The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student)________________________
Summary

The main objectives of this project relate to testing interactive television services. They can often be very unreliable, leading to the implementation of a strict manual testing process. This is likely to involve a single member of the transmission team spending considerable amounts of time repeatedly testing any number of applications hosted by that company, time that could be spent performing other tasks within the department. It was after observing this process that the idea of producing an automated equivalent was conceived. If it were possible to develop a system that could replace this mind numbing human process, and offer additional value to the business, a gap in the broadcast market has been found.

Despite realising the problem exists the remainder of the projects aims are related to researching the issues and developing a system in concept, through investigating the environment to establish the true requirements and to identify the possibility of firstly the feasible of a solution and secondly the architecture needed to support it.

This project considers these topics and goes some way to developing a conceptual solution using a variety of different technologies that would solve this problem.
Acknowledgements

Contributors

I would like to thank all those that have contributed in anyway to the content of this report, specifically the following who have provided comments and opinions on both the concept and the solution.

Stuart Ingram of Pace Micro Technologies for the help during the early stages of project

Tim Wilson for taking the time to share his expertise, currently working for Simply TV

Peter Staling, former DITG engineer currently Engineering Manager for YooMedia
# Contents

## 1 Problem Definition 1

1.1 Introduction ................................................................. 1
1.2 The Origin ................................................................. 1
1.3 The People involved ...................................................... 2
1.4 Project Direction ............................................................ 3
1.5 Pace Micro Technologies Ltd ............................................. 3
1.6 Summary .......................................................................... 5

## 2 Methodology 6

2.1 Development Methodologies ................................................ 6
2.2 Introduction ....................................................................... 6
2.3 The Waterfall Model and System Development Life Cycle (SDLC) . 7
2.4 Rapid Application Development and Dynamic System Development Method .................................................. 7
2.5 The Unified Process ............................................................ 9
2.6 Agile Techniques .................................................................. 9
2.7 The Actual Methodology Followed ......................................... 10
2.8 Summary .......................................................................... 11

## 3 Background Research 12

3.1 Introduction ...................................................................... 12
    3.1.1 To Broadcast .................................................................. 12
3.2 Digital Television .............................................................. 13
3.3 Interactive Television .......................................................... 13
3.4 Set-top Box ....................................................................... 14
3.5 Satellite television .................................................. 15
3.6 Cable Network .......................................................... 16
3.7 Return Path .............................................................. 17
3.8 Simulating the functionality of a Development Set-top Box ............... 18
3.9 RMI (Remote Method Invocation) and Distributed Objects ............... 19
3.10 Middleware and the set-top box software architecture .................... 20

4 Development .......................................................... 24
4.1 Introduction ............................................................. 24
4.2 Real-World Broadcast TV Architecture .................................. 24
4.3 The Set-Top Box with regard to application development ................. 27
  4.3.1 IR Blaster ............................................................ 28
  4.3.2 Middleware .......................................................... 28
  4.3.3 Low Level Control of the BOX .................................. 29
4.4 The Development of the Concept Solution ................................ 29
  4.4.1 General Development Process and Language choices ................. 30
  4.4.2 Java ................................................................. 30
  4.4.3 Eclipse development suit ........................................... 31
  4.4.4 JavaRMI ............................................................. 32
  4.4.5 CVS - Concurrent Versions System .............................. 32
4.5 Back-End System ....................................................... 32
  4.5.1 Testing Unit - Application Control ................................ 33
  4.5.2 The Testing Process ............................................... 34
  4.5.3 Error Logging Unit ................................................ 35
  4.5.4 Email and Text .................................................... 36
  4.5.5 Timer ............................................................... 37
  4.5.6 Database and Properties ......................................... 38
  4.5.7 GUI ................................................................. 39
  4.5.8 Simulating the Distributed Platform ................................ 39
     4.5.8.1 Packaging of the values .................................... 41
  4.5.9 Summary of the units developed .................................. 41
4.6 Front-End System ................................................................. 41
4.6.1 Using JSP Using Eclipse ................................................... 43

5 Evaluation ........................................................................... 44
5.1 Introduction ....................................................................... 44
5.2 Framework for evaluation .................................................. 44
5.3 The development process its self ....................................... 45
5.4 The software functionality in its own right ....................... 46
5.5 Is the functionality appropriate to the solution? ............... 48
5.6 Is the solution conceptually correct? ............................... 49
5.7 Is the architecture developed appropriate to such a testing application? ................. 50
5.8 Overall strengths and weaknesses .................................... 51

6 Conclusion ......................................................................... 52
6.1 Qualifying the final conclusions of the project as a whole ................. 52

Bibliography ........................................................................ 54
Chapter 1

Problem Definition

1.1 Introduction

The content of this chapter presents the focus of the project, where the problem originated and the issues surrounding the solution. The final section will detail the aims of the project and give a brief description of what these involve. Also detailed within the contained sections is the deviation from the original specification, along with the reasons for the changes and how these affected the project as a whole.

1.2 The Origin

To set the background for the project requires the introduction of a broadcasting house situated in east London. This company is responsible for the video content of several main stream television channels, as well as, providing their interactive content. It was at this location that the issue of testing interactive TV content was observed (2003).

The company mentioned above, namely DITG were one of the key players in the Digital interactive TV (iTV) market. The company was first established in 2001 and from that year onwards offered services that were previously only available from Sky themselves[21]. This opened up the iTV market to many more broadcasters who need the independence of a third party company, to provide their interactive content. Several major channels adopted DITG as the company to fulfil this role with the BBC, Channel 4, Channel 5 and MTV being some of DITGs early customers. The DITG portfolio of
both customers and interactive applications grow over the following years as did its success within the market leading it to win several awards[21]. The DITG name was dropped after its merger with YooMedia plc in April 2005 which has further propelled the company into new areas of interactive application development and service generation.[20]

YooMedia essentially have an obligation to ensure that each of the services it hosts is 100%. Therefore, each service is tested several times throughout its designated live-time. As no automated method is currently available, the only way that this testing can be implemented is manually, this involves a member of the transmission team performing a test using a standard set-top box receiver. This process is far from streamlined and it can often be very slow and cumbersome to navigate through an application, taking several seconds for a single page to load. However, this form of testing does ensure that an interactive application has started correctly and is functioning appropriately. If any part of the application fails the engineer responsible for that service is informed. When this testing process is scaled proportional to the number of clients that YooMedia have, this process begins to involve more than a single transmission operator. This is obviously a great burden on the resources of the company and the time of the individuals performing the checks. It was also realised that operating under this workload has led to short cuts in the testing processes, which is of great concern to the transmission management. Each of these claims have been confirmed by an ex member of the transmission management team, namely Tim Wilson and there thoughts are presented as part of 6.1.

