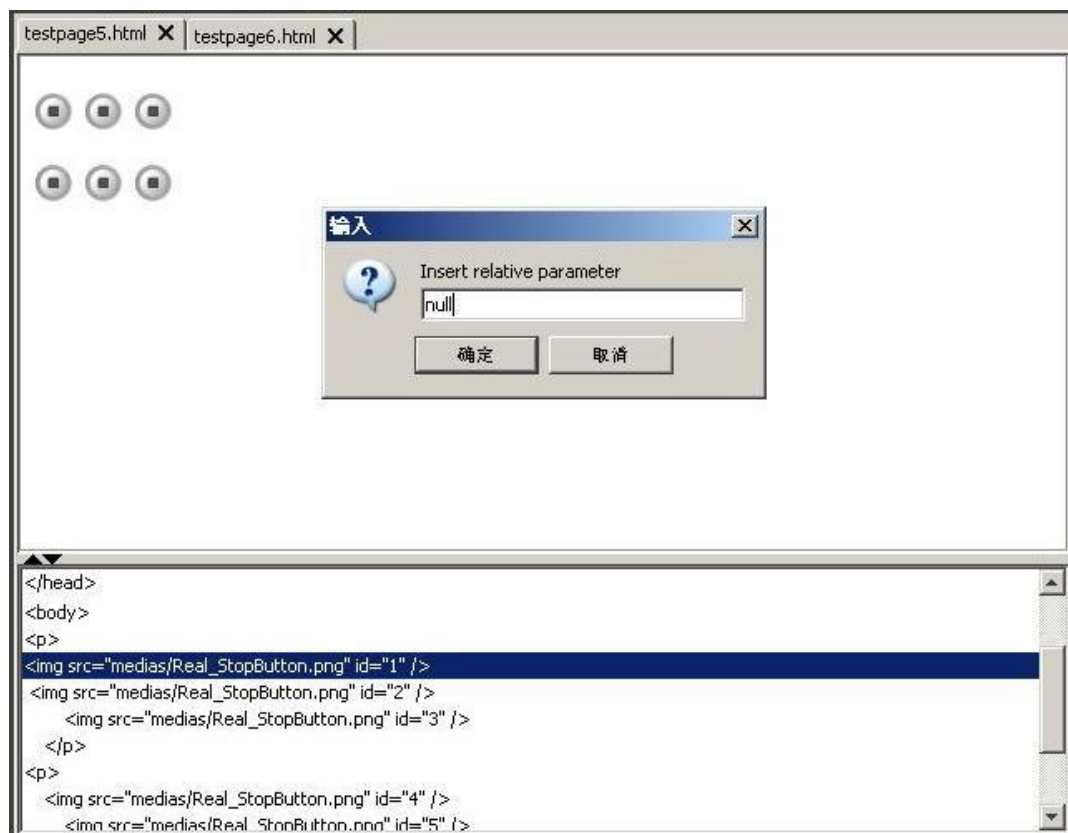


Figure 5.33: Another dialog window pops up for relevant parameter after confirmation

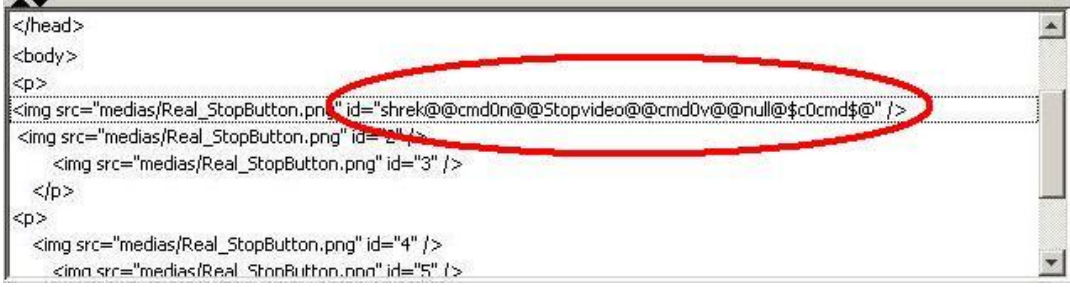
	 <p>Figure 5.34: A special ID that based on XHTML and “ID Event” presentation method is added as an attribute in the target element</p>
	<p>Evaluation: Successful output; the editor selects an interactive application from the RightPane; this can be dragged and dropped to the target element in the service page to implement the interactive application configuration.</p>
5	<p>“Save XHTML File” hotkey of the Hotkeys</p> <p>Evaluation: Successful output; after pressing the “Save XHTML File” in the hotkey bar, all the modifications to the target service page according to the XHTML and “ID Event” presentation method is saved and a corresponding reconfigured “IDEventFactory.java” for the terminal platform update is outputted automatically to the specified directory.</p> <p>“Save XHTML File...” option of the File Menu</p> <p>Evaluation: Successful output; after pressing the “Save XHTML File” in the Menu bar, all the modifications to the target service page according to the XHTML and “ID Event” presentation method is saved and a corresponding reconfigured “IDEventFactory.java” for the terminal platform update is outputted automatically to the specified directory.</p>
UI testing result evaluation	
<p>The UI testing cases have been successfully executed together with White-box testing. All the playback results match the corresponding recorded expected outputs. Overall, the software has passed UI testing with no errors thrown.</p>	

Table 5.17: UI testing result and evaluation

5.3.3 TEST RESULT EVALUATION

According to all the test results gained from the above implementation of the test cases and the testing process, we are able to evaluate the proposed semi-automatic service creation tool as follow: the results of the integration testing and the system testing have shown that the components of the proposed software are able to work properly with each other or together as an integrated software with minor errors. Moreover, the proposed software is tested to have all the functions as required and further be able to assist designers manipulating and creating XHTML and “ID Event” based MDTV interactive service pages semi-automatically.

Both of the errors occurred during the testing process do not affect the functionality of the proposed software. Also they are considered not originally caused by the author’s

design: the first error occurred because the Java SE Swing API does not support well to the new file system in Windows 7; the second error occurred because the default XHTML parser within the Java SE SDK has bugs and could not render the XHTML pages properly sometimes for some unknown reason (those pages were rendered successfully in Microsoft Internet Explorer 7). Some extra configuration work for the target software is needed to avoid this error.

5.4 TESTING EXECUTION OF THE MDTV SERVICE TERMINAL SOFTWARE PLATFORM

5.4.1 TESTING PREPARATION

The testing environment refers to Table 5.1 and the test data and the test case design refer to the description in Section 5.2.3.

5.4.2 TEST RESULTS

Integration Testing: The interface testing cases have been successfully executed. Interfaces 1, 2, 3, 4, 5 have had their actual outputs matched with the corresponding expected outputs as listed in Table 5.10. The software functional components are therefore tested to interface well with each other and the target software has thus passed the integration testing successfully with no errors.

Black-box Testing: The equivalent partitioning input testing method has been successfully executed. All the inputs except for case No.6 have had their actual result outputs matched to their corresponding expected outputs as listed in Table 5.11. Errors occurred when Case No.6 was executed: although the video was playing normally, the video frame was not displayed in the correct position within the service page (refer to Figure 5.35). This has caused some of other GUI components of the service page to be covered and also the lower part of the video frame was cut off by the bottom edge of the screen. As a result, the target software has passed the black-box testing

successfully with an error occurring in the layout implementation.



Figure 5.35: Failure on displaying video frame in proper layout

White-box Testing: Test cases have also been successfully executed. All the actual result outputs have matched the corresponding expected outputs listed in Table 5.13 with the exception of two of which failed:

1. When the service platform was trying to display the “testpage6.html”, the text components were not displayed in the proper layout (refer to Figure 5.36).



Figure 5.36: Failed to display the two text components in proper layout

2. When the “play video” application in the “testpage5.html” was executed, the service platform failed to display the video frame in the proper layout resulting in some of the other GUI components on the service page to be covered by it; the video

frame does not fit the screen and the lower part of the video frame is cut off by the bottom edge of screen (refer to Figure 5.35).

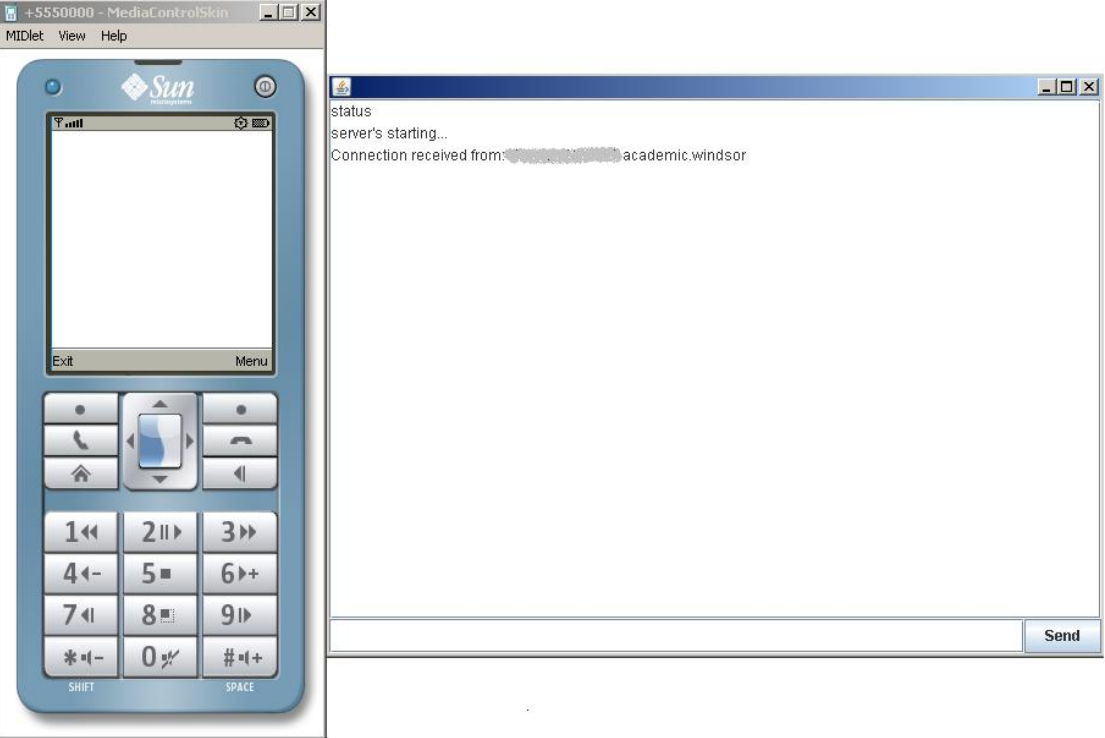
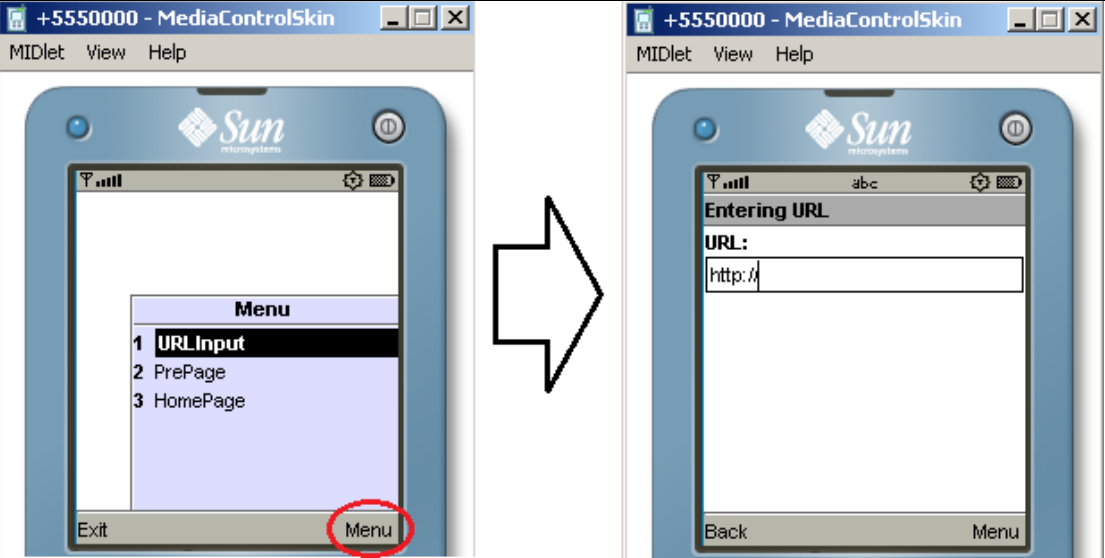
UI testing result	
Case No.	Target UI Component
1	Step 2: Start the demo server and BrowserMIDlet
	
	<p>Figure 5.37</p> <p>Evaluation: The demo server starts first and when the BrowserMIDlet starts up and automatically connects to the server, the demo server reports that the connection was successful. Successful output as expect.</p>
	Step 3 “URLInput” command of the HomeCanvas
	

Figure 5.38

Evaluation: When pressing the “Menu” option, the menu pops up; after selecting the “1 URLInput”, the screen refreshes and the URLInput page is set on top of the screen with the text field ready to be edited. Successful output as expected.

Step 4-1: “Text field” manipulation of the URLInput

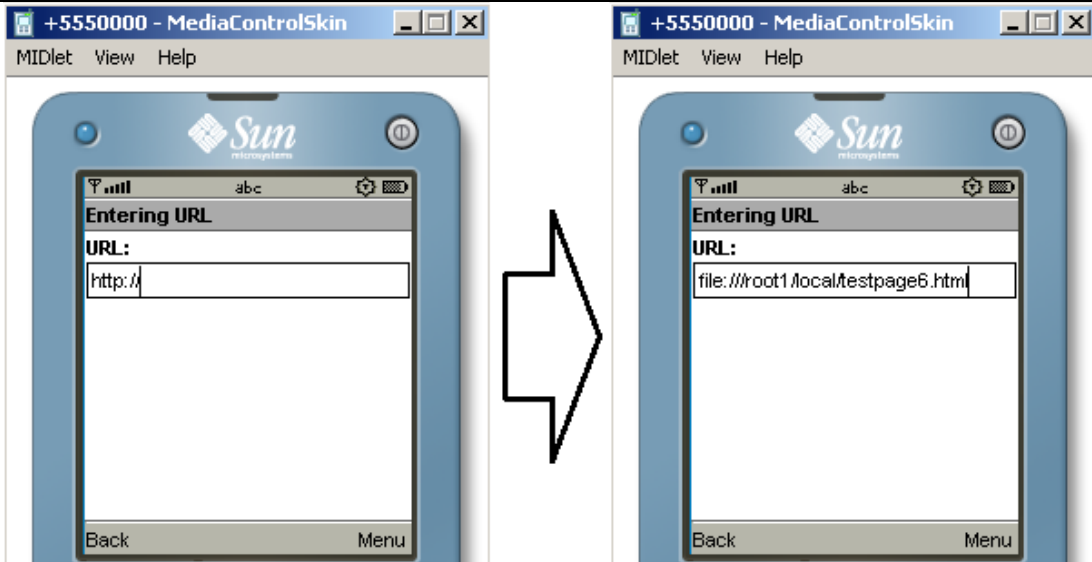


Figure 5.39

Evaluation: The user is able to enter the service URL into the text field. Successful output as expected.