The exact nature of what is required to automate this process is covered in detail within the chapter 4.5 of this report. However, in summary of the problem as it exists; there are issues relating to performing a guaranteed monitoring service for all the iTV services that YooMedia host. This is currently facilitated by the manual checking of each application by a transmission operator. This however has led to certain concerns raised by staff throughout the company structure of this processes shortcomings. Some of these have been identified by Tim Wilson and are presented in Appendix B.

1.3 The People involved

With the problem identified, is there a real desire for the solution? To discover the answer the allocation of responsibility needs to be understood. Essentially the responsibility for hosting the interactive service
from the point of view of the customer, rests with YooMedia despite many of the processes being handled by 3rd party companies; namely BSkyB and BT Broadcast Services. In return for this YooMedia gets paid a fee and as providing interactive services is one of the companys key products it therefore, becomes of significant management importance that these services are maintained, giving more gravitas to developing an automated solution. In addition to this being able to provide actual records of a services continued operation to the customer is a significant advantage, as this would allow YooMedia to validate its side of the agreement with each of its clients.

The automated monitoring of these iTV services will not only act as a management tool providing quantifiable data but would also release the companies human resources to be able to carry out other transmission activities. In addition to this there are also the benefits relating to the technical support of these services and the value they will gain from the proposed system.

The functionality afforded to the various staff related to iTV services is presented in figure 1.1, which shows the different levels of the company structure and how each of them is likely to take value from such a system.

1.4 Project Direction

From the details presented it is clear that there are issues relating to the automated testing of interactive TV applications. It is also clear that there are areas that need to be further explored to fully understand the inherent difficulties associated with such a problem. The general direction of this project is to explore and understand the issues associated with such a testing applications and to develop a conceptual architecture suited to such a system, as could be translated and implemented within a real-world broadcast environment. Using technologies and paradigms akin to such a real-world system to support the understanding of the related issues. In the architecture and development chapter of this report 4, the exact nature of the solution will be presented along with examples of both-real world and simulated system implementations.

1.5 Pace Micro Technologies Ltd

“Pace is a leading developer of digital TV technologies for the global payTV industry.”[19] A statement largely refers to the development of set-top box hardware and software for both domestic and profes-
sional users. Pace is also one of the largest and most involved developers within the global industry of digital television with a large UK base and a substantial research and development centre local to the Leeds area[22]. It was therefore, realised and stated in the original specification of this project, that in seeking the support and expertise of this heavily related company would greatly advantage the project. Not only would with technical support and the provision of the required hardware be valuable but also the improved leaning process.

After an initial period of expressed interest by Pace it became evident that such support was not going to be possible. This was essentially realised after meeting with the lead engineer for testing applications (Stuart Ingram). It was at this meeting that a provisional architecture and the project’s objectives were presented with the aim of finalising the support required. However, after this discussion Stuart highlighted several reasons why supporting this specific project was going to be difficult. This was by no means a reflection on the willingness of Pace, but more related to the actual feasibility of
coordinating the cooperation of all the third party companies that would need to be involved. It was therefore, decided to continue the project along a slightly modified course, maintaining the original brief but slightly tailoring the criteria to suit the development of a conceptual solution, presenting similar functionality in a simulated fashion.

The time spent discussing this project with pace was by no means wasted as Stuart Ingram was able to provide valuable information that would aid in the final specification of the simulated architecture, as well as, identifying certain technical obstacles. These comments have been recorded and presented as part of Appendix C, along with some of the correspondence exchanged and the original brief presented to Stuart.

1.6 Summary

In summary, the aim of this project is; to investigate the real-world environment surrounding the testing of interactive applications and to develop a conceptual testing application and supportive architecture. It is through this process that; the difficulties and obstacles inherent to such a system will be identified. This architecture will then be considered and its potential evaluated with a view to further low level development and translation to a working implementation. The functionality of this conceptual testing application as identified and illustrated in figure 1.1 will form the basis of the software element of this project, this is further discussed in Chapter 4.

The essential role of this testing application is to provide to iTV companies a way to support their business objectives of “develop[ing] innovative interactive TV formats and provide[ing] a range of interactive TV solutions to broadcasters.”[21] The above sections have presented the problem from several different viewpoints and have given an understanding of the domain and the issues relating to it. Many of the terms used throughout this report may appear to be a little abstract, however, the following chapter take many of these terms and presents a comprehensive definition suitable to the context of this project and further discusses some of the topics touched on in this chapter.
Chapter 2

Methodology

2.1 Development Methodologies

Up until this point, the projects aims and motivation have been presented; detailing its origin and the technology that currently surrounds this field. In the following sections the development methodology is considered, outlining the standard practise and detailing the projects design and development stages.

2.2 Introduction

The term methodology, as used in this report refers to the underlying guidelines that are followed to reach the development goal. There are many variations of development methodologies, each with their own strengths and weaknesses and many having grown and adapted to suit different applications and styles of development. In the very early stages of the project it was realised that this project differed in many ways to the majority of projects that these methodologies evolved around. It was also realised, that in selecting a single approach to follow rigidly, that this would fail to encapsulate all of the areas that an individual project is likely to face. It was therefore, decided that a more appropriate analysis and development approach, would be to select the sympathetic components of several methodologies best suited to the specific aspects of this project. Several of these aspects are presented below, followed by a number of sections offering a brief understanding of the different methodologies contributing to the development process of this project.
• The realisation of the projects aims being the responsibility of a single individual and not that of a team of analysiss and developers.

• Both the duration and the number of people involved reflecting the scale of the project, suggesting that being over burdened with documentation is not something that is not valuable to this project.

• The style of the project, developing a conceptual prototype over an end-to-end implementation. Suggesting that there is likely to be aspects that are unrelated to the aims of this project.

2.3 The Waterfall Model and System Development Life Cycle (SDLC)

The essence of the Waterfall model is; that each individual stage is part of a sequence of events that when complete form a working and implemented project. Each stage in this sequence refers to an element of the development process. The naming of these stages is often dependent upon the implementation being followed. Essentially the process starts with an inception stage and continues through analysis, development and implementation stages until the application is complete. At the end of each stage a review of the progress and success is carried out before it is passed off to the next in stage in the sequence. Each stage encapsulates a specific set of tasks that are the responsibility of a specific team of developers[4], however the exact nature of these tasks are beyond the concern of this section. The main points that are important to understand are; the progression in a sequential manor and the realisation that the development of a working system falls into several distinct sections. An adaptation of the Waterfall model first conceive by the National Computing Centre in 1969 realised that a project is more likely to evolve in phases and not in a single development process. This introduced the concept of the System Development Life Cycle (SDLC) which essentially took the same precess as the waterfall model and iterated through it more than once and at different levels of abstraction gradually raising the level of completeness.[4]

2.4 Rapid Application Development and Dynamic System Development Method

The Rapid Application Development approach (RAD) never actual achieved methodology status, however, a framework for applying its methods was established by the Dynamic System Development Method consortium in 1995. The RAD methods are based on the realisation that projects failing in
the 1980s and 1990s, were mainly due to the prolonged arrival of the solution.[3] This prolonged single phase development did not allow for the change of requirements that had often taken place in the interim. Therefore the RAD uses an alternative approach to development, that both involves the user and realises the continual changing needs of the application’s environment. It encapsulates many of the core stages of the Waterfall methodology, however, one of the main aspects of the RAD framework is the use of prototypes.[3] Prototypes are used to feedback issues to the users whilst ensuring that, what is been developed suites the needs to the customer. This is distinct from a prototype led development as this often uses prototyping to support the initial stages or concepts of development i.e. proof of concept or architecture and never realises a complete solution. The RAD approach uses prototyping throughout the development process allowing prototypes to evolve into a working application.[4]

![Diagram of Dynamic System Development Method](image)

Figure 2.1: Illustrating the typical software development life cycle of the Dynamic System Development Method.[4]

Similarly to the SDLC mentioned above, the core aspects of the development process are iterated through a number of times before the final solution enters the implementation stages. There are many other features of the RAD and DSDM development frameworks that are well documented elsewhere. It is, however, important that the prototyping aspect of these frameworks be realised as an effective way to feed back progress and functionality to both the developer and the users when developing software applications. [4]
Figure 2.1, shows the typical development process of the Dynamic System Development Method. As is shown the major section of the development process is the iterative prototyping process, carried out prior to implementation. It is this section that directly relates to the development process undertaken by this project as is discussed in section 2.7.

### 2.5 The Unified Process

The Unified Process (UP) is a popular software development process that uses both an incremental and adaptive approach to object oriented software development. The UP incorporates many support techniques that aid in the lightweight generation of analysis documentation. [8] The UP uses a 4 phased development process, with each phase having a different focus these, are namely; Inception, Elaboration, Construction and Transition [3]. These 4 phases reflect progression through the end-to-end development process. Observing these 4 phases presents a framework for where the concentration of effort should be focused. Another feature of the UP is that it is use-case driven, this refers to the functional requirements of the project been the main focus. Specifically a usecase is a single activity that the software application must be able to perform, these often relate to the functional needs of a user i.e. enter customer details. This gives a greater focus to the functionality of the software throughout the development process and allows the focus of the project to reflect the users requirements. [3]

A very basic summary of the UP is presented by Bennett(2006) et al. Do some investigation, model the requirements, analyse them, do some design, do some coding, test the code, then repeat the whole process. [3] It is this flexibility that allows the UP to be adapted to suit many software development projects.

### 2.6 Agile Techniques

A lot of what is mentioned above has grown to be complimentary in modern software development and therefore there are significant overlaps in the methodologies previously outlined. This section describes the advantages of how aspects taken from the above methodologies can be applied with a new agile perspective providing new opportunities in recognition of their previous weaknesses.

An agile approach to software development can be summarised under the following four titles; incremental, cooperative, straightforward and adaptive. These were presented by Abrahamsson(2002)
et al in a Review of Agile software development [2]. In the same report, it was also stating that; the focus of Agile development was its simplicity and speed with regard to only the aspects of the development process that related to the functionality of the software.[2] Essentially an agile approach to development can be applied to almost all development processes, reducing the heavy burden of documentation and increasing the speed of analysis. These have always been important to the development process and therefore suggests that adopting an agile process has many advantages.

2.7 The Actual Methodology Followed

The several methodologies outlined above each have positive aspects that can be applied in the development of a small, single manned application development process. Bennett(2006) et al suggests that the adoption of an appropriate approach to development, specific to each application, is crucial to the success of that application. What follows is a breakdown of the contributing aspects of the above methodologies and the benefits that they afford the development of this project.

As stated, it is important that the methodology used for software development suits the nature of the project. To achieve this, the union of several aspects presented in the previous sections have been applied to form an appropriate development process suited to this project. Taking aspects such as lightweight analysis, agile support techniques and prototyping combined with an iterative and phased development cycle specific to the requirements of a single person development team has provided a well suited and adaptive development process. In the following section the details of the individual aspects and how they contributed to the development process will be presented.

As mentioned at the beginning of this chapter one of the main influencing factors is that the development team consisted of a single developer. This same individual is also responsible for performing the analysis of the application environment. Due to this single individual holding all the knowledge for both analysis and development the generation of needless documentation to support these processes would have significantly slowed the project’s pace. This single fact led to the adoption of an Agile approach that allowed the project to take a functional direction and for documentation to be generated where absolutely necessary. One of the other influencing factors likely to affect the development stages was that a lot of the technology being used was going to be new and therefore the understanding of this
technology was going to be a progressive process. Therefore the development process has to be able to allow for changes and be adaptive to new requirement when the technology was realised. This in turn led to the adoption of a phased and iterative development process. Lastly the fact that the project was essentially developing a functional prototype for a concept solution led to the adoption of elements taken from the RAD methodology, specifically the evolutionary prototype allowing the product to develop through several iterations. With regard to testing, this is incorporated at the end of each phase and takes a functional evaluation approach over a unit driven development approach as is appropriate when developing a solution in concept.

The reason for the inclusion of the Waterfall methodology is that many of the contemporary methodologies have elements that are based on it. For this single reason it is important that the key features of the Waterfall methodology be understood, to give context to the improvements as applied in the latter methodologies. The aspects important to this project are the discreet steps that a project passes through and the processes encapsulated within each stage.

2.8 Summary

To summarise the development process as applied in this case; an on the whole UP led development, whilst adopting the evolving prototype element of the RAD process and the Agile techniques often applied to object oriented software development. In turn allowing for a lightweight and rapid development of a high fidelity, functional prototype to the concept solution of interactive TV application testing. This section has summarised the methodologies important to software development and specifically this project. It has presented details of the aspects found to be valuable in performing, lightweight and rapid development of a software application and highlighted the issues specific to this individual project.
Chapter 3

Background Research

3.1 Introduction

This section describes the background details relating to the projects subject area. Through out this section the details of the terms, technology and hardware involved in realising this projects aims will be described in sufficient detail as to provide an understanding of this specialist area.

3.1.1 To Broadcast

With this project sitting between the television and computing domains, several of the terms used within this report have differing meanings. To clarify the definition of one such term, which operates in the context of both television and computing the relevant definition will be presented bellow.

Broadcast; essentially the word broadcast regardless of domain refers to the same main point; that something is being presented from a single point of transmission to a greater number of receivers. This could refer to a radio, television or network transmission or even as simple as human speech.[1]

Within the Television world ‘broadcasting’ is often used to refer to the industry its self and can also often be prefixed to many processes to identify them as belonging to the domain of Television specifically, i.e. Broadcast Satellite.[5] The reason that this is important is that within the chapters of this report both the Computing and Television domains are discussed and in a later chapter the details of the architecture are presented with the word broadcast being used to describe the simulator and the portals used, within
the same chapter the actual architecture of the Satellite transmissions system will also be summarised. It is, therefore, important that throughout this report that the specific context is observed and the correct definition associated.