Step 4-2: “Back” command of the MIDlet Menu

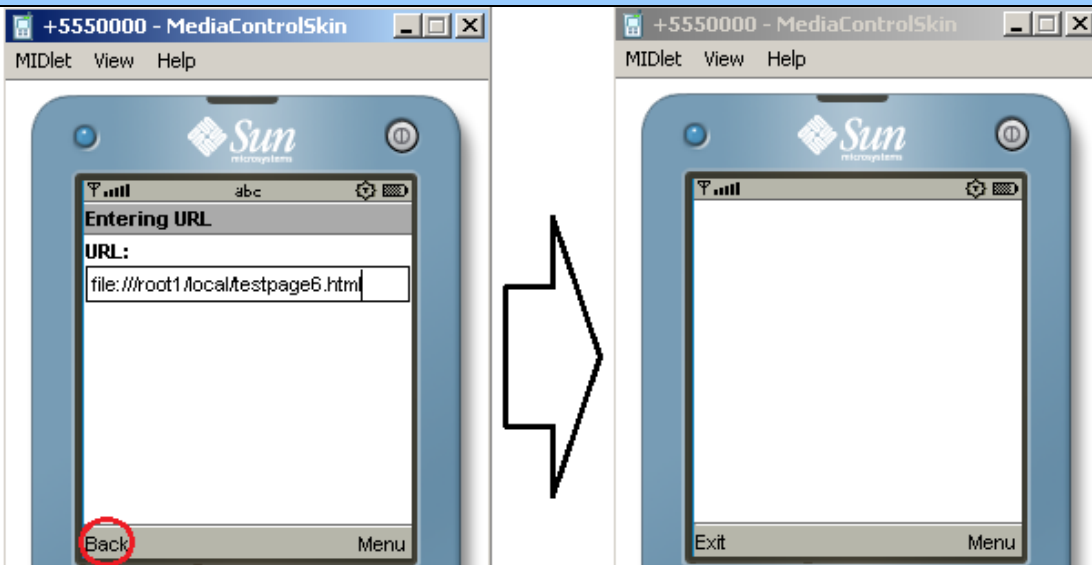


Figure 5.40

Evaluation: When pressing the “back” option, the screen goes back to the HomeCanvas page

Step 4-3: “URLget” menu command of the URLInput

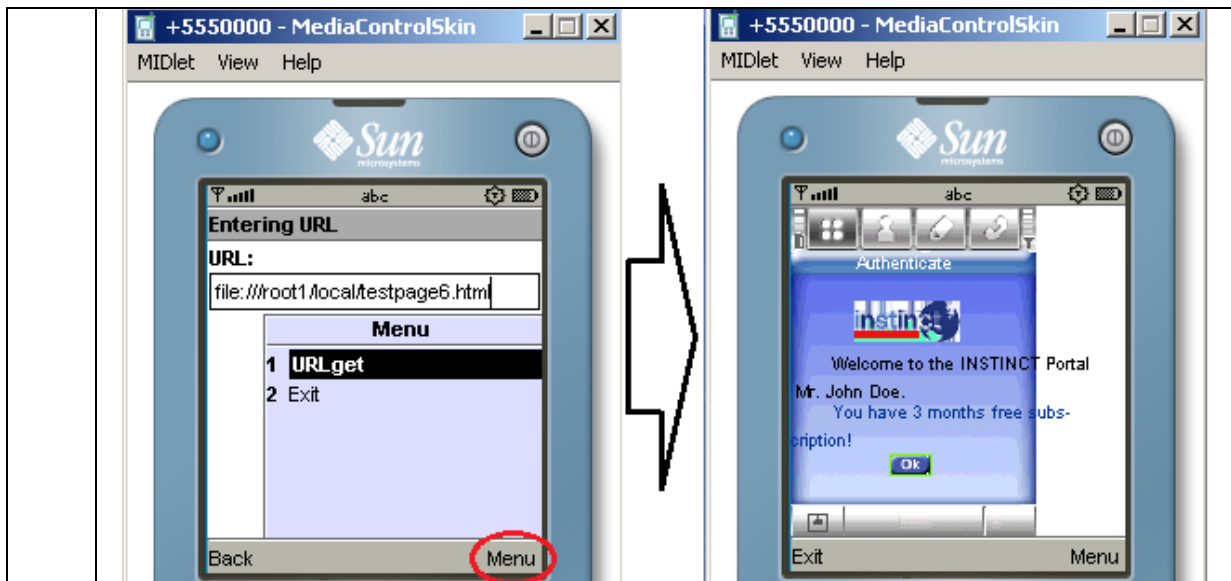


Figure 5.41

Evaluation: When pressing the “Menu” option and selecting the “URLInput” after entering the service URL, the service page is displayed in the HomeCanvas. Successful output as expected.

“Homepage” command of the HomeCanvas Menu



Figure 5.42

Evaluation: After selecting the “HomePage” selection in the menu, the default service home page is displayed in the HomeCanvas. Successful output as expected.

3 “General web-like navigation” operation of the HomeCanvas

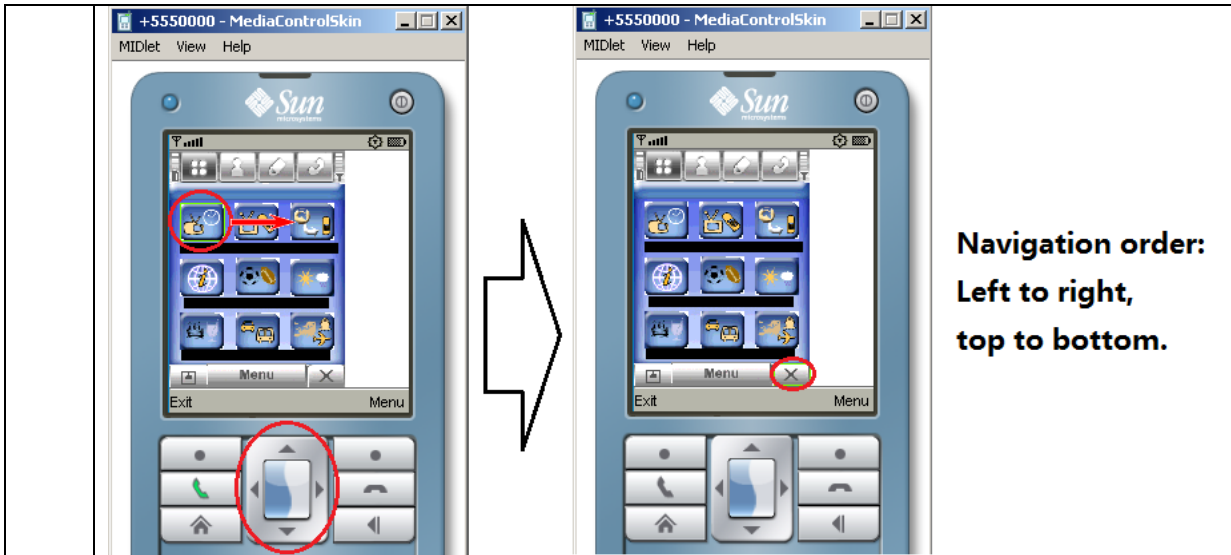


Figure 5.43: Navigation within the same service page

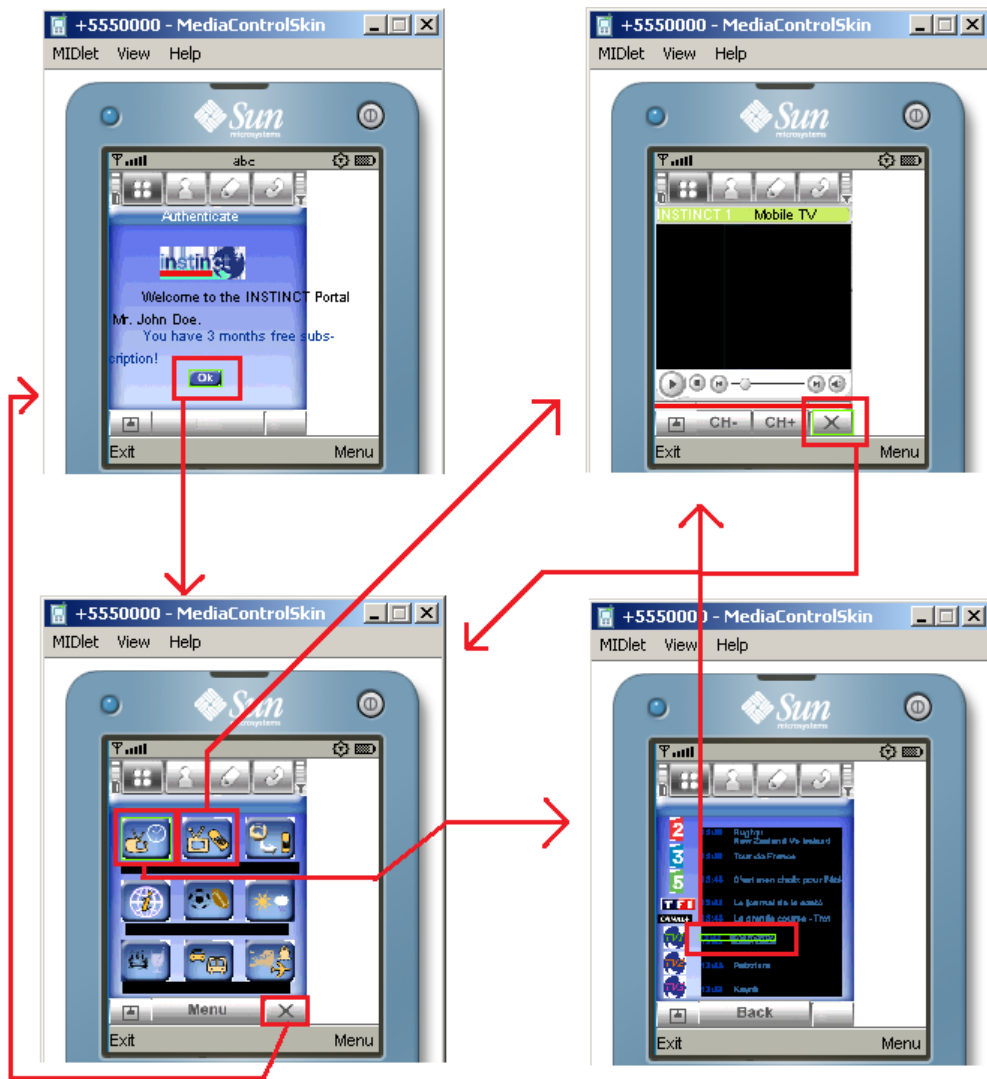


Figure 5.44: Navigation between different service pages

Evaluation: The navigation order within a page is: left to right and top to bottom. When pressing the “up” or “left” button, the navigation rectangle goes to the previous active item; when pressing the “down” or “right” button, the navigation rectangle goes to the previous active item. When selecting a

hyperlink in a service page, the linked page shows up on the front. Successful output as expected.

“Prepage” command of the HomeCanvas Menu



Figure 5.45

Evaluation: When selecting the “PrePage” option in the menu, the scene goes back to the service page, which is the one the tester has already browsed before the current page. Successful output as expected.

“Interactive application” operation of the HomeCanvas

4

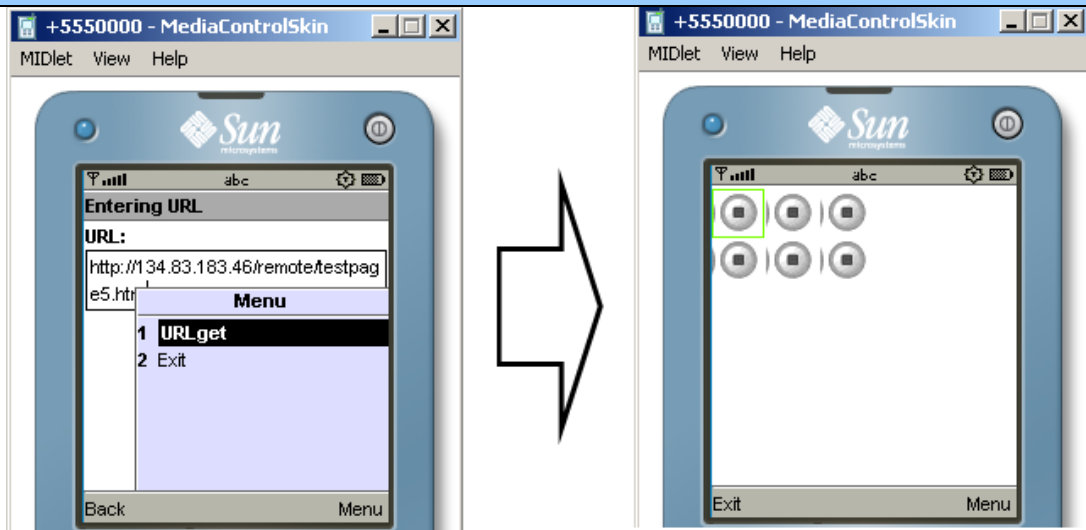
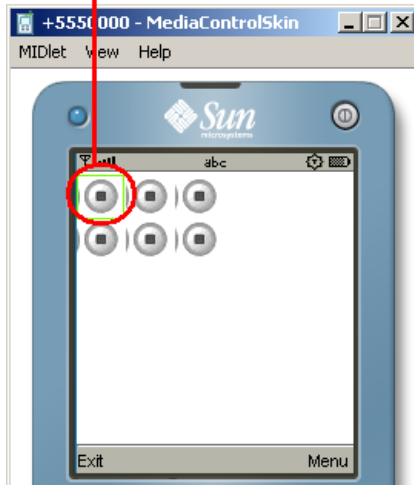


Figure 5.46: Open the interactive application testing page, testpage5.html

"Stop video" application



Press the "stop video" button to stop playing the video.

"Play video" applicaiton



Press the "play video" button to start playing the the video program.

Figure 5.47: "play video" and "stop video" applications



"Request for live data feed" application

Live data appears at the north part of the screen.

The bottom of the page is cut once the live data application is running.

Figure 5.48: "request for live data feed" application

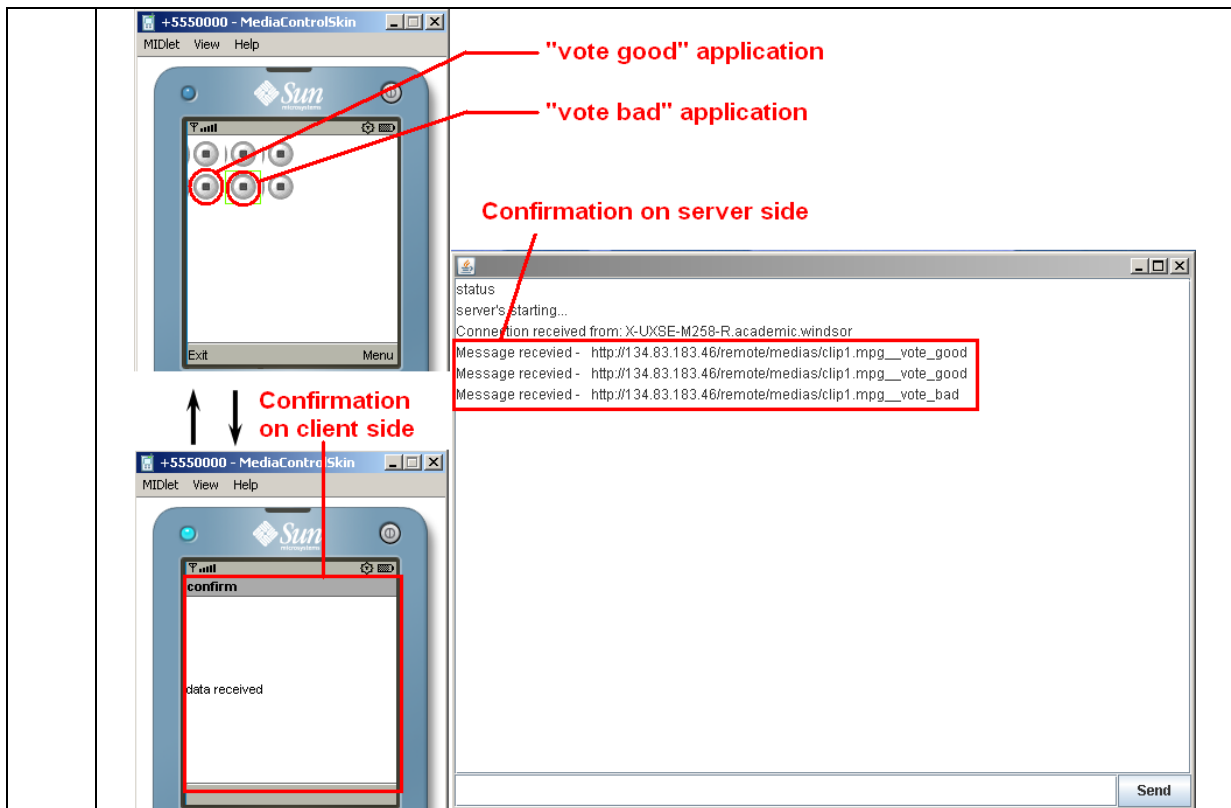


Figure 5.49: Vote applications: "vote good" and "vote bad"

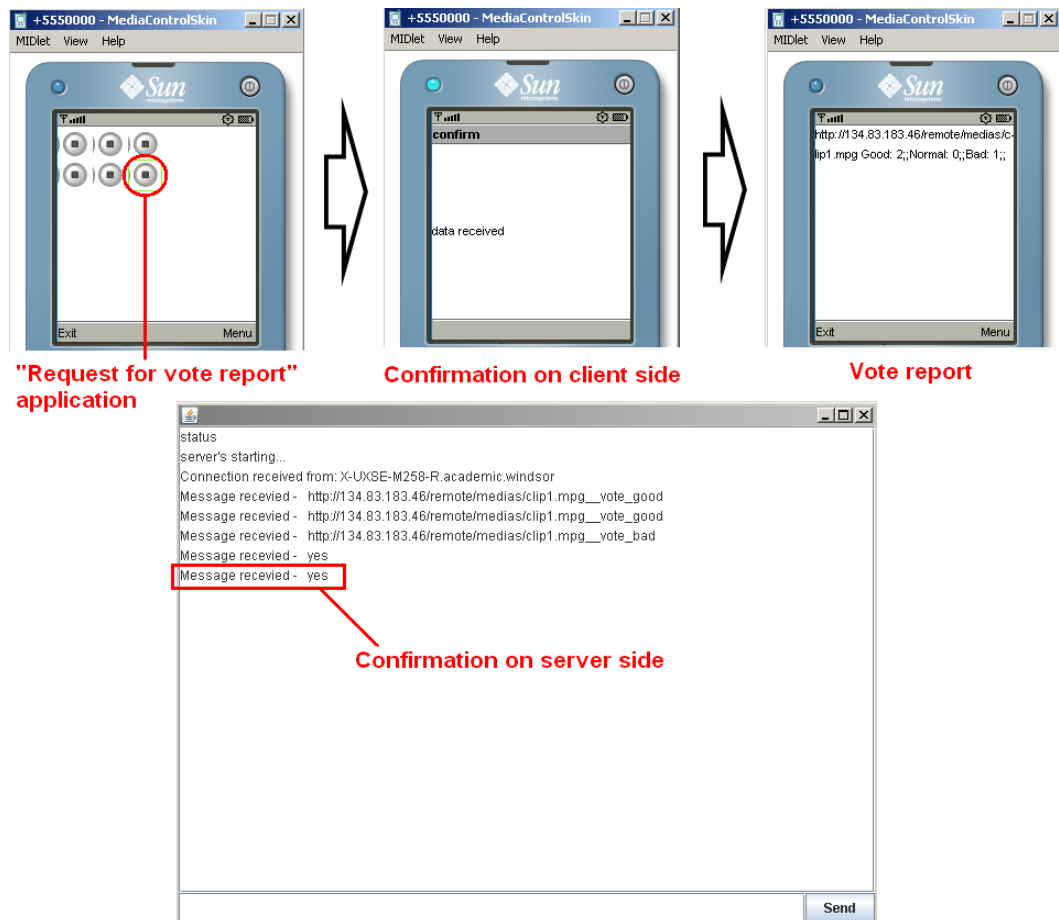


Figure 5.50: Vote application – "request for vote report"

Evaluation: By conducting operations through the I/O unit of the emulator and the relevant UI of the proposed service platform, all six interactive applications based on the XHTML and “ID Event” presentation method performed well and their corresponding outputs were the same as expected. Some errors occurred on displaying a few service pages (testpage5 and testpage6) during this testing case due to the insufficient layout engine of the service platform.

“Exit” command of MIDlet Menu

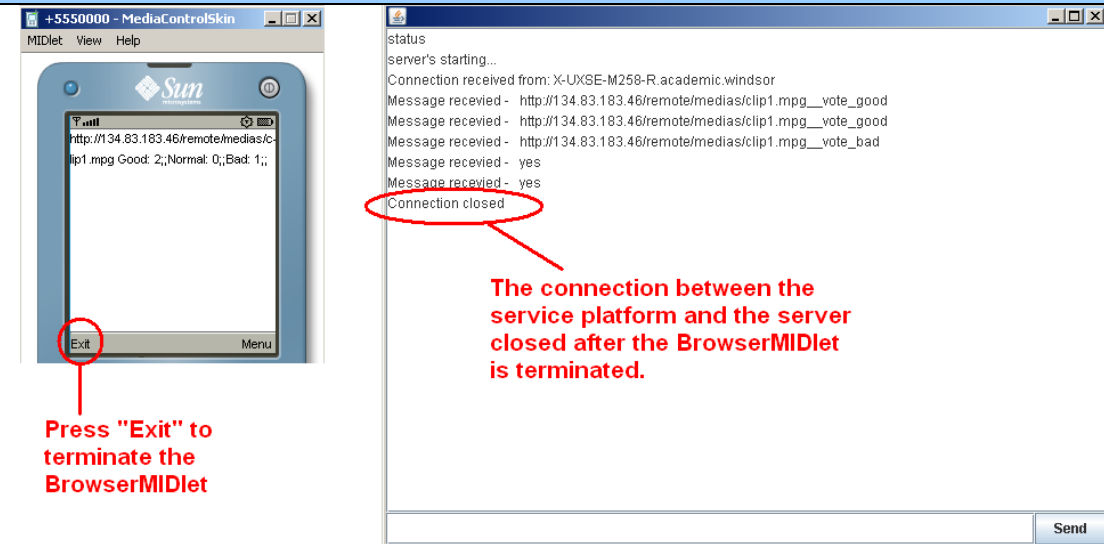


Figure 5.51

Evaluation: Successful output as expected.

UI testing result evaluation

All the UI testing cases have been successfully executed. All the UI components worked well as expected and the corresponding “playback” outputs have matched the “record” outputs; however an error was caused by a bug in the layout engine of the proposed service platform.

Table 5.18: UI testing result and evaluation

5.4.3 TEST RESULT EVALUATION

According to all the test results gained from the above implementation of the test cases and the testing process, we are able to evaluate the proposed terminal device service platform as follow: the results of the integration testing and the system testing have shown that the components of the proposed software are able to work properly with each other or together as an integrated software with minor errors. Moreover, the proposed software is tested to have all the functions as required and further be able to assist the implementation of the semi-automatic service creation process by rendering the XHTML and “ID Event” based MDTV interactive service page as well as enabling the consumption of such MDTV service for the user.

However two bugs have been found out during the test. Both of them are considered

to be caused by the author's design although these bugs do not affect too much to the software functionality. The first bug was caught when playing the video stream that the video frame size was too big to cover most of the other service components behind. This is due to a page layout rendering bug in the layout engine of the proposed service platform that the video frame size could not be set to fit in the page layout properly when there are video components in a service page. The second bug was caught when the service platform was trying to display a service page with “<div>text</div>” elements. The text in the “<div>” element could not be wrapped to fit in the page layout properly and this is considered to be a bug within the layout engine again. The layout engine could either not be able to render the position attributes of the “<div>” element properly or not be able to wrap the text components within the “<div>” element properly. Both of these aforementioned bugs have been recorded as the reference for the modification work to the proposed service platform in the future.

5.5 INTEGRATION TESTING OF THE SERVICE CREATION TOOL AND SERVICE PLATFORM

Test Result: All the integration testing cases have been successfully executed. All the result outputs have matched the corresponding expected outputs as listed in Table 5.16. The service platform is able to render the service pages outputted from the service creation tool. If the service platform is not updated with the new version of “IDEventFactory.java” outputted from the service creation tool, it could not be able to identify and respond to the newly-added interactive applications on those service pages. Once the service platform is updated with the new version of the “IDEventFactory.java”, it is able to identify those interactive applications on the service pages and further respond to the user's interactions. Overall, the semi-automatic service creation tool together with the terminal service platform has passed the integration testing with no errors.

Evaluation: According to result gained from the above integration testing between the two proposed software, we can further evaluate as follow: the service platform is able to render the MDTV service page outputted from the semi-automatic service creation tool; by replacing the original version of the functional class (“IDEEventFactory.java”) with the new version outputted from the service creation tool, the service platform is able to render and handle new interactive applications added to the service page. As a result, two software components are able to work with each other properly to enable the functionality of the proposed MDTV service creation and consumption system and further achieve the Thesis’s aim.

5.6 ACCEPTANCE TESTING

The acceptance testing of the proposed service creation tool and service platform is necessary since we need to investigate whether the target user requirements are satisfied and if the usability of the software meets the target user needs. By conducting the acceptance testing, we will be able to evaluate:

- Whether the semi-automatic service creation tool and the terminal device service platform could achieve their functionalities with no major error in practice?
- How is the usability of these two software components in practice including the function and the UI performance?

We here design a series of acceptance test cases by using Alpha testing methods.

5.6.1 ALPHA TESTING PREPARATION

First we defined the principle objectives as follow:

- a) To test if the two proposed software is usable to achieve the relative software design requirements;
- b) To test if the users are satisfied with the graphic user interface (GUI) of the two

software;

c) To test whether there are any errors and problems from the users' perspective;

d) To test to find out what improvement can be made to software, its functionality and GUI, as suggested by the user feedback.

The test was conducted in a public research student office located in Brunel University campus. Since the main strategy of this test is based on a qualitative rather than quantitative analysis, 10 persons with different profiles (age and occupations) were thus chosen to participate in the test. A Desktop PC was employed and preset as the computer environment for the participants to do the test cases.

Before preparing the test case design and other related documents, an application form to the University's Ethics Committee was submitted to formally apply for the permission of launching such test. The University's Ethics Committee granted my application and the ethics approval was then issued to me. Based on the ethics approval, a consent form was attached into the test documents for participants and the participants were asked to sign the consent form before they started the test procedure. All the test documents including the ethics approval, the consent form as well as other test documents are included in the Appendix 1. Then the test case was designed as follow:

- In the test procedure, the participants are firstly asked to sit in front of the Desktop PC for the test use. The participants are then offered a copy of the test document to read. The document includes the consent form and the task instruction;
- The test starts when the participant has read through the consent form and is willing to sign on it for his/her participation
- The participant is asked to read the task instruction and conduct the Task one and two by following the steps in the instruction. The entire process is under the tester's observation and the participant is free to ask for help from the tester

during the process. The tester completes the observation record as the testing case designed during the process;

- Once the tasks are finished, the participant is asked to finish a questionnaire related to the tasks he has conducted.
- The participant hands in the questionnaire once (s)he finishes it, along with the signed consent form. The testing process on this participant is accomplished and the whole testing procedure above shall start over for the next participant.

The task details can be found in the Appendix 2. The testing process observation record form for the tester can also be found in the Appendix 2.