### 3.2 Digital Television

Digital Television, regardless of all its technical advantages, is simply another way for a television signal to be broadcast from a point of transmission and received by the end users, i.e. the viewers. The exact technical implementation of how this is done is outside the scope of this report, however, there are many functional reasons why transmission in a digital form is better than its soon to be redundant analogue predecessor.[10] Some of these are the added features that can be multiplexed within the digital signal and later used by the Set-top box; added security (encryption), system updates and interactive services are a mere few that are used across many of the digital platforms available.[10] These digital television services are extended to UK users over several platforms; the cable television network, satellite television network, IPTV, and digital terrestrial or Free View as it has become known. Each of these services carries a subset of the digital services that are available, but all provide a degree of user interaction. The extent of the interaction varies across each of the networks, as does the way they are implemented. The most common term that many interactive users have come to know is ‘press red to…’, which originally referred to interactive services available over the digital satellite network, inviting users to press the red button on their remote control to activate the interactive service, this technique has since been adopted by many of the other digital networks.[27] This move into a digital broadcast environment is another example of the ever increasing convergence between computing and television and it is this technology that is still very much under development allowing providers to offer a greater range of money making services.[10] Many of which are touched on in the following sections.

### 3.3 Interactive Television

Interactive television is another term for enhanced television, they both refer to the same core aspects. A definition taken from the Sky Interactive website states that; “Enhanced TV is a simple way for viewers to interact with a TV show or extend their experience of a show by viewing more than just the linear broadcast.” [28] It is this extended experience that is at the core of this project and what has been referred to in previous chapters as an iTV service. There are many different types of interactive services...
available, which are generally classified into 3 bands; low, moderate and high. Each of these bands represents a differing degree of user interaction. A low level of interaction involves adjusting the volume and changing channels, a moderate level of interactivity is delivering on demand movies, with a high level of interactivity covering aspects such as voting systems, where the actions of the user directly affect the channels output. These different levels of ability stem from the technical capabilities of the provider and the hardware of the user, however any further understanding of why they are categorised in this way is beyond the scope of this report.[9]

Interactive services can be a slightly ambiguous concept, leading to much discussion amongst those who provide them as to what actually constitutes true interactive content. The term also encapsulates a much broader spectrum of services, than are necessary for the understanding of this document and project. Therefore the term interactive service and interactive television will refer to a specific type of service used throughout the various digital networks. This service is a form-based service and is often offered as a convenient way of capturing details entered by a user. Many services make use of this technique and have incorporated its functionality somewhere within their service. An information request running throughout a commercial break for a specific product is a simple example, a registration form for a further service is another. These form-based pages are often the gateway to what the host is actually providing and therefore, represent an important aspect of the service as a whole.[10]

For more information on Enhanced Television and Digital Television the Sky Interactive website offers many examples and case studies that refer to specific formats of Enhanced TV available over the Sky platform. http://www.skyinteractive.com

3.4 Set-top Box

Set-top box (STB) and Digi-box are terms that have become inseparable from what is associated by many to be a standard piece of entertainment equipment present in many living rooms across the country. It represents a piece of hardware that, within the UK, allows the user to interface with the services of a chosen television platform. In many cases this piece of hardware acts as a simple receiver and decoder which allows a viewer to see and hear the chosen channels content. However many modern STBs offer access to a greater number of features and are beginning to incorporate many other features that would
be associated with a visual entertainment system. For example, digital recorders replacing the need for VHS recorder, Electronic program guide (EPG) giving channel information and automated program reminders to name but a few. There are however many more services that are offered by individual channels and not solely from the STB. It is these services, as outlined in the Interactive Television section, that are the area of interest throughout this project and it is through the STB that the user is granted access to them. The STB acts as a single point of access allowing a user to interact using their remote control and television set. [10]

Each of the networks offering interactive services use a STB as it is a robust and acceptable device providing a secure and adaptable access to their services. From the prospective of this project, a development STB will be used to act as the point of access to the interactive services that are going to be tested. It is through the use of this development STB that the original form will be received and displayed that the completed form will be transmitted back to the services host. This transmission generally happens through a channel that is known as the platform’s return path and is a term that basically refers to the link used to connect the user to the host.[9] There are many different way that this can be implemented and these are detailed separately in a later section. A development STB essentially provides the same functionality as a standard STB with the added ability to interface with a PC.

### 3.5 Satellite television

Satellite TV, Sky Digital and Sky TV are all names for what within the UK refers to; BSkyB’s digital satellite subscription package. In 1990, around the time when Sky Digital was first established, it offered more than 200 digital channels and was the first consumer subscription package available.[16] It offered new services that were not available on any other platform of the time and was a direct competitor of the digital cable network. Sky Digital also introduced the availability of subscription only services which, 16 years on, are now common to many digital television platforms.[10]

Each of the names associated with this service in some way refer to the infrastructure over which the television signals are distributed, derived from the fact that the broadcast aspect of the transmission is achieved through the use of geostationary communication satellites. This method of public broadcast television was first implemented in 1962 when a television signal was relayed from Europe to the Telstar
satellite over North America [9]. Its main advantages are its increased bandwidth, allowing for better picture quality, and space for additional content to be multiplexed within the digital video signal, e.g. provision for other multimedia and interactive services. The sky platform also has a large coverage that offers increased public reach.[10]

Figure 3.1: Illustrating the typical transmission setup of the satellite platform.[28]

3.6 Cable Network

As mentioned above Sky Digital and the Cable networks were, and to a certain degree still are, in direct competition, often providing identical channels. The cable network first came into being in 1938 when it was introduced to allow those that could not receive terrestrial television signals an alternative method of receiving television. Large community masts were installed and each house cabled to it, these were often in remote or mountainous regions where signals strength was very low.[17] Today (2006) there are only a hand full of independent cable networks left in operation, with the majority of the infrastructure being run by NTL or Telewest; these two companies have since merged to form NTL incorporated.[13] The main difference between the Satellite infrastructure and the Cable infrastructure is that the former is broadcast and the latter is effectively a point to point connection. This seemingly small difference is in fact of significant interest, as being able to communicate with an individual user over a dedicated connection opens up many more avenues of services that can be implemented.[18] This aspect of the communication infrastructure will be discussed separately in the following section and discussed within the architecture chapter later in the report. However, to summarise the issues, up until recently when more interactive and multimedia content has been provided to the consumer, there was no real need
for a single high speed connection extended to every user.[10] A fact that has now changed and the realisation of this has led to much development in the area of return path technologies by many of the large manufactures and service providers. [29]

3.7 Return Path

As previously mentioned the ‘return path’ is the term that is used to refer to the connection that is established between the user and the host. It is important to recognise that it is the set-top box that establishes the connection with the host when it needs to do so and not the other way around. It is synonymous throughout the Broadcasting industry regardless of platform and is a regarded as a critically important aspect of all contemporary user interaction. Essentially it is via this connection that a user can send data to the host’s back-end systems. For example; a completed questionnaire, a vote or registration information would all be types of information that are sent over this channel. Billing and payment information for any of the pay TV services are also collected via this same channel, this however, is carried out in a slightly different manner but still supports the value and importance of this communication link. Interactive games also make use of this link; however, it is only of limited bandwidth and generally only deals with small amounts of traffic. [9]

"DITG is the only return path service provider for the Sky platform in the UK other than Sky themselves. [We handle] over 6 million financial transactions each month in addition to voting and other responses...”[21]

The short statement above, taken from DITGs website (2003), shows the scale of this specialist field and how few companies had the ability to provide this service. It is, therefore, of strategic advantage to be able to maximise the functionality of this communication channel for increased interactivity and a service that the demand has only grown for in the passing years. When specifically referring to Sky Digital, the return path is implemented via the customers telephone line[23]. This has inherent limitations and has led to the other possible implementations being investigated, network connections and higher bandwidth modems are being integrated into many of the new set-top boxes, to allow for increased connectivity [22], with also research in the United States looking into the feasibility of using wimax for return path transmissions[29].

The cable networks and IPTV networks have a superior return path. They can make use of the point to point link that exists as part of their platform implementation, creating a one to one relationship between
each user and the host. This implementation offers the opportunity for greater communication between the two parties, with the added advantages of using the connection oriented protocols of the TCP/IP stack.[9] The recently launched IPTV system from Microsoft and BT is an example of such a service that makes use of both broadcast channels and the high bandwidth backbone of BT to enable its on demand and interactive services.[24] More information is available on the MicrosoftTV website. 

http://www.microsoft.com/tv/default.mspx

The return path arrived with the launch of Sky Digital and it has slowly been developed and exploited by the likes of Sky and DITG who are at the forefront of the interactive TV market.[30] It is due to the return path that an interactive service can be truly interactive, as without this there would be no two way communication and therefore, no high level interaction[10]. Figure 3.1, showing the architecture of the sky transmission also features the return path. Also the the figures presented in chapter 4 offer further illustration of its conceptual implementation.

### 3.8 Simulating the functionality of a Development Set-top Box

As previously described one of the most significant aspects of this project is the development of the simulator. It is this simulator that will take the place of the development set-top box and the associated functionality, as detailed in section 3.4. The simulator has grown to be a major part of the project as the project has changed direction after deciding not work alongside Pace Micro.

The original aim of this project was to use a development set-top box, to allow the test application to interact directly with interactive services running over a given platform. It has since been realised that to achieve this would involve the cooperation of several large companies, namely; the infrastructure owner - Sky, the channel whose service was to be tested and the interactive service host. As this was not achievable in the relatively short time available, an adapted version of the original brief was created. This alteration replaces the hardware set-top box with a simulated set-top box that will be developed as part of the project specification. It is, therefore, important that the functionality of this simulator fulfils all the areas that are relevant to this project.

The simulator itself will take the form of a distributed system, with the server providing interfaces
to the functionality required and the test application interacting with the client side operations. The schematic drawings included within chapter 4, illustrate the aspects of the real world infrastructure that will be encompassed by the simulator.

3.9 RMI (Remote Method Invocation) and Distributed Objects

RMI is essentially a way of managing remote classes over a distributed system, allowing distributed methods and values to be accessed from a remote location as if the classes themselves were local to the call. This is done through the use of interfaces, where the remote class makes certain methods available to distributed clients via one or more interfaces. These interfaces extend the methods contained within the class needed to access the values of the remote object. It operates as a client and server implementation where the object server managing the remote objects, methods and values associated with them and the client invoking these remote methods through the use of a proxy. The proxy is downloaded to the clients address space upon requesting and binding to a distributed object. The specifics of how this is done are not important for this introduction. Figure 3.2 shows the typical client and server scenario relating to distributed objects.

Figure 3.2: Illustrating the client and server architecture of a typical RMI implementations.[12]

As shown in figure 3.2, the client proxy, is in effect a copy of the remote objects interface and it is the functionality presented by this interface that the client can call. This call is then marshalled by the proxy into messages and then passed over the network to the object server where the server side skeleton, un-
marshals this call and invokes the method of the actual object itself. The result is then marshalled by the skeleton and passed back over the network where it is un-marshalled and passed to the client. The lower levels of this process are transparent to the client and the remote object appears to be local to the calling class.

Figure 3.3: Shows a more detailed representation of the RMI messaging system.

Figure 3.3, shows the actions of the proxy relative to the invocation of a method of a remote object. The image itself has been taken from a series of slides presented by Dr Karim Djemame as part of his Distributed Systems course as taught at the University of Leeds, UK, 8 December 2006.

In summary the remote architecture facilitates the ability to have objects located over a distributed network. The methods of these objects are then able to be accessed in a controlled and managed manner via the objects interface. Instance variable and other results can then be returned to the client through the marshalling process controlled by the client and server proxies.

3.10 Middleware and the set-top box software architecture

An important aspect of the set-top box architecture is the way that the iTV applications are handled and where they reside. To understand this allows for greater access to the core of the application and the ability to manipulate the functionality of the set-top box to the needs of this project. This section will present some of the aspects relating to the middleware software architecture and give details of its role in interactive television content.
The general role of middleware in any architecture is to allow for an element of abstraction from the low level details of the hardware implementation.[1] This is very much the same in the set-top box architecture which is becoming an important area of development with many different implementations of middleware software becoming available, with each trying to offer a competitive advantage over the other.[10] An issue relating to the development of this project, is the different transmission platforms that exist. This is important due to the different middleware implementations that are used for each platform. Cable uses different middleware to satellite and both of these use different middleware to the new IPTV systems that are becoming available.[6] The New Media Markets article, a guide to digital-television middleware in Europe as included within Appendix F, presents details of the distribution of the different middleware implementations. It shows that the main middleware architectures at the time of the article going to print (2004) were, OpenTV, MHP and Media Highway. This is still the case today with the addition of the OCAP system.[9]

Despite the differing platforms and middleware implementations, the core functionality of the middleware and software stack present in every set-top box is the same; to present the low level functionality of the platform and transmission methods to the higher level applications. This relates to the original OSI specification for the seven layer architecture as an underling structure for interconnecting computer devises.[10] Figure 3.4, shows an example of the software stack implemented within a set-top box and the separate layers and protocols associated with the processing of a Digital Television signal and iTV content. This is by no means the definitive implementation and should only serve as an illustration of the components involved.