5.6.2 RESULTS AND EVALUATIONS

5.6.2.1 Demographics

Criteria	Details
Age range	26-35: 6 participants; 36-45: 3 participants; Unknown: 1 participant
Gender	Male: 9 participants; Female: 1 participants
Occupation	Student: 9 participants; Research Fellow: 1 participant
Graduation major	Mechanical Engineering: 1 participant; Electronic and Computer Engineering: 2 participants; Material Science: 1 participant; Computer Science: 1 participant; Wireless Communication: 2 participant; Medical science: 1 participant; Unknown: 2 participants

Table 5.19: Participant Profiles

5.6.2.2 Test result evaluation of Task 1

Based on the results (refer to Table A in Appendix 3 and Figure 5.52) gained from the Alpha Testing Questionnaire of Task 1, we evaluate the semi-automatic service creation tool as follow:

- The majority of the participants think that the target software is useful overall, regardless of their experience on the software engineering profession;
- The majority of the participants feel the target software easy to use in overall but 3 out of 10 participants had difficulty in manipulating the service page by editing the source code and 3 out of 10 participants had difficulty in assigning a function

to a UI component. Refer to the corresponding results of question 1, 2 and 3 in Phase 1, this reaction tends to be related to their knowledge and experience on the corresponding techniques (such as HTML authoring experience);

- All the participants are satisfied with the design of the target software GUI and the majority of them feel it is easy to learn and use;
- The majority of the participants think that the target software is useful in assisting them to accomplish Task 1 although 4 out of 10 participants think the that software was not as effective as they expected during the task;

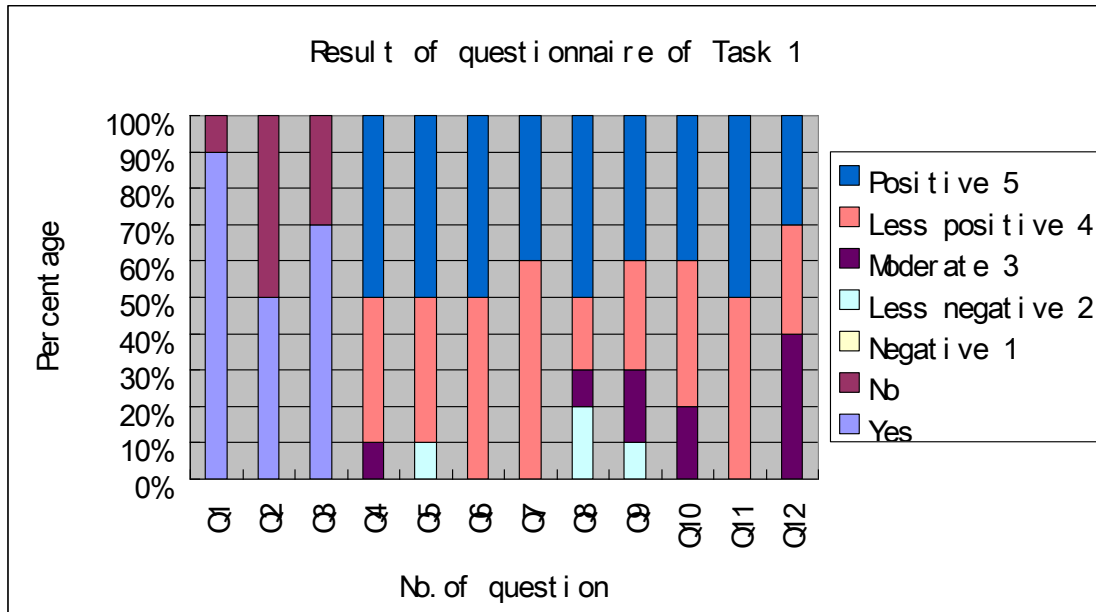


Figure 5.52: Bar chart for the Alpha Testing Questionnaire of Task 1

Based on the analysis on the Alpha Testing Process Observation Record of Task 1 (refer to Table A in Appendix 3), we evaluate the proposed semi-automatic service creation tool as follows:

- All the participants were able to use the proposed software to accomplish the given task, indicating that the software is usable and able to achieve its design requirement;
- The software GUI is easy to use since participants paused only 3.3 times on average due to difficulty with the GUI and most of the difficulties occurred because the participants could not find the “Element Mode” hotkey button;
- The target software’s performance during the tasks has been acceptable since no

errors occurred throughout the entire Task 1 with 10 participants participating but the software crashed 2 times out of 10 times of operations;

5.6.2.3 Suggestions to the proposed semi-automatic service creation tool after Task 1

The participant suggestions were collected throughout the Acceptance Testing Questionnaire Phase 1, Question 13 and 14. Most of the participant suggestions for Task 1 focus on the GUI aspect. Participants suggest that there should be more functional hotkeys on the hotkey bar for participants to execute the functions more easily than looking up them in the menu list; Participants also suggest that a help function should be added into the software; Besides, a few participants suggested that the hotkeys should be designed to be more recognizable among the different GUI components so as to make it easier for participants to find these.

5.6.2.4 Testing result evaluation of Task 2

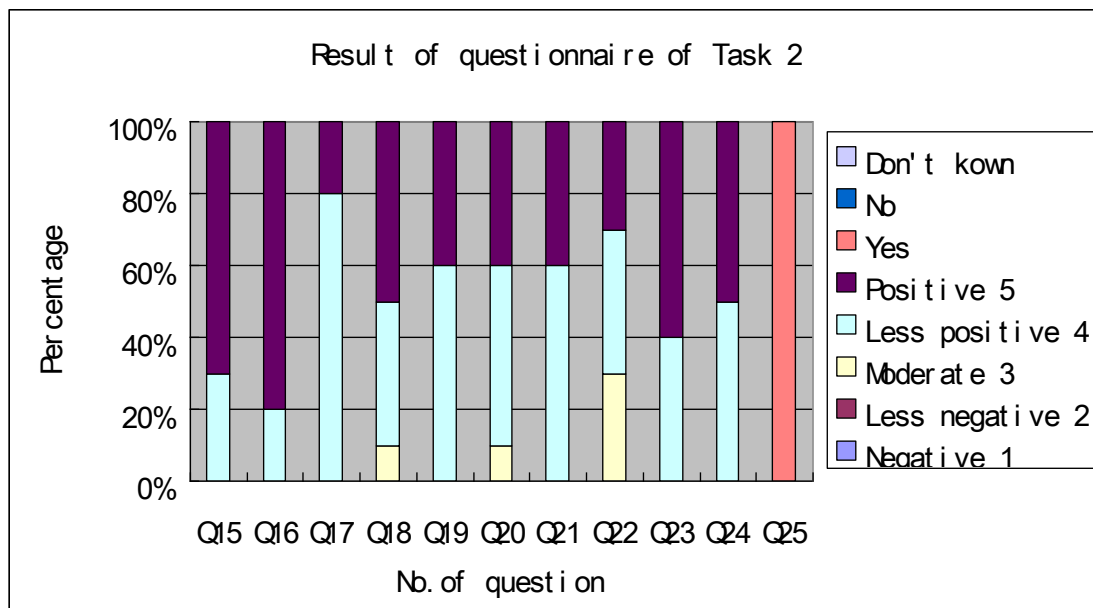


Figure 5.53: Bar chart for Alpha Testing Questionnaire of Task 2

Based on the analysis on the results (refer to Table B in Appendix 3 and Figure 5.53) of the Alpha Testing Questionnaire of Task 2, we evaluated the proposed MDTV service platform as follows:

- All participants think the target software is very good overall and useful in

assisting them to perform the required tasks.

- 8 out of 10 of the participants think the software performed very well during Task 2; more than 5 participants out of 10 think that the service page rendering speed of the software is fast; The design of the navigation algorithm within the same page is ranked very good by all participants but 6 of them did not give the top rank as they were expecting the vertical navigation also available rather than merely navigating horizontally. The design of the navigation algorithm between service pages was considered to be acceptable by majority of participants but 3 participants had difficulties with it; this is mainly because the navigation was not as direct by simply pressing a button but instead participants had to navigate to the specified button first before the button could be pressed.
- The performance of the interactive applications is considered to be good by all the participants. Moreover we can infer from this result that the interactive applications based on the XHTML and “ID Event” presentation method are acceptable to all participants.
- 9 out of 10 participants think that the GUI is easy to use and all the participants commented positive on the GUI overall design. One of the main reason why 6 of the participants did not rate “very good” on the GUI design is that some of the participants were not used to use the handset emulator software environment; most of the participants had to get familiarised with the emulator first before they could start using the software to be tested.
- All the participants that participated in the test expressed very positive views on the question of interacting with Mobile TV services/applications if they were provided with such services.

Based on the analysis of the Alpha Testing Process Observation Record of Task 2, we evaluate the proposed MDTV service platform as follow:

- All the participants were able to use the proposed software to accomplish the given task, which indicates that the software is usable and able to achieve its design requirement;

- The software GUI is easy to use since participants paused only 3.4 times on average due to difficulty with the GUI and most of the difficulties occurred because some of the uses were unfamiliar with the emulator environment and some navigation design drawbacks confused the participants during the task.
- The target software's performance during the tasks has been acceptable since no errors occurred throughout the entire Task 2 but the software crashed 2 times during 10 times of operations.

5.6.2.5 Suggestions to the proposed MDTV service platform after Task 2

Participant suggestions were collected throughout the Acceptance Testing Questionnaire Phase 2, Questions 26 and 27. Two aspects of suggestions are as follows:

1. Improvement Suggestion:

- the service platform should also support touch screen to receive a better participant experience;
- The navigation system should be further improved to be more convenient to use such as supporting the vertical navigation within the same page and make some of hotkeys or functional components linked to the keyboard number buttons.

2. Ideas on new type of interactive application:

- TV-online shopping;
- Interactive advertisement;
- Shopping assistant service such as locating the nearest stores and etc.;
- RSS feeds.

5.7 COMPARISON WITH INSTINCT SERVICE CREATION PROCESS

Having conducted a series of testing cases, the two proposed software components have been validated against their key functionalities and design requirements. More importantly, this further proves that the proposed semi-automatic service creation and

consumption system is effective on facilitating the current MDTV service creation in practice. In order to present the advantage of the Thesis's proposal more clearly, a comparison between the Thesis's proposed solution and a practical DTV service implementation solution, based on the INSTINCT project (which is the most relevant similar to the Thesis's work), is to be performed as a part of the overall testing. According to the different scopes of these two solutions (as discussed in Chapter 3), the comparison will mainly focus of the service creation process.

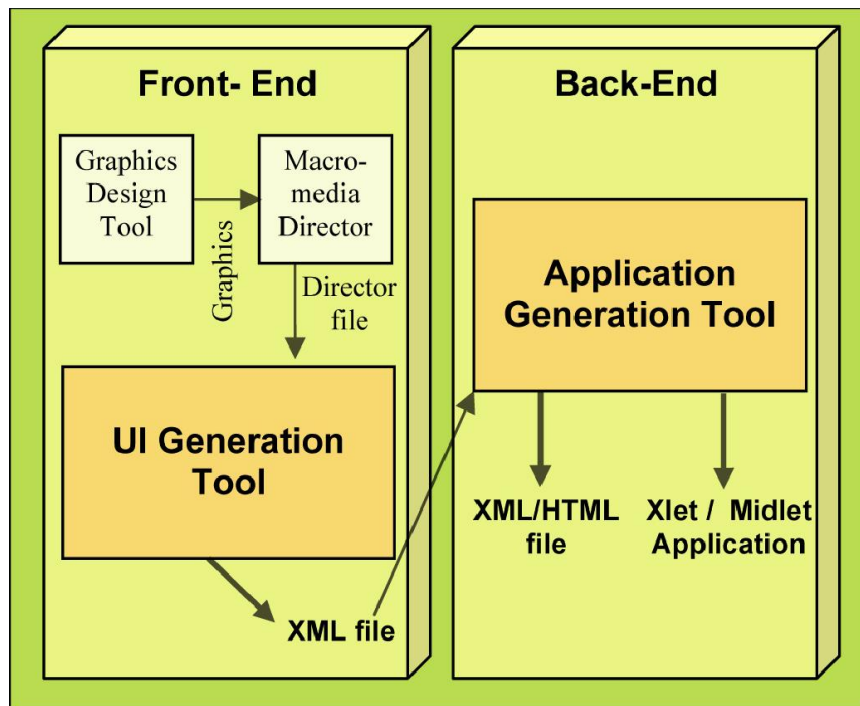
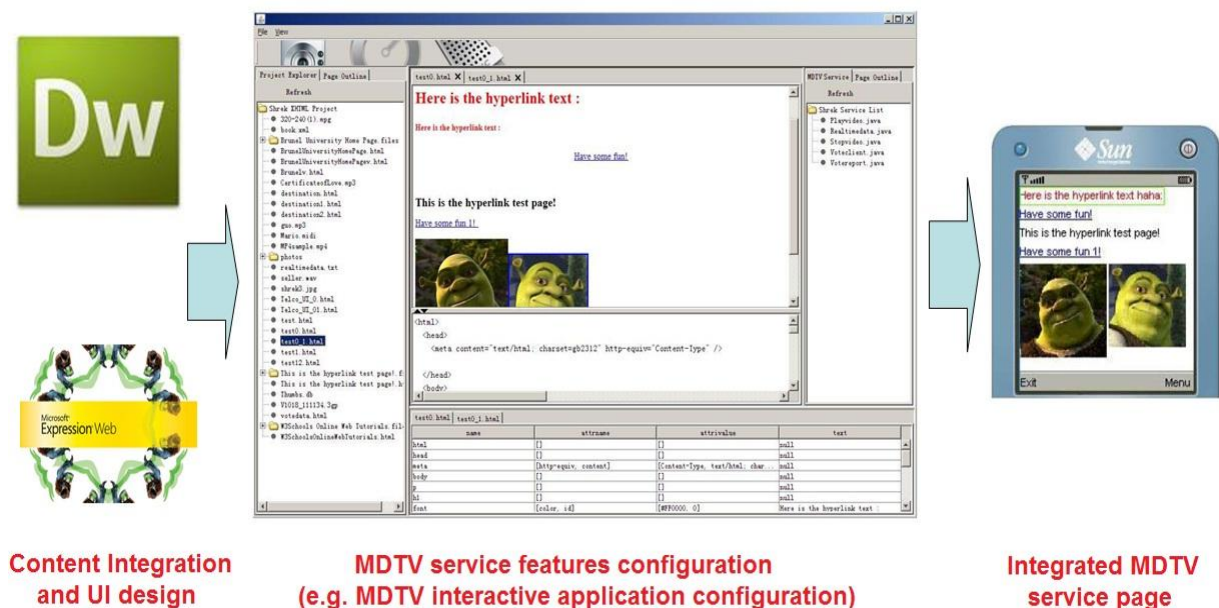


Figure 5.54: INSTINCT project service creation process ([132] Figure3)



Content Integration and UI design

MDTV service features configuration (e.g. MDTV interactive application configuration)

Integrated MDTV service page

Figure 5.55: Thesis's proposed semi-automatic service creation process

In the INSTINCT project, the service creation process includes two steps (as illustrated in Figure 5.54): the front-end for the UI components creation and the back-end for the interactive components creation. On the front-end creation segment, the designer firstly creates the graphical components by using commercial graphics authoring tool such as Photoshop, in order to prepare original materials for the UI design. With these graphical components, the designer then uses Macromedia Director to design the prototype of the service UI and layout. The project proposed UI Generation Tool is then used by the designer to convert the Director formatted service UI into a custom XML language designed specifically for that project. On the back-end creation segment, the designer will utilize the project's proposed Application Generation Tool to further edit the XML formatted service UI by mapping iTV applications (e.g. interactive application) to the UI components according to the service design requirements. Finally, the Application Generation Tool will assist the designer in converting the XML formatted iTV service into the requested format (such as HTML format) to finally output the integrated version of the iTV service.