Figure 3.4, does however, show the many commonalities with all the middleware languages. One of the most important of these is the use of the Java virtual machine, it is, in fact the case that many of the middleware languages available today have close ties to the Java programming language. To say that this was the core of the architecture would be short sighted and inaccurate as there are many other complex aspects that are associated with MPEG manipulation and the video streams them selves that are incorporated within the middleware layer. However having said this, Java TV is under development by SUN Microsystems that is an entirely Java based middleware solution.[25]
This evolving market is a result of the increasing demand for more dynamic and truly interactive applications, however the core role of the middleware software is to manage the iTV applications and the resources of the set-top box. There is of course a long list of functions that the middleware performs, but the following selection presented in figure 3.5, will suffice to give an idea of the processing involved regardless of implementation or platform. [9]

<table>
<thead>
<tr>
<th>Stage</th>
<th>Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – Basic OS interfaces</td>
<td>Threading, memory, debugging support, file system, synchronization primitives, and so on</td>
</tr>
<tr>
<td>2 – Graphics</td>
<td>Basic graphics primitives, drawing to TV display</td>
</tr>
<tr>
<td>3 – Java VM</td>
<td>Other elements needed to run the JVM</td>
</tr>
<tr>
<td>4 – MPEG filtering</td>
<td>Tuning, PES filtering, section filtering</td>
</tr>
<tr>
<td>5 – Audio/video support</td>
<td>MPEG decoding and other graphics or audio formats</td>
</tr>
<tr>
<td>6 – Security</td>
<td>MHP security, certificate management</td>
</tr>
<tr>
<td>7 – Other TV functionality</td>
<td>Remote control input, teletext decoding, subtitles/closed caption support, modem support</td>
</tr>
<tr>
<td>8 – Conformance testing</td>
<td>MHP/DCAP conformance testing and final debugging</td>
</tr>
</tbody>
</table>

Figure 3.5: Some of the functionality extended by a typical middleware implementation.[9]

To summarise, the middleware industry is vast, it is rapidly growing and adsorbing new technology and realising new potential. It is therefore, by no means standardised giving rise to many implementation and platform discrepancies. Despite its relative infancy the middleware available is very powerful and encapsulates massive functionality and flexibility. It is the applications developed for these middleware
environments that this projects testing solution is aimed at. It is also at this middleware implementation level that the concept solution is pitched.

For further details of the implementations available and the current development in progress the following Websites offer a greater depth of knowledge, also included in Appendix K are a collection of papers relating to the functionality of the most popular implementations.

http://www.opencable.com/ocap/
http://www.mhp.org/
http://www.opentv.com/
Chapter 4

Development

4.1 Introduction

The following sections detail the actual development of the solution, presenting the architecture surrounding this real world domain and the architecture developed as a concept solution to interactive TV application testing. The differences between these two architectures will also be discussed and the issues associated with them presented. The first part of this chapter will detail the real world environment, in order to provide some context to the concept architecture that makes up the second part of the chapter.

4.2 Real-World Broadcast TV Architecture

The real world architecture essentially refers to the environment and infrastructure associated with the broadcasting industry, specifically the areas that relate to this concept solution. It is these areas that will offer a greater awareness of the technical architecture associated with interactive TV and the services that are available. Starting from a conceptual view point and progressing in detail towards a more comprehensive definition of the systems that are in use across the interactive TV industry. With the aim of providing an understanding suited to the conceptual solution of interactive application testing that is presented in the following section.

The basics of the broadcasting system can be thought of as a client and host scenario, where the viewers are the clients residing at home using standard set-top boxes and the host is essentially the broadcasting
house processing and transmitting the channel. Regardless of interactive content this will be the same for all channels and platforms. At this very high level it can be thought of as to be the same as any other service where a user has little idea of the processing that occurs behind the actual service. The internet is an example of such a service a user operating from a standard terminal faces very little, if any of the processing and routing that takes place between ISPs and backbone operators.

The users are often unaware of the communication that occurs as they use the services and the technicalities remain transparent throughout the operation, so long as everything works the users are happy and the service continues. If however an error in the system is discovered the user is likely to change channel and begin using another systems as they have no way of reporting the fault. It is, therefore, important that the testing application is able to find the fault first and give the engineers chance to fix it before it begins to affect the viewing figures. As with the manual testing system detailed chapter (problem definition) the testing was carried out from the users perspective i.e. using similar equipment and the same return path configuration.

Figure 4.1, shows this configuration and the various locations involved in providing this end-to-end service. The following section will present more details on the different components of the diagram and elaborate on certain aspects relevant to the testing architecture.

![Diagram of Transmission setup of the satellite platform.](image)

Figure 4.1: Transmission setup of the satellite platform.

This figure is based on first hand experience of the broadcast system as well as an understanding of
the several chapters relating to this topic presented in the essential guide to iTV.

Figure 4.1 essentially shows the end-to-end transmission of the broadcast stream. This includes five main aspects in this simplified diagram, the views shown to the right of the diagram, the geostationary communication satellite where the multiplexed and modulated broadcast stream is distributed to the viewers receivers, the transmission house facility where the broadcast stream is up linked to the satellite, the interactive TV firm which in this case encapsulates all processing and video generation associated with the generation of an interactive TV service and lastly the return path, which as described previously is the link between the back end systems of the interactive TV firm and the viewers. This simplistic drawing does not specify the nature of the return path link but can be thought of as to be a telephone line connection. The circle surrounding the interactive TV firm also encapsulates a Tester element; it is shown in this way as to represent the testing of the services taking place as if it were external to the company, i.e. from the view point of the user.[10]

Very few details so far have been given on the actual process of creating a digital TV service and to a certain extent the details are not important, however, a brief outline of the process will be given to show its complexity and the reasons why testing does not currently take place else where in the process.

Other than the pure video and audio generation there is of course the actual interactive content or data content that form part of the broadcast stream. These are the components that give the channel its enhanced features and additional content which is multiplexed into a single stream and sent to the transmission facility before being passed through a modulation process and up linked to the satellite. This seemingly simple process involves a complex number of other aspects that take place as part of this multiplexing process and often occur with Sky themselves. These logical units are namely; the conditional access system which is responsible for only allowing a viewer access to the services that they are eligible for, the compression and encoding of the stream and the modulation of the stream into an up linkable state.[10] These various operations render the stream in a suitable state to be located, decoded and uncompressed by the set-top box at the receiving end.