Comparing the above process with the Thesis's proposed process (as presented in Chapter 3 and tested in Chapter 5), the Thesis's solution has adopted a similar methodology as the INSTINCT solution during the back-end interaction creation. However on the front-end UI creation segment, the proposed semi-automatic service creation process shows advantages on several aspects:

The XHTML has been adopted in the proposed creation process instead of the custom XML language developed in the INSTINCT project. This is one of the biggest advantages when comparing with INSTINCT solution: firstly, as discussed in Chapter 3, the XML is not suitable for visual presentation and a XML based visual presentation technologies usually requires extra parser or rendering engine. Thus in the INSTICT solution, a customized XML schema is introduced to assist XML achieving the UI generation. However, this customized schema is not compliant well

to the current DTV/MDTV service implementations and thus additional tools have been developed to reformat it to be more compatible to the bearer standards. Extra time and effort will have to be conducted in this case especially for all the conversion processes. In contrast, the Thesis's proposed creation process involves widely-adopted XHTML technology that is compatible with most of the DTV/MDTV standard. The Thesis's solution strictly follows the syntax in W3C XHTML specification during the service creation without creating any additional protocol or schema. Therefore, no extra conversion process is needed in the Thesis's solution and the proposed service creation process is thus shorter and less complicated. Secondly, by utilizing XHTML, the "Marcomedia Director" step and the "UI Generation Tool" step of the INSTINCT solution can be merged and simplified as in the Thesis's solution. This is because by using XHTML authoring tools such as Adobe Dreamweaver and Microsoft Expression Web, the designer can do the UI prototyping and UI generation at the same time. Moreover, the XHTML authoring is relatively easier than the work based on Marcomedia Director due to the popularity of these two kinds of tools.

Moreover in the INSTINCT project, the relative testing performed on the UI generation segment has revealed that the efficiency of the conversion from the Director file to the XML file is not as expected. This is due to the Marcomedia Director has various functions and the UI prototyping is only a small subset of them. Thus the conversion may fail unless specific set of design and authoring rules are followed prior to the import of the file in the UI Generation Tool. In other words, the designer must ensure his/her design follows specific requirements and guidelines when using Marcomedia Director [132]. As to the Thesis's creation process, the XHTML technology is much more familiar to most of the design oriented professionals. More importantly, most of the XHTML authoring tools have their main focus concentrated on visual design and the W3C XHTML syntax is always followed during the service creation. Therefore the designer can employ conventional web service design processes and skills without additional attentions to any extra rules during the MDTV service creation.

In such comparison, the Thesis's proposed service creation process is therefore more efficient and less complicated than the INSTINCT's process. Several effective improvements have been conducted especially during the front-end segment to help designers accomplish the design work more easily and conveniently. Additionally with the assistant of the proposed client implementation environment including the service platform, the Thesis's proposed semi-automatic service creation and consumption solution is thus able to contribute the current MDTV industry field more evidently.

5.8 TEST CONCLUSION

By conducting a series of software testing procedures, we manage to evaluate the proposed software components more comprehensively. Through the component testing, integration testing and system testing to the software components, we have found the bugs and drawbacks in the target software and relative modifications will be done according to these findings. More importantly, these tests have verified that the target software components have been designed with solid logic and completed functionality. Both of the semi-automatic service creation tool and the terminal device service platform have matched their design requirements and further their functionality enables the proposed MDTV service creation and implementation solution. Through the Acceptance Testing, we managed to test the software components in practice and gain feedbacks from users. Such feedback has pointed out more problems existing in the software, which are also the valuable reference for future work. Moreover at the prototype stage of the proposed solution, the feedback has further validated the solutions' contributions:

The semi-automatic service creation tool has been considered to be useful during a MDTV service creation process. It has been able to help the participants achieve the MDTV interactive service page manipulation and creation, regardless their knowledge to the MDTV and the software engineering related technologies and skills. This

means as long as a designer has the knowledge on how to use computer programs and has the basic experience on XHTML authoring, he is able to do the MDTV service creation through the proposed tool. The technical demands to the designer are therefore reduced and the MDTV service creation process is further simplified. Moreover as mentioned in the Thesis's aim, more design oriented professionals are able to get involved more easily and the development of the MDTV service can be further encouraged.

Also through the terminal device service platform, most of the conventional services and interactive applications (such live data feeding and voting) have been implemented. This in turn means that without using ECMAScript, the proposed solution can achieve most of the MDTV applications as the commercial solutions could provide. The Java ME based "ID Event" method has thus been proved to be effective for presenting MDTV interactive applications. From this aspect, the potential benefits (as discussed in Chapter 4) of using Java ME technology and the "ID Event" method during the MDTV service implementation are proved.

Summary:

This chapter has implemented a series of software testing procedures to the proposed software components including semi-automatic service creation tool and the terminal device service platform. Bugs and drawbacks have been found through the test and will be used as the reference for future work; evaluation based on the test result has also been conducted that the proposed software components have been verified and validated to be able to achieve their functions to enable the proposed MDTV service creation and implementation solution. Moreover such proposed solution is evaluated to be able to conduct the Thesis's aim.

6. CHAPTER 6: FURTHER WORK AND CONCLUSION

6.1 FURTHER WORK

Based on the implementation of the proposed work and the corresponding test results, this section provides a set of discussions regarding improvements that can be made to the creation tool and the service platform developed as a part of this Thesis as well as some thoughts on future work.

6.1.1 SEMI-AUTOMATIC SERVICE CREATION TOOL

Firstly, several modifications to the software GUI are necessary to be conducted as it has been concluded by the testing results and evaluation in Chapter 5:

1. To re-design the hotkey bar by changing the outlook to be easier to recognize and by adding more functional hotkeys to improve the user experience and operation efficiency;
2. To update the page editing area in the central pane, to enable the page display frame to offer users direct service page manipulation rather than only source code manipulation that is currently provided;
3. To modify the project explorer in the left pane by adding file filtering function to prevent users from opening a file in any format other than “.html”.

Secondly, in order to integrate with the Mobile TV Electronic Service Guide (ESG) function better, we consider adding a new function to the creation tool that will automatically output the service page metadata file during the service creation process; this will facilitate the ESG creation process by allowing the user of our tool to output the relevant mobile interactive application service metadata in an automated way and in the format that is usually defined in the underlying MDTV transport layer

specifications (such as OMA-BCAST and DVB-CMBS).

Thirdly, we intend to make an enhancement to the XHTML and “ID Event” based MDTV service presentation method applied during the proposed service creation process. Such enhancement is considered to be conducted by involving SVG as assistant technology to Bitmap format that is currently used as the image format in the proposed presentation method. In Chapter 3, we have offered a discussion on the advantages and disadvantages of using Rich Media during the MDTV service creation and we chose XHTML as our proposed technology in the methodology instead of the Rich Media solution. Furthermore, in chapter 5, a series of test case results have indicated that the XHTML and “ID Event” based presentation method provides as good functionality as Rich Media, based on the participant feedback. However, the image format, Bitmap, which we apply in our presentation method, has several drawbacks: Bitmap is a format of graphics that stores the image as a map of bits, which means that the higher resolution of the graphics we demand, the more bits of storage medium or bandwidth are needed to store or transmit the graphics. This fact affects the performance the service on the MDTV terminal when there is a need of higher resolution to the image components on the service page. The page loading speed may be slower and the terminal device has to provide more storage space to store it and display it. Regarding to this problem, we are considering involving Scalable Vector Graphics (SVG) as the assistant graphics format during the proposed service creation process in the future work for the following reasons:

1. SVG is stored as XML code rather than map of bits and therefore less storage medium or bandwidth is needed when comparing with Bitmap. Moreover, the higher resolution is required, the more capacity will be saved;
2. SVG is based on XML, which can be integrated in the XHTML code during the service creation and transmission process;
3. SVG graphics complies to different resolution demands, which means that an

image in SVG only have to be created once for multiple resolution requirement; bitmap graphics on the other hand must be created several times according each different terminal resolution to ensure an acceptable visual quality;

4. More importantly, since the proposed terminal device service platform is developed with Java ME which has internal JSRs (JSR226 and JSR287) to support SVG Tiny, it is therefore much more convenient now to apply SVG during the proposed service creation process.

6.1.2 THE MDTV SERVICE PLATFORM

According to the testing results and evaluation presented in Chapter 5, several modifications to the MDTV service platform will be conducted:

1. The navigation system of the platform will be updated to provide more convenient navigation user experience: by updating it to support full positioning (vertical and horizontal) navigation; updating it to support hotkey operation by linking platform functions to MDTV terminal device I/O unit (such as a numbered keyboard);

2. Since touch screen technology is becoming more and more popular on mobile devices, we plan to redesign the software GUI to support touch screen operation in order to meet the new user requirement;

3. Correct the display bugs in the layout engine of the service platform to ensure that multiple components can be displayed in a proper layout. Enhance the layout engine to be compliant to multiple screen sizes and resolutions of the MDTV terminal devices;

4. Enhance the XHTML parsing engine to support more XHTML elements and attributes. Optimize the XHTML page parsing algorithm to improve the page parsing and rendering efficiency so as to shorten the service page loading time on the platform;

5. As the user requirement to the MDTV service evolves in the future, we plan to develop new interactive applications (such as interactive advertisement, internet access and wireless communication) to meet the popularity and to enrich the service platform aiming at developing an integrated multimedia data exchanging portal for the future.

Other aspects that will enhance the service platform, which are more related to new technology and better compatibility, are as follow:

1. Utilize JSR226 and JSR287 to develop a SVG Tiny rendering sub-system in order to integrate SVG Tiny support on the service platform;
2. When the JSR272 standard is implemented and the relevant version become available, we aim to develop the service platform further to integrate it with the JSR272 MDTV middleware in order to match the software environment better to the MDTV system. This will be conducted by adding a functional sub-system to support general MDTV functions described in JSR272 (such as user account management, ESG and several more.);

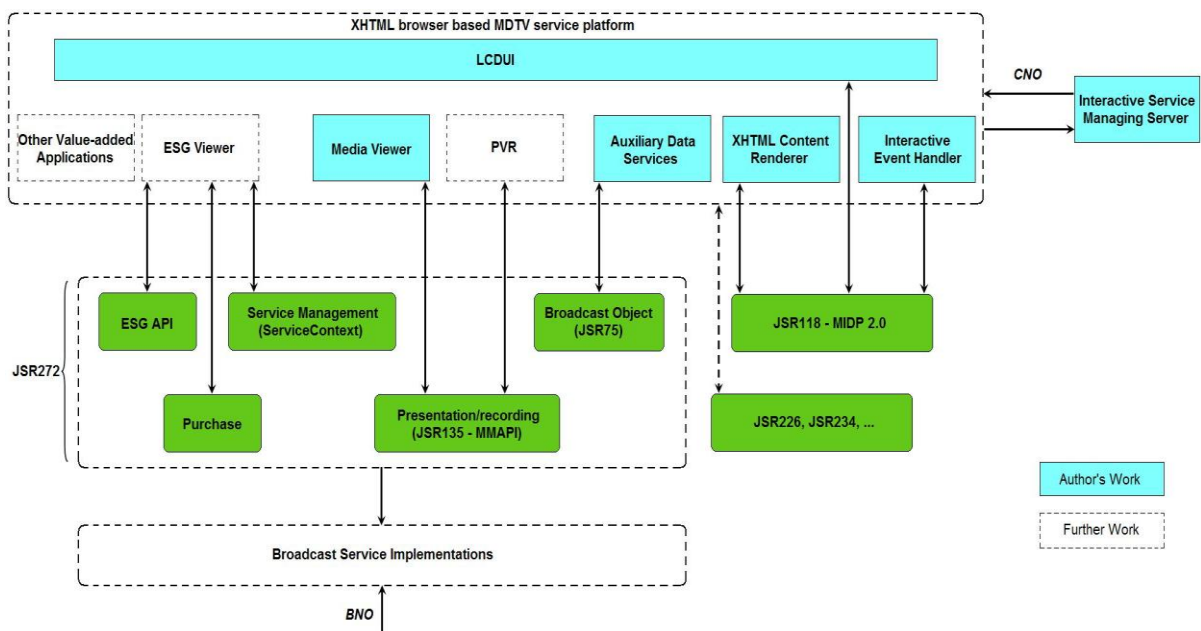


Figure 6.1: MDTV-CIE architecture extended from JSR272 implementation

In order to ensure that the proposed solution is compliant with JSR272's implementation (as shown in Figure 2.6 Chapter 2) properly, the MDTV-CIE architecture (as shown in Figure 4.1) has been further designed and merged with the JSR272 architecture as depicted in Figure 6.1:

Applications such as an ESG viewer, Media Viewer, PVR and Auxiliary Data Services are used to be in a separate form on top of the MIDP + JSR272 platform. Considering the MDTV service as an integration, our proposed XHTML browser based service platform is built as a service managing centre providing ESG retrieval, application execution platform, navigation mechanism, service content parsing, playing and displaying content, and an interaction mechanism. Moreover, the service platform is made to be extensible for additional functions and new value-added applications. Functionally, all the components in the Table below, including LCDUI, Media Viewer, Auxiliary Data Services, XHTML Content Renderer, Interactive Event Handler and Interactive Service Managing Server, are the key functional units in the MDTV-CIE, whereas the other components in the Table below, such as the Value-added Applications, ESG Viewer and PVR can be treated as further work.

Function Units	Relevant JSR Components	Description
LCDUI	JSR118 – MIDP 2.0	GUI and Navigating mechanism for general operations the service contents and applications within the browser based service platform.
Media Viewer	JSR272 Presentation aspect + JSR135 -- MMAPI	MDTV Audio/Video service contents player and controller with the support of MMAPI
Auxiliary Data Services	JSR272 Broadcast Object aspect + JSR75	MDTV services that are developed according to our proposed XHTML based service application model.
XHTML Content Renderer	JSR118 – MIDP 2.0	XHTML based auxiliary service contents renderer for content parsing, visual presenting.
Interactive Event Handler	JSR118 – MIDP 2.0	Catches the interactive service events based on our application model and executes them in cooperation with Server
Interactive Service Managing Server	Java Standard Edition	In cooperation with the interactive event handler to achieve the interactive services.
Other Value-added Applications	Application function dependent	Further work; MDTV Service requirement dependent
ESG Viewer	JSR272 ESG, Purchase and Service Management aspects	Further work while awaiting a public release of JSR272; Main portal for MDTV service with functions of service listing, service purchasing and service management.
PVR	JSR272 Recording aspects	Further work; In charge of MDTV A/V program

Table 6.1: Function Units in the MDTV-CIE

3. Further testing on different operating systems (such as Nokia Symbian, Google Android and Linux) is also considered to be a part of the future work. The purpose of this is to improve the compatibility and portability of the software to different hardware environment (such as mobile phones or set-top boxes).

6.1.3 FURTHER SOFTWARE TESTING PROCEDURE

When all these modification works are conducted to the proposed system and the software components, a series of advanced software testing procedures are planned to be implemented. First an advanced usability test will be carried out to the proposed semi-automatic service creation process. The test will try to select more participants from the practical web service and content design field aiming at evaluating: how the participants accept the proposed creation process; if the semi-automatic creation tool is really effective on helping them achieve the MDTV service creation; if the semi-automatic service creation process is effective on involving design oriented professionals into the MDTV service creation. Second, a comparison testing will be launched between the proposed terminal device service platform and some commercial MDTV client service platform solutions, with the purpose to test the software's functionality and performance more quantitatively and in turn evaluate if the proposed service platform is robust for implementing the MDTV service consumption in practice.