As well as these processes the actual interactive application generated is often done so at a different location and by a different company who then provides this to sky to be uploaded to the system so that
it can be downloaded by the set-top box when requested by a user. It is this separated company that has access to the accounts systems and the other database and servers that control the content of the interactive application it is also this company that will have the return path terminated with them as to receive the requests and direct them to the correct content. Now as the number of companies has increased and the resources are spread over several locations, often involving a data centre it is becoming more complicated to create a testing application that can perform the tests pre TX (transmission). Therefore it is simpler for the test to be carried out post TX and for a separate system to be designed that will allow for a certain amount of dynamic change to the architecture that provides the service its self.

With this said the testing application its self needs to be considered, where it should sit and what does it need to talk to. This essentially can be a remote application as will be shown in the following chapter the architecture designed in the conceptual solution allows for a distributed nature to be assumed and it is not restricting to any of the geographical locations previously mentioned. As for what aspects of the existing architecture that the testing application needs to communicate with, these are limited if any at all. As the testing application is intended to be independent the tests will run continually with no external dependencies other that a satellite feed and a return path connection. One issue that is raised by this testing process is that any requests generated by the system are going to be regarded as genuine requests from a genuine consumer. This is of course what the testing system is aiming for however the iTV application its self would process this request every time and therefore this would create an extra work load for the application as well as waist physical resources further down the line e.g. information literature request for a car. A possible solution to this is to create code names or spoof details that would not be regarded are real customers and could be bypassed by the back-end operations of the iTV application. This is yet another complication to the testing architecture that makes developing a solution so difficult to represent in an accurate way.

### 4.3 The Set-Top Box with regard to application development

With the test application acting from the user perspective and therefore post transmission, it is clear that the application will in some way have to interact with a set-top box. This is the only way that the broadcast stream and inherent interactive applications can be accessed.
Despite this, there are still several options available to the development of an application of this type, each with their own advantages and limitations. The following sections detail these options and describe the surrounding issues.

### 4.3.1 IR Blaster

The simplest and most straightforward of the options available to the development of an application testing suit is to use the IR (infrared) control port. This IR port is the same port that allows the user to change channel and enter the details with their remote control. The same functionality can be afforded to a computer using a IR Blaster, which would allow for a pragmatic sequence of commands to be sent to the set-top box. This would be the quickest solution to this problem and would effectively perform the exact same role as that of the member of transmission control staff that was detailed earlier in the report.

One of the major problems with this technique is that the set-top box may crash and therefore any commands sent to the unit would have no effect. With this been the case it would have to be monitored by a member of staff and would therefore provide only a slight improvement on the existing process. Another problem with this is that there is no way for the set-top box to be able to feed information back to the testing application, therefore no error logs or progress report could be generated that would give any real qualitative information. With this said it would be the quickest to develop and would provide a reduced burden on the transmission staff.

### 4.3.2 Middleware

Despite the merits of the above solution, it fails to allow the application the access required to perform the value adding tasks that have been previously detailed. Therefore it would be far more beneficial to be able to interact in some way directly with the set-top box and gain access to the lower level functionality. It is having this ability to have control of the set-top box and issue commands in an automated fashion that is key to the success of the testing application. With this said, the only remaining method is to use a specialist set-top box commonly referred to as a Development Box which has the ability to be interfaced with via a PC.

There are many advantages of using a middleware implementation for this testing application. Hav-
ing the ability to run the testing application on a development set-top box that is in turn able to control the functionality of the hardware and gain access to the low level details of the platform, allows for a significantly more powerful implementation. As has been discussed previously the middleware available contains some very powerful features and grants exposure to those low level features that are need to provide the functionality of the testing application. As mentioned above the application will run via a PC, that via a direct connection to the set-top box is able to take complete control of its functionality. Using this approach to an implementation will grant exposure to both the interactive content and the supportive architecture needed to facilitate the resat of the functionality.

4.3.3 Low Level Control of the BOX

The last of the methods available would be to use the low level embedded c code that runs directly on the set-top box hardware and makes up the units operating system. Using these layers of the protocol stack would allow for access to all of the low level functionality of the STB. There are still issues associated with operating at this level, the more trivial been that it is a more skilled operation generating code that will run directly on the hardware of the unit and is more likely to be carried out by the set-top box manufacture. It is also less dynamic and likely to be expensive to alter in the future providing for a very rigid and unfriendly setup.

Not being able to produce dynamic code may not be entirely the problem that has been made out to be as a wrapper class can be written that extends the functionality that is needed to the higher levels of the architecture. This means that the middleware languages can make use of these low level features that were not previously available and can be written in a more dynamic and user-friendly manor.

4.4 The Development of the Concept Solution

Up until this point the majority of the material discussed has related to the real-world systems that surrounds this topic. However, it is the software solution that is at the heart of this project in representing the research and investigation into the possibility of developing a real-world iTV testing application. The testing application and supporting architecture are presented in the following sections along with details of their implementation and overall structure. The simulator developed to emulate the distributed nature of the networks that the testing application would run on, is also detailed later in this section.
The developed architecture can be broken down into a number of different components, each of which having its own purpose and functionality. These can be split into two distinct areas; the front-end system and the back-end system. As is traditional the front-end system is largely concerned with presenting details to a user and in this case consists largely of a secure web based portal. The back-end system in affect is the testing application itself but also encompasses all the other processing relating to testing, results and alert systems. Firstly, however, several sections giving details on the development environment, language choices and support tools used will be presented.

4.4.1 General Development Process and Language choices

Choosing the correct language for the development of a project is a decision that requires careful consideration to ensure that all the features, functionality and support are available. There are many other factors that affect this decision, but deciding upon the correct language can make or break a project.\[3\] One of the issues important to this process, is the project's purpose and context. When considering this project specifically, it is important to remember that the software being developed is a concept solution and in reality a high fidelity prototype. This introduces a new angle to the appropriateness of a given programming language and if anything increases the flexibility for making this decision.

4.4.2 Java

It was decided that the software element of the back-end solution was to be developed using the Java programming language. This decision was made for several reasons and offered a number of advantages to the development process. Java being an object oriented programming language with a large amount of support and development libraries\[7\], offered great potential for the development as a whole. As well as this, there are also many other libraries that support specific features required by this project produced by third parties, i.e. email services and graph plotting\[26\]. Whilst disregarding the convenience angle, there is also the fact that, many of the languages used for set-top box middleware are based on the Java platform and make use of an implementation of the Java virtual machine\[9\] as previously discussed in chapter 3. Therefore based on Java's close ties with many of the middleware languages, it is logical to choose Java as the development language. The main argument that supports this claim is that a real-world testing application would most likely have to interact considerably with the functionality of a middleware implementation. As this would be the favoured approach over the alternatives presented
in the previous chapter, namely an embedded implementation and an independent infrared implementation. Therefore, in choosing Java many of the issues and methods encountered and overcome during the development of this concept solution would be similar to those found in a real-world implementation. This reduces the work involved in translating the units developed into actual working units within a real-world implementation.

One drawback from using Java as the development language is that if it was decided that the testing application was to be implemented at the core of the set-top box architecture i.e. as part of the embedded C code and interacting directly with the on-board operating system, this would then be significantly harder to translate from this conceptual solution. Being able to perform some mapping between objects used in the concept architecture and the objects and methods used in a working solution gives far greater value to the development of a conceptual implementation.

One of the most popular publicly available development environments for the Java Language is Eclipse[15], this was used extensively throughout the entire project providing support for the code management of both the front-end and back-end systems.

4.4.3 Eclipse development suit

As mentioned above the eclipse development workbench was adopted to allow for the simple management of the project throughout its development process. This development suit has many features that aid the development and management of Java software applications. Eclipse also has the ability to manage many additional libraries and plug-ins that assist in the development of specific aspects of a project. CVS and RMI are two such examples that were important to this development process, as well as, the J2EE development plug-in specific to the management of web applications. As discussed in section 3.9, RMI is a remote message passing system that allows for the remote invocation of remote objects. As this formed an important aspect of the simulator unit it was essential that this too could be managed alongside the testing application. The ability to have CVS incorporated within the development suit was also a great advantage as this allows for the entire project to be centrally managed and version control to be seamlessly incorporated within the development process.
4.4.4 JavaRMI

In addition to selecting Java as the language to develop the software element of this project it was also considered suitable to elect the JavaRMI implementation of the remote messaging system to support the distributed nature of the project. Not only is JavaRMI, as the name suggests Java based, but it is also an implementation that provides high level abstraction of the low level details related to remote messaging. This suited the nature of the solution and allowed for the distributed element to be developed alongside the testing application and managed within the Eclipse environment. The benefits of choosing a higher level approach to the messaging system is that; the low level functionality of what is by its nature a complex setup, could easily detract from the essence of the development itself. Leading to the possibility of the project overrunning its schedule and becoming overly complex for little return. Alternatives such as low level socket programming and other object based messaging systems were considered but for many reason shared with choosing Java as the main development language led similarly to choosing JavaRMI as the messaging implementation.

4.4.5 CVS - Concurrent Versions System

A CVS system is a method that allows a central repository of files to be maintained, this provides the ability for files to be checked out at remote sites and altered, updated and removed as needed. It allows for a more flexible working practise and maximises the working time of those developing the system, it is a standard feature available within the Eclipse development suit, allowing any number of individuals to use the central repository. This is not only a more productive way of working but also allows files to be archived, backed up and version numbers to be automatically assigned. Throughout the development of this project the use of CVS has increased the productivity of the software, as having access to remote files was was no longer a limiting factor.

4.5 Back-End System

As described, the back-end system is effectively all the components that when combined create the actual testing application. This also includes all periphery aspects of the system that support the testing application, like, database access, engineer alert and log file generation. Throughout this section the exact functionality of the testing application will be described, before moving on to present further details of how each of these aspects achieves it’s specific functionality. Figure 4.2, shows how each of
these units interacts and illustrates the role they play within the overall architecture.

Figure 4.2: The back-end architecture of the conceptual solution.

To summarise the concept solution; it should firstly be capable of supporting the testing processes as detailed in chapter 1 and 3, and secondly, should be implemented in such a way that is suitable to the industry that it is designed for. What is shown in figure 4.2 is a simple relationship drawing that illustrates the units involved in the testing process this architecture consisted of a system that is able to perform a single test repeatedly, on a single service, using data that has been entered by a user and stored in a database. The outcome of this test is recorded to a log file and in the event of an error a link is sent via email to the engineer responsible for the service. This is a somewhat simplistic description but captures the essence of the diagram and the application developed. The following sections give a breakdown of each unit and the specifics of how it was developed and what functionality it provides.

4.5.1 Testing Unit - Application Control

The testing unit as shown in figure 4.2 and outlined above is the central controlling element of the overall testing process, coordinating the functionality of the other units in order to perform a series of tests. This single test unit represents the functionality necessary to test a single service, to test further services other test modules have to be developed that contain the code specific to the requirements of that interactive service. The collection of test modules are then triggered by a single Java class that acts as a scheduling unit. This scheduling unit initiates the test and passes a number of values that determine
the number of times the test should run and at what time interval. Using this configuration allows the number of services tested to change, as well as, the schedule of tests themselves to be changed. The test class itself contains very little of its own functionality and basically uses the services extended from the other units that are described in the following sections. This software architecture allows additional functionality to be developed independently of the existing units to facilitate the needs of new services. Having this flexibility to alter and implement new features leads to a robust testing architecture that can adapt to new and changing services.

As part of the concept solution several simulated interactive services were developed to illustrate the functionality of the solution. The first and most simple of these was named the basic service and contains very little functionality hence the name. This basic service represented a single page that requires a user to enter an email address. This replicates an interactive service that may appear during a commercial break, where in order to be added to a mailing list for more information on some given product, the viewer has to enter their email address. It is this entering of the email address that constitutes the interactivity and therefore being able to complete this field, transmit the interactive form and receive an acknowledgement back from the host in an automated fashion, is the extent of the test requirements for this service. To be able to test this functionality repeatedly, requires the support of several other components. Firstly; storing the email address that is going to be used for testing this service, the ability to start the tests at the correct time and only perform the test for the duration that the commercial is live, the ability to record the transmission times and the results to a log and lastly the ability to alert an engineer at the event of an error.

The above details have presented the actual functionality of the test module and what functionality it controls, what follows is a lower level look at the processes involved in an end-to-end test and the structure of the test module of the basic service.

### 4.5.2 The Testing Process

Due to the way that the scheduling aspect of the Timer class is implemented with in the Java libraries, the Test class has to implement a method called ‘run’ this method defines the aspects of the Test class that are to be repeated. Therefore when an instance of the basic service class is created the constructor sets some initial variables, namely the alert message for that service and the number of iterations. When
this is done the graphical interface is created that presents the details of the test throughout the several iterations and then the ‘run’ method is called. The run method then repeats the following process until the number of iterations remaining is zero; uses the database unit to access the database and retrieve the test email address, packages and sends the value using the RMI unit, checks the status returned and then waits until the delay period expires. If an error is detected at any point throughout this process the email unit is initiated and an email is then send with the relevant details supplied by the Test class. Regardless of status, at the end of every iteration, an entry is added to the error log using the logging unit which is also supplied with all the relevant data relating to the test in progress. This core functionality is replicated through each of the other services implemented with the only differences being the retrieval of the data form the database and the packaging and sending of the data to the remote host.

4.5.3 Error Logging Unit

The error logging unit is one of the most valuable units as it offers the most quantifiable data out of the whole system. It is this fact that makes the error logging aspect of the application so valuable to several different levels of the hosting company; documenting the progress of each service as it runs gives assurances to both management and operational staff. It is the data recorded in this log that allows for statistical analysis and modelling