6.2 THESIS SUMMARY

Mobile Digital Television has become one of the future media service trends and meanwhile mobile interactive applications/services have become more important as a key attractive component to the MDTV user. Research institutes as well as commercial entities are developing interactive services to be more functional and to perform in a more efficient way. However, the results have proved to be still far from

promising in some aspects.

With this background situation, the Chapter 1 of the Thesis firstly conducts a brief overview on the Digital TV industry and market focusing on the Mobile Digital TV aspects. The Thesis's aims are then set with regard to cope with the challenges in the current situation. Having discussed the different implementation solutions of the MDTV service as well as the major MDTV standards, Chapter 2 then highlights the application layer of the entire MDTV system, where all the middleware and software specifications are defined to provide standardized technical reference to the actual creation and implementation of the MDTV service applications.

The Thesis then presents deep research on different MDTV standard middleware/software as well as a series of studies on commercial MDTV service solutions. Discussion was then offered on the current MDTV service creation and implementations solutions, which led to an initial conclusion that there is a need of developing a better solution that could cope with the drawbacks of the current solutions. Based on the JSR272 MDTV middleware standard, the Thesis then proposed a universal MDTV implementation solution with better compatibility and adaptability across different underlying network standards. The proposed solution is conducted in form of a MDTV service creation and consumption system, of which two key components were mainly proposed to be implemented in the Thesis: namely the semi-automatic service creation process and MDTV client implementation environment.

Chapter 3 firstly discussed the motivation of proposing the semi-automatic service creation process. Moreover through a research on the Application Model, which is defined both in MDTV standards and in commercial solutions as a service implementation reference, as well as the various MDTV presentation technologies currently applied, the methodology was further worked out. The semi-automatic service creation process was then implemented by developing a semi-automatic MDTV service creation tool with XHTML and Java ME as the basis of the service

presentation technologies. The semi-automatic service creation tool provides a more efficient interactive service creation platform with less technical demands to the designer and further the XHTML and Java ME based presentation method ensure the inter-compatibility and open-source characters of the outputted MDTV service content.

In Chapter 4 the MDTV client implementation environment were presented to assist the implementation of the proposed semi-automatic service creation process. The client side of the environment, namely the terminal device service platform was proposed as the other main software components. Based on the research on various MDTV client-end solutions, the service platform was then decided to be developed based on the browser-based architecture with all the functional components fully customized. A discussion on how such client implementation environment could contribute to the proposed solution was also conducted within the chapter.

A series of testing processes have been conducted and the evaluations of the results have also been presented in chapter 5 of the Thesis. As a result, both of the proposed software components have performed well in all tests with very positive feedback from the target audience and through an overall discussion, the proposed MDTV service creation and consumption system was further verified to meet the aims of the Thesis. Finally in this chapter, based on the testing results, the Thesis presents future and further work that will further enhance the proposed system and will advance the research in this area. Lastly the summary and the conclusion end this Thesis.

6.3 LESSONS LEARNED & CONCLUSION

From the research work in this Thesis, it can be understood more clearly that there is need of the high quality of multimedia and interactive applications in the current MDTV service market although the current service creation and implementation solutions could not satisfy this situation. The current MDTV service implementation solutions suffer from several problems: firstly, the current MDTV standards have

utilized different presentation technologies during their service creation, resulting in their service contents not being inter-compatible with each other. This has caused the creation of a same service/application for different bearer standards. Secondly, there is lack of open-standard MDTV service implementation solutions that most of the current ones are developed under the authority of commercial entities (e.g. Streamezzo and EXPWAY). Their proprietary character has resulted in the isolated service implementation environment, in which the service creation and maintenance are relatively hard and expensive. Thirdly, the Rich Media and ECMAScript based presentation technologies, such as OMA-RME and MPEG-LASeR, have dominated the current MDTV service creation but there are still several drawbacks when using them in practice. For example, the Rich Media based MDTV service GUI is not adaptable to mobile devices with different screen sizes and resolutions once created. The Rich Media technologies usually requires the use of specified/customized authoring tools during the service creation and thus advanced technical knowledge is demanded from the designers. As a result, all of these findings have motivated the proposal of the new MDTV service creation and consumption solution as presented in the Thesis.

Through the development of the proposed solution, we have learned that to reduce the technical demands on the designer is an effective way of encouraging the development of the MDTV services in practice. Also the introduction of open-source and inter-compatibility characters can further increase the efficiency and meanwhile reduce the cost and efforts during the service creation process. Moreover the Rich Media and ECMAScript has been proved not only the effective group for presenting the MDTV service that the proposed XHTML and Java ME based method has also got the ability of implementing most of the popular MDTV services and applications.

Based on all these lessons learned, the Thesis has been able to propose the solution and achieve the contributions.

The Thesis has proposed a universal MDTV service creation and implementation

solution to cope with the problems and defects in the current MDTV industry field. Such solution is implemented based on a novel MDTV service creation and consumption system, of which the semi-automatic service creation process and the client implementation environment are the main sub-systems.

The semi-automatic service creation process conducts the contribution by facilitating the current MDTV service creation and reducing the technical demands during the creation. The core software components, namely the semi-automatic service creation tool, is developed to assist the designer creating the MDTV service page semi-automatically and by using this proposed service creation tool, the designer do not have to know well on the MDTV service creation technologies and skills to achieve the work. Moreover the XHTML and Java ME based service presentation method ensures the service contents' inter-compatibility through various MDTV standards. It also helps preventing the proprietary character during the service creation process and further enables the involvement of more design oriented professionals into the MDTV service development.

In the client implementation environment of the proposed system, the terminal device service platform has been the core contributing component. It is developed based on the browser-based architecture aiming at providing service consumption client side platform for rendering and handling the MDTV service content outputted from the proposed creation tool. In this way the functionality of the proposed MDTV service creation and consumption system are fully achieved. Also, the service platform achieves in implementing most of the popular MDTV service applications as the current commercial solution could provide. This further supports the XHTML and Java ME based MDTV service presentation method to be an alternative choice other than the Rich Media and ECMAscript based methods.

Further through a series of the software testing procedure, the two software components in the solution have been verified to meet their design requirements and validated to be capable of achieving their functionalities in the proposed system. All

the aims of the thesis' are tested to be met. Additionally, the test results and the test participants' comments and advices also motivate the plan of the further modification and enhancement work to the software and the proposed system in the future.

As conclusion, this Thesis has proposed a new MDTV service creation and implementation solution. With the hope of implementing the further work, this project represents an advanced research and exploration in the MDTV service creation field.

Summary:

This chapter presents the future work to the proposed solution and software components. The summary provides the review of the Thesis and the conclusion states the lesson learned through the research work and finally highlights the contribution of the Thesis.

V. References

- [1] Barry Fox, "Digital TV comes down to earth", *IEEE Spectrum*, pp23-29, October 1998.
- [2] Artur Lugmayr et al, *Digital Interactive TV and Metadata: Future Broadcast Multimedia*, New York: Springer-Verlay, 2004.
- [3] Dr Ulrich Reimers, *DVB: The Family of International Standards for Digital Video Broadcasting, Second Edition*, Germany: Springer, 2005.
- [4] Margherita Pagani, *Multimedia and Interactive Digital TV: Managing the Opportunities Created by Digital Convergence*, UK and USA: IRM Press, 2003.
- [5] Amitabh Kumar, *Mobile TV: DVB-H, DMB, 3G Systems and Rich Media Applications*, Elsevier Inc, 2007.
- [6] Van den Broeck, W. & Pierson, J., *Digital Television in Europe*, Brussels: VUBpress, 2008.
- [7] Thomas Crampton. (1998, July 29). Why Japan and Korea Dominate in Mobile TV. [Online]. Available: <http://www.thomascrampton.com/media/why-japan-and-korea-dominate-in-mobile-tv/>.
- [8] Emmanuel Tseklevs, "Converged Digital TV Services: The Role of Middleware and Future Directions of Interactive Television", *International Journal of Digital Multimedia Broadcasting*, Volume 2009, pp19-41, 2009.
- [9] Winning in Italy. [Online]. Available at: http://www.gemalto.com/mobile_tv/3g_italy.html. Last visit on: 31/7/2010.
- [10] La 3, [Online]. Available: <http://www.la3tv.it/>. Last visit on: 31/7/2010.
- [11] Vodafone. [Online]. Available: <http://www.Vodafone.co.uk/>. Last visit on: 31/7/2010.
- [12] MobiTV. [Online]. Available: <http://www.mobitv.com/>. Last visit on: 31/7/2010.
- [13] AT&T, [Online]. Available: <http://www.att.com/>. Last visit on: 31/7/2010.
- [14] Qualcomm. [Online]. Available: <http://www.qualcomm.com/>. Last visit on: 31/7/2010.
- [15] MediaFLO. [Online]. Available: <http://www.mediaflo.com/>. Last visit on: 31/7/2010.
- [16] DVB-H. [Online]. Available: <http://www.dvb-h.org/>. Last visit on: 31/7/2010.
- [17] AT&T Selects Qualcomm's MediaFLO USA for Mobile Entertainment Services. [Online]. Available: http://www.qualcomm.com/news/releases/2007/070212_att_selects_s.html. Last visit on: 20/8/2009.
- [18] KEVIN J. O'BRIEN. (2008, May 6). Mobile TV Spreading in Europe and to the U.S.. *NYTimes.com via Yahoo! Finance*. [Online]. Available: <http://biz.yahoo.com/nytimes/080506/1194771946810.html?.v=18>.
- [19] Qualcomm. Creating a Mobile Broadcast Platform: The Media FLO Product Technology Overview. [Online]. Available: <http://www.mediaflo.com/mediaflo/index.html>. Last visit on: 12/10/2009
- [20] Frank Hartung, et al, "Delivery of Broadcast Services in 3G Networks", *IEEE Transaction on Broadcasting*, vol.53, NO.1, March 2007.
- [21] MobiTV. ATSC-M/H: The Promise of Free to Air Mobile Simulcast. [Online]. Available: <http://www.mobitv.com/>. Last visit on: 12/10/2009.
- [22] Guardian. [Online]. Available: <http://www.guardian.co.uk/technology/2007/jan/17/news.media>. Last visit on: 12/10/2009.
- [23] Simon Mason, "Mobile TV – results from the DVB-H trial in Oxford", EBU Technical

Review, April 2006.

- [24] <http://www.reportlinker.com/p0119504/Global-Mobile-TV-Forecast-to.html>, Report Summary
- [25] EICTA. (2008 January). HD TV & HD ready Logos. [Online]. Available: http://www.digitaleurope.org/fileadmin/user_upload/document/EICTA_-_HD_factsheet_logos.pdf. Last visit on: 31/7/2010.
- [26] DVB Project. (2010 May). Introduction to the DVB Project, DVB Fact Sheet. [Online]. Available: http://www.dvb.org/technology/fact_sheets/. Last visit on: 31/7/2010.
- [27] Keith Jack, *Video Demystified: A Handbook for the Digital Engineer, Fifth Edition*, Elsevier Inc, 2007.
- [28] *Part1 – ATSC Mobile Digital Television System, Part 4 – Announcement, Part 5 – Application Framework, Part 6 – Service Protection, Advanced Television Systems Committee, ATSC-Mobile DTV Standard A/153*, 2009.10.
- [29] *Data Coding and Transmission Specification for Digital Broadcasting, Association of Radio Industries and Businesses, ARIB STD-B24 v5.2 English Translation*, 2008.
- [30] W Liang, et al, “Digital Terrestrial Television Broadcasting in China”, *IEEE Multimedia*, Volume 14, pp92-97, August 2007.
- [31] *ABS-S White Paper*, The State Administration of Radio Film and Television (SARFT), China, 2008.
- [32] *DMB-TH White Paper*, Beijing Lingxunhuaye Tech. Ltd. Tsinghua University, China, 2006.
- [33] Shanghai Jiaotong University. Advanced Digital Television Broadcasting – Terrestrial (ADTB-T). [Online]. Available: <http://www.ratiog.org/ADTB-T.htm>. Last visit on: 31/7/2010.
- [34] Y Shi, et al, “A New Generation Satellite Broadcasting System in China: Advanced Broadcasting System-Satellite”, in *4th IEEE International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM'08)*, pp1-4.
- [35] Jian Chi, Xiong-jian Liang, “The Building of Mobile Phone TV Standard in China”, in *IEEE International Conference on Management and Service Science*, 2009, pp1-3.
- [36] World Cellular Information Service. Subscriptions by Technology. [Online]. Available: http://www.wcisdata.com/newt1/wcis/research/subscriptions_by_technology.html. Last visit on: 8/7/2010.
- [37] *The evolution of EDGE: White Paper*, Ericsson, September 2008, available: http://www.ericsson.com/news/090901_evolution_of_edge_20100510174715?query=edge, last visit on 8/7/2010.
- [38] International telecommunication Union. What really is a Third Generation (3G) Mobile Technology. [Online]. Available: http://www.itu.int/ITU-D/imt-2000/Revised_JV/IntroducingIMT_item3.html. Last visit on: 8/7/2010.
- [39] *Universal Mobile Telecommunications System (UMTS); Vocabulary for 3GPP Specifications*, Europe Telecommunications Standards Institute Standard ETSI TR 121 905 v3.30 (2001-10), 2001.
- [40] *What is HSDPA? White Paper*, SAMSUNG, available: www.samsung.com/uk/business/b2b/pdfs/case_studies/hsdpa.pdf, last visit on 8/7/2010.
- [41] Moray Rumney, “Introducing Single-Carrier FDMA”, *Agilent Technologies*, 2008,

- available: <http://cp.literature.agilent.com/litweb/pdf/5989-7898EN.pdf>, last visit on 8/7/2010.
- [42] The 3rd Generation Partnership Project. IP-Multimedia Subsystem (IMS). [Online]. Available: <http://www.3gpp.org/article/ims>. Last visit on: 8/7/2010.
- [43] The 3rd Generation Partnership Project. About 3GPP, 3GPP Scope. [Online]. Available: <http://www.3gpp.org/About-3GPP>. Last visit on: 8/7/2010.
- [44] Gilles Bertrand. (2007, May 30). The IP Multimedia Subsystem in Next Generation Networks. [Online]. Available: http://www.rennes.enst-bretagne.fr/~gbertran/files/IMS_an_overview.pdf. Last visit on 8/7/2010.
- [45] *Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Service requirements for the Internet Protocol (IP) multimedia core network subsystem (IMS); Stage 1*, European Telecommunications Standards Institute standard ETSI TS 122 228 v9.2.0 (2010-01), 2010.
- [46] Third Generation Partnership Project 2. (2007). Third Generation Partnership Project Agreement for 3GPP2. [Online]. Available: http://www.3gpp2.org/Public_html/Misc/ProceduresHome.cfm. Last visit on 9/7/2010.
- [47] *Broadcasting to Handhelds, Digital Terrestrial Mobile TV. DVB Fact Sheet – April 2009*, Digital Video Broadcasting Project, 2009.
- [48] P. Leroux, et al, “Synchronized Interactive Services for Mobile Devices over IPDC/DVB-H and UMTS”, *2nd IEEE/IFIP International Workshop on Broadband Convergence Network, 2007*, pp1-12.
- [49] Jo Guejo. Digital Multimedia Broadcasting. [Online]. Available: http://www.itu.int/ITU-D/tech/digital-broadcasting/MiscMeetings/ITU_AIBD_Pakistan_Nov07_ITUPresentation.pdf. Last visit on: 9/7/2010.
- [50] The 3rd Generation Partnership Project 2. About 3GPP2. [Online]. Available: http://www.3gpp2.org/Public_html/Misc/AboutHome.cfm. Last visit on: 9/7/2010.
- [51] K. Kang, H. Shin, “Modeling the Energy Consumption of Reed-Solomon Decoding with Interleaving on Fading Channels”, *IEEE Transactions on Wireless Communications*, Vol. 7, No. 11, pp4100-4104, November 2008.
- [52] J. Lee, et al, “OpenCA: Conditional Access System in MediaFLO”, in *2008 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*, 2008, pp1-6.
- [53] Qualcomm. FLO Technology Overview. [Online]. Available: <http://www.mediaflo.com/mediaflo/index.html>. Last visit on: 10/7/2010.
- [54] *Rich Media Environment Architecture Candidate Version 1.0*, Open Mobile Alliance standard OMA-AD-Rich_Media_Environment-V1_0-2008014-C; Open Mobile Alliance Ltd. Standard, 2008.
- [55] Fabio Allamandri et al, “Service Platform for Converged Interactive Broadband Broadcast and Cellular Wireless”, *IEEE Transaction on broadcasting*, Vol.53, No.1, pp200-211, March 2007.
- [56] ACM SIGCHI. (2009, July 29). Curricular of Human-Computer Interaction, Chapter 2, 2.1: Definition of HCI. [Online]. Available: <http://old.sigchi.org/cdg/cdg2.html>.
- [57] Ha Yoon Song, Jongwook Park, “Design of an Interoperable Middleware Architecture for Digital Data Broadcasting”, *IEEE Transaction on Consumer Electronics*, Volume 52, pp1433-

1441, Nov.2006.

[58] *Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.2.2*, DVB Project Document A140, June 2009.

[59] DVB Project. (2009 July). "DVB MHP Fact Sheet Multimedia Home Platform – Open Middleware for Interactive TV. [Online]. Available: <http://www.mhp.org/>.

[60] Jon Piesing, "The DVB Multimedia Home Platform (MHP) and Related Specifications", in *Proceeding of the IEEE*, 2006, vol.94, No.1, pp237-247.

[61] DVB MHP. DVB-HTML Questions and Answers. [Online]. Available: <http://www.mhp.org/>. Last visit on: 31/7/2010.

[62] *Digital Video Broadcasting (DVB); Globally Executable MHP (GEM) Specification 1.2.2 (including IPTV)*, DVB Project Document A139 r4, June 2009.

[63] DVB Project. (2009 July). DVB Fact Sheet Globally Executable MHP (GEM): DVB's Open Middleware for Interactive Applications. [Online]. Available: <http://www.mhp.org/>.

[64] *OpenCable Application Platform Specifications: OCAP 1.1 Profile*, Cable Television Laboratories (OpenLabs) standard OC-SP-OCAP 1.1.2-090930, September 2009.

[65] *ATSC Standard: Advanced Common Application Platform (ACAP)*, Advanced Television Systems Committee Inc. ATSC Document A/101A, February 2009.

[66] *DTV Application Software Environment Level 1 (DASE-1) Part 1: Introduction, Architecture, and Common Facilities*, ATSC Document A/100-1, March 2003.

[67] ARIB STD-B23 v1.1 English Translation (2004), Application Execution Engine Platform for Digital Broadcasting, Association of Radio Industries and Businesses, available online: <http://www.arib.or.jp/english/index.html/>.

[68] AKIHIRO HORI et al (2006), "Japanese Datacasting Coding Scheme BML", in *Proceedings of The IEEE*, Vol. 94, No.1, 1, pp312-317, January 2006.

[69] Java Community Process. (2006). Java 118 Expert Group Mobile Information Device Profile for Java 2 Micro Edition Version 2.0. [Online]. Available: <http://jcp.org/aboutJava/communityprocess/mrel/jsr118/index.html>.

[70] Java - Sun Microsystems. Mobile Information Device Profile (MIDP); JSR137, JSR118 Overview. [Online]. Available: <http://java.sun.com/products/midp/overview.html>. Last visit on: 31/7/2010.

[71] Java - Sun Microsystems. Mobile Information Device Profile Data Sheet. [Online]. Available: <http://java.sun.com/javame/overview/techpapers/index.jsp>. Last visit on: 31/7/2010.

[72] Java - Sun Microsystems. Digital Television for the Java ME Platform Mobile Devices: Introduction to JSR272. [Online]. Available: <http://developers.sun.com/learning/javaoneonline/j1sessn.jsp?sessn=TS-4693&yr=2006&track=mobility>. Last visit on: 6/12/2009.

[73] Ivan Wong et al, "Digital Television for the Java ME Platform Mobile Devices: Introduction to JSR272", in *JavaOne Conference*, Session TS-4693, 2006.

[74] *Mobile Broadcast Service API for Handheld Terminals (JSR272) JCP Specification*, Java Community Process standard, available: <http://www.jcp.org/en/jsr/summary?id=272>.

[75] *Digital Audio Broadcasting (DAB); Middleware; Part1: System aspects*, European Broadcasting Union ETSI standard ETSI TS 102 365 – 1 v1.1.1, 8/2009.

[76] *Digital Audio Broadcasting (DAB); Middleware; Part 2: DAB*, European Broadcasting

- Union ETSI standard ETSI TS 102 365 – 2 v1.1.1, 8/2009.
- [77] Nokia Mobile TV Forum. (2007 Feb). Nokia Mobile Broadcast Solution Release 3.2., [Online]. Available: <http://www.mobiletv.nokia.com/solutions/>. Last visit on: 6/12/2009.
- [78] Nokia Mobile TV Forum. Air Interface Implementation Guidelines. [Online]. Available: <http://www.mobiletv.nokia.com/solutions/airif/>. Last visit on: 6/12/2009.
- [79] Nokia. (2007). Nokia DVB-H Mobile TV Implementation Guidelines, Release 3.0, Air Interface. [Online]. Available: http://www.mobiletv.nokia.com/solutions/airif/mtvig_documents_3-0.php. Last visit on: 7/12/2009.
- [80] EXPWAY. Mobile TV products. [Online]. Available: <http://www.expway.com/mobile-tv.php>. Last visit on: 7/12/2009.
- [81] Silicon & Software Systems. onHandTV Production Overview. [Online]. Available: <http://www.s3group.com/tv-technology/tv-products/onhandtv/>. Last visit on: 7/12/2009.
- [82] Thin Multimedia, Inc (tmi). thinMobileTV Middleware. [Online]. Available: http://www.thinmultimedia.com/products/thinmobile_tv/. Last visit on: 7/12/2009.
- [83] *Digital Video Broadcasting (DVB); IP Datacasting over DVB-H: Use Cases and Services*, European Broadcasting Union ETSI standard ETSI TR 102 473 v1.1.1, 2006.
- [84] Ian Sommerville, *Software Engineering Seventh Edition*, Pearson Education Limited, International edition, 2004.
- [85] *Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Architecture*, European Broadcasting Union ETSI standard ETSI TR 102 469 v1.1.1, 2006.
- [86] *Universal Mobile Telecommunications System (UMTS); LTE; Dynamic and Interactive Multimedia Scenes (DIMS) (3GPP TS 26.142 version 9.0.0 Release 9)*, European Broadcasting Union ETSI standard, ETSI TS 126 142 v9.0.0, 2009.
- [87] TATA ELXSI Engineering Creativity. Tata Elxsi Product Solution, DVB-CMBS stack. [Online]. Available: http://www.tataelxsi.com/htmls/pds/pds_mobile_TV.htm. Last visit on: 15/12/2009.
- [88] Thin Multimedia, Inc (tmi). Mobile TV Middleware: thinDVB-H. [Online]. Available: http://www.thinmultimedia.com/products/thinmobile_tv/. Last visit on: 7/12/2009
- [89] Stephan Skrodzki (GMIT). (2006). Handheld interactive synchronized TV – HisTV version 1.3. [Online]. Available: <http://www.histv.org/>. Last visit on: 7/12/2009.
- [90] ACCESS. (2008). NetFront Browser DTV Profile DVB-H Edition: Deliver an Innovative and Consistent Mobile DTV Experience Across Multiple Mobile Platform. [Online]. Available: http://www.access-company.com/products/mobile_solutions/mobile_tv/dvb-h_profile.html. Last visit on: 15/12/2009.
- [91] Streamazzo. (2008). Streamazzo Interactive Mobile TV, Streamazzo rich media technologies. [Online]. Available: <http://www.streamazzo.com/eng/solutions/mobile-tv.php>. Last visit on: 15/12/2009.
- [92] Alfred Baier, et al, (Vodafone), “Mobile TV – From pure Broadcast to Interactivity”, in *Workshop on Multiradio Multimedia Communication (MM '06) “Interactive TV for Handheld Devices”*, Berlin, 2006.
- [93] Dr. Jorg Heur, Siemens, “What is on top of DVB-H? An intro to DVB IPDC and Middleware concepts for interactive applications”, Siemens Corporate Technology.
- [94] OpenTV. OpenTV Worldwide Customers. [Online]. Available at:

<http://www.opentv.com/customers/customers.htm>. Last visit on 14/7/2010.

- [95] A. Younus, et al, “Dynamic Interactive Multimedia Scenes in Mobile Broadcast Environments”, in *IEEE International Conference on Communication*, pp1833-1838, 2007.
- [96] *Rich Media Environment Technical Specification Candidate Version 1.0 – 14 Oct 2008*, OMA-TS-RME-V1_0-20081014-C, Open Mobile Alliance standard, 2008.
- [97] *White Paper on Rich Media Environment Technology Landscape Report Candidate – 14 Oct 2008*, OMA-WP-Rich_Media_Environment-20081014-C, Open Mobile Alliance standard, 2008.
- [98] H. Song, J Park, “Design of an Interoperable Middleware Architecture for Digital Data Broadcasting”, *IEEE Transactions on Consumer Electronic*, Volume 52, pp1433-1441, 2006.
- [99] M.M.Saleemi, et al, “System Architecture and Interactivity Model for Mobile TV Applications”, in *3rd International Conference on Digital Interactive Media in Entertainment and Arts*, 2008.
- [100] *Enabler Release Definition for Rich Media Environment Candidate Version 1.0 – 14 Oct 2008*, Open Mobile Alliance Ltd standard, 2008.
- [101] *ECMAScript Mobile Profile: A Wireless Markup Scripting Language Approved version 1.0 – 20 Oct 2006*, Open Mobile Alliance Ltd standard, 2006.
- [102] M. Liu, et al, “Semi-automatic creation of graphically-rich mobile Television services and applications using an XHTML browser and J2ME”, in *2010 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*, pp1-7, 2010.
- [103] R. L. Cody, J. Cosmas, E. Tseklevs, “Open-standards Rich Media Mobile Platform & Rapid Service Creation Tool”, in *Global Mobile Congress 2009*, pp1-6, Oct.2009.
- [104] M. Spika, “An XML-based Software Platform for DVB-H and IP Datacast with Full Java-logic Capability”, in *International Symposium on Consumer Electronic (ISCE 2008)*, pp1-4, April 2008.
- [105] B. Lee, J. Song, Y. Nam, “Converged Mobile TV Services Supporting Rich Media in Cellular and DVB-H Systems”, *IEEE Transactions on Consumer Electronic*, Vol. 54, Issue 3, pp1091-1097, August2008.
- [106] J. Dufourd, et al, “LASeR: the MPEG Standard for Rich Media Services”, *IEEE Multimedia*.
- [107] Adobe. Getting started with Flash Lite 2.x and 3.0. [Online]. Available: <http://www.adobe.com/support/documentation/en/flashlite/>. Last visit on 21/7/2010.
- [108] Wikipedia. Soccer ball animated.svg. [Online]. Available: http://en.wikipedia.org/wiki/File:Soccer_ball_animated.svg. Last visit on: 21/7/2010.
- [109] J. Dufourd, “LASeR: The Lightweight Rich Media Representation Standard”, *IEEE Signal Processing Magazine*, 2008.
- [110] INSTINCT. (2004). IP-based Networks, Services and Terminals for Convergence Systems. [Online]. Available: <http://www.ist-instinct.org/>.
- [111] R. Schatz, S. Wagner, A. Berger, “AMUSE – A Platform for Prototyping Live Mobile TV Services”, in *Mobile and Wireless Communication Summit 2007* 16th IST, pp1-5, July 2007.
- [112] P. Leroux, et al, “UIML Based Design of Multimodel Interactive Applications with Strict Synchronization Requirements”, in *2009 Second International Conferences on Advances in Computer-Human Interactions*, pp175-180, 2009.

- [113] W3C Recommendation. (2008, December 22). Scalable Vector Graphics (SVG) Tiny 1.2 Specification. [Online]. Available: <http://www.w3.org/TR/SVGTiny12/>. Last visit on: 8/1/2010.
- [114] Adobe. Flash Lite. [Online]. Available: <http://www.adobe.com/products/flashlite/>. Last visit on: 21/7/2010.
- [115] W3C Recommendation. (2008, December 1). Synchronized Multimedia Integration Language (SMIL 3.0). [Online]. Available: <http://www.w3.org/TR/2008/REC-SMIL3-20081201/>. Last visit on: 21/7/2010.
- [116] W3C Recommendation. (2008, July 29). XHTML Basic 1.1. [Online]. Available: <http://www.w3.org/TR/2008/REC-xhtml-basic-20080729/>. Last visit on: 21/7/2010.
- [117] W3C. XML Essentials. [Online]. Available: <http://www.w3.org/standards/xml/core>. Last visit on: 21/7/2010.
- [118] W3Schools. Introduction of XML. [Online]. Available: http://www.w3schools.com/xml/xml_what.asp. Last visit on: 21/7/2010.
- [119] Emmanuel Marilly, et al, “Rich Media Service Creation for Interactive Mobile TV”, in *2007 IEEE International Conference on Next Generation Mobile Application, Service and Technologies (NGMAST 2007)*, pp85-90, 2007.
- [120] Hyun-Cheol Kim, et al (Broadcasting & Telecommunications Media Research Department, ETRI), “Development of Personalized DMB Services and System”, in *11th International Conference on Advanced Communication Technology 2009 (ICTACT 2009)*, pp2023-2026, 2009.
- [121] WebKit. The WebKit Open Source Project. [Online]. Available: <http://webkit.org/>.
- [122] IEBlog. The Internet Explorer 8 User-Agent String (Updated Edition). [Online]. Available: <http://blogs.msdn.com/ie/archive/2009/01/09/the-internet-explorer-8-user-agent-string-updated-edition.aspx>. Last visit on: 21/7/2010.
- [123] Dev.Opera. Opera Presto 2.1 – Web standards supported by Opera’s core. [Online]. Available: <http://dev.opera.com/articles/view/presto-2-1-web-standards-supported-by/>.
- [124] Mozilla. Gecko. [Online]. Available: <https://developer.mozilla.org/en/Gecko>.
- [125] Jonathan Knudsen. (2002). Parsing XML in J2ME, Sun Developer Network. [Online]. Available: <http://developers.sun.com/mobility/midp/articles/parsingxml/>.
- [126] KObjects.net. About kXML. [Online]. Available: <http://kobjects.org/kxml/index.php>.
- [127] Roger S. Pressman, *Software Engineering: A Practitioner’s Approach, Six Edition*, McGraw-Hill Education, 2005.
- [128] Shari L. Pfleeger, Joanne M. Atlee, *Software Engineering: Theory and Practice, Third Edition*, Pearson Education, 2006.
- [129] Atif M. Memon, “GUI Testing: pitfalls and process”, *IEEE Computer*, pp87-88, August 2002.
- [130] A. Memon, I. Banerjee, and A. Nagarajan, “GUI Ripping: Reverse Engineering of Graphical User Interface for Testing”, in *Proceedings of the 10th Working Conference on Reverse Engineering (WCRE’03)*, pp260-269, 2003.
- [131] Glenford J. Myers, *The ART of SOFTWARE TESTING, Second Edition*, John Wiley & Sons, Inc, 2004.
- [132] E. Tseklevs, et al, “Semi-automated Creation of Converged iTV Services: From Macromedia Director Simulations to Services Ready for Broadcast”, *Journal of Virtual Reality and Broadcasting*, Volume 4, no.17, 2007.

VI. Appendix 1: Consent Form

Letter of Information

User testing of a Mobile Digital Television service software system

What is our study about?

This project is basically in the field of Mobile Digital TV (MDTV), where the Digital TV program including audio/visual and auxiliary data services are provided through mobile devices (mobile phone, PDA and etc.) to the end user. Regarding to the low efficiency and incompatibility of the current MDTV service creation and implementation process, this project has proposed a software system aiming at improving the creation process. In order to test the software system in a comprehensive manner, we decided to organize a testing case with actual users involved.

What will the participation involve?

If you are willing to participate, you will be asked to do a set of tasks by using our proposed software and complete a relevant questionnaire at the end of the tasks. The whole process should take approximately 20 minutes of your time. You may refuse to participate, refuse to answer any question, or withdraw from the study at any time with no penalty. By participating in this study, you are also agreeing that your results may be used for scientific purposes, including publication in scientific journals, as long as your anonymity is maintained. There are no known risks associated with participation in this research.

If you have any questions now or after the study, please do not hesitate to contact us:

Thank you,

Moxian Liu
Email: Moxian.Liu@brunel.ac.uk
Office Address: 258, Michael Sterling, Brunel University.

This letter is for you to keep

Consent Form

I _____ (please print) have read the attached information sheet and discussed the investigation with **Moxian Liu** who has explained the procedures to my satisfaction. I am writing to undergo the investigation, and understand that I am free to withdraw at any time without having to give an explanation and that doing so will not affect any treatment I may receive.

Signed

Date

Witness's Name

Witness's Signature

VII. Appendix 2: Participant Document

User Task Testing

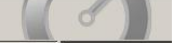
1. Creation of MDTV interactive service pages by using the proposed semi-automatic service creation tool.

Task tutorial: how to create an interactive service page using the semi-automatic service creation tool.

Step1: feel free to try the GUI components of the software (left-click or right-click the mouse on the menus, buttons, windows, etc.);

Step2: double-click the testpage5 file name in the file list, in the left frame to open the target XHTML service page (testpage5);

Step3: add the “text” element under the first “<p> ... </p>” section in the source code of the target service page displayed in the lower frame on the central pane to edit the page;

Step4: press the  hotkey on the hotkey bar to change the central pane to “element mode”;

Step5: select an interactive application from the interactive application list in the right pane;

Step6: drag that application from the list and drop on the target element (id = “Element 1”) in the source code display area;

Step7: follow the pop-up message and enter the relevant parameter (**ask the tester for parameters**);

Step8: repeat Step6 and Step7 to add different interactive applications to Element 2 (id = “Element 2”) – Element 6 (id = “Element 6”);

Step9: press the “Element Mode” hotkey on the hotkey bar again to change the central pane from “Element Mode” back to “Text Mode”;

Step10: press “save XHTML file...” hotkey on the hotkey bar to save all the modification to the service page;

Step11: press the “X” figure on the tab in the central pane to close the target XHTML service page;

Step12: check if the output file – IDEventFactory.java is on the desktop.


2. Browse the MDTV interactive service pages through proposed terminal service platform.

Task tutorial: how to browse a service page in the terminal platform.

(Please use the mobile phone emulator keyboards for the following operation)


Step1: feel free to try out the GUI components (menu, options and etc.);




Step2: press “menu” and select “homepage” to start browsing the service pages;




Step3: press  on the service page to move on; when the new service page is displayed on the screen, press the navigation button on the emulator keyboard to navigate within one service page;

Step4: press “OK” button on the emulator keyboard when the green navigating rectangle is on any hyperlink component/image on the service page so as to navigate between multiple service pages;

Step5: feel free to select the “PrePage” option in the menu to go back to the previous page;

Step6: go to testpage7 (refer to Figure 1), press  to start playing the video program;

press  to stop the video; press  to vote if you like the program; press  to vote if you do not like it;

Step7: press  to go to testpage61 (refer to Figure 1), press  to request the vote result report; go back to testpage61, press  to request for live weather data feed;

Step8: go to “menu – URLInput” to start browsing the service page by typing in the URL (URL=”http://127.0.0.1/remote/testpage6.html”); press “menu – URLget” to load to page.

Step10: select “Exit” option to exit the service platform.

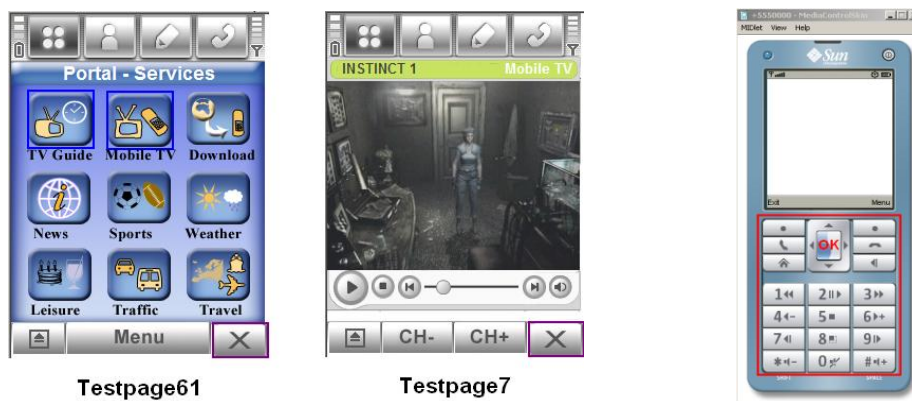


Figure 1

Acceptance Testing Questionnaire

User Profile

Age: 18 – 25, 26 – 35, 36 – 45, 46 – 55, 56 – 65.

Gender: Male / Female

Occupation:

Degree primary subject:

Phase 1: Semi-automatic service creation tool

1. Do you have any experience on software programming? Yes/No
2. Do you have any experience on HTML/XHTML authoring? Yes/No
3. Do you have any experience on graphic/user interface design? Yes/No
4. How useful was the software in assisting you to accomplish the given task? Not useful 1 2 3 4 5 Very useful
5. How easy or difficult was it for you for you to get familiarised with the Graphical User Interface (GUI) of the software? Very Difficult 1 2 3 4 5 Very Easy
6. How easy or difficult was it for you to use the GUI of the software? Very Difficult 1 2 3 4 5 Very Easy
7. How do you rate the GUI design? Very Poor 1 2 3 4 5 Very Good
8. How easy or difficult was it for you to manipulate the service page by editing the source code? Very Difficult 1 2 3 4 5 Very Easy
9. How easy or difficult was it for you to assign a function/application to a User Interface graphic/component? Very Difficult 1 2 3 4 5 Very Easy
10. How easy or difficult was it for you to use the software? Very Difficult 1 2 3 4 5 Very Easy
11. How do you rate the software overall? Not useful 1 2 3 4 5 Very useful
12. How effective is the software in assisting you to accomplishing the given task? Not effective 1 2 3 4 5 Very effective
13. Do you have any suggestion on improving the functionality of the software?

14. Do you have any suggestion on improving the GUI of the software?

Phase 2: Terminal service platform

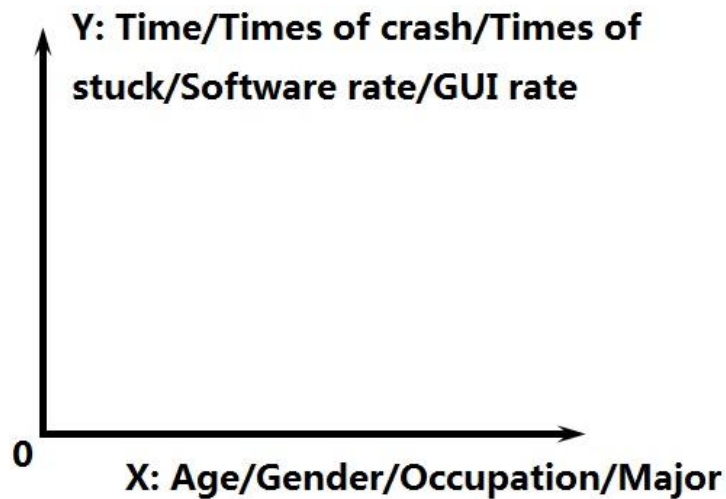
15. How useful was the software in assisting you to accomplish the given task? Not useful 1 2 3 4 5 Very useful
16. How did the software perform while you were doing the task? Very Poor 1 2 3 4 5 Very Good
17. How easy or difficult was it for you to get familiarised with the GUI of the software? Very Difficult 1 2 3 4 5 Very Easy
18. How easy or difficult was it for you to use the GUI of the software? Very Difficult 1 2 3 4 5 Very Easy
19. How do you rate the GUI design? Very Poor 1 2 3 4 5 Very Good
20. How would you rate the rendering of the MDTV service page elements (including audio/visual elements) in terms of speed? Very Poor 1 2 3 4 5 Very Good
21. How would you rate the navigation within the same page? Very Poor 1 2 3 4 5 Very Good
22. How would you rate the navigation between different service pages? Very Poor 1 2 3 4 5 Very Good
23. How did the interactive applications perform overall? Very Poor 1 2 3 4 5 Very Good
24. How do you rate the software overall? Very Poor 1 2 3 4 5 Very Good
25. Would you like to be able to interact with mobile TV applications through this or a similar platform? Yes / No / Don't know
26. Do you have any suggestion on improving the software? (functionality or GUI)
27. What other interactive applications do you expect from a Mobile TV service?

Thank you very much for your patience and feedback!

Alpha Testing Process Observation Record by Tester

	Task 1	Task 2
How many times did the user pause during the task due to difficulty on understanding the task?		
How many times did the user pause during the task due difficulty with the GUI of the software?		
How many times did the user pause during the task due to the software crashing?		
How many errors did the user make while using the software?		
Could the user finish the tasks?	Yes/No	Yes/No
How long did the user take on each task?	mins	mins
Do the outputs of the user tasks meet the software requirements?	Yes/No	Yes/No

Curves sample:



VIII. Appendix 3: Acceptance Test Result

Acceptance Testing Questionnaire Phase 1										
Phase 1	Participants (p)									
	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10
1	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
2	No	Yes	No	No	Yes	Yes	Yes	Yes	No	No
3	No	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No
4 (Rank)	3	5	5	4	4	5	5	4	4	5
5 (Rank)	2	5	5	4	5	5	4	4	5	4
6 (Rank)	4	5	5	4	5	5	4	4	4	5
7 (Rank)	4	4	5	4	5	4	4	5	4	5
8 (Rank)	2	4	5	5	5	5	5	4	3	2
9 (Rank)	2	3	5	3	5	5	5	4	4	4
10 (Rank)	3	4	5	4	4	5	5	5	3	4
11 (Rank)	4	4	5	4	5	5	5	4	4	5
12 (Rank)	3	3	5	4	5	4	5	4	3	3
Alpha Testing Process Observation Record on Task 1										
Criteria	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10
Record 1	9 times	12 times	5 times	2 times	0 times	0 times	10 times	7 times	5 times	11 times
Record 2	5 times	8 times	4 times	1 times	0 times	1 times	6 times	1 times	0 times	7 times
Record 3	0 times	0 times	0 times	0 times	1 times	0 times	0 times	1 times	0 times	0 times
Record 4	0 times	0 times	0 times	0 times	0 times	0 times	0 times	0 times	0 times	0 times
Record 5	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Record 6	8mins	10mins	10mins	9mins	5mins	4mins	7mins	8mins	12mins	12mins
Record 7	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Table A: Test result of Task 1

Acceptance Testing Questionnaire Phase 2										
Phase 5	Participants (p)									
	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10
15 (Rank)	4	5	5	5	5	5	5	4	4	5
16 (Rank)	4	5	5	4	5	5	5	5	5	5
17 (Rank)	4	4	5	4	4	5	4	4	4	4
18 (Rank)	4	4	5	5	5	5	5	4	4	3
19 (Rank)	4	4	5	4	5	4	5	5	4	4
20 (Rank)	5	3	5	5	4	5	4	4	4	4
21 (Rank)	5	4	5	4	5	4	5	4	4	4
22 (Rank)	4	3	5	4	5	4	5	4	3	3
23 (Rank)	4	5	5	4	5	5	5	4	4	5
24 (Rank)	4	4	5	4	5	4	5	5	4	5
25	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Alpha Testing Process Observation Record on Task 2										
Criteria	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10
Record 1	5 times	7 times	3 times	1 times	0 times	0 times	9 times	8 times	10 times	9 times

Record 2	7 times	10 times	7 times	0 times	0 times	0 times	2 times	0 times	0 times	8times
Record 3	0 times	0 times	0 times	1 times	1 times	0 times	0 times	0 times	0 times	0 times
Record 4	0 times	0 times	0 times	0 times	0 times	0 times	0 times	0 times	0 times	0 times
Record 5	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Record 6	4mins	8mins	10mins	8mins	6mins	3mins	11mins	8mins	15mins	11mins
Record 7	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Table B: Test result of Task 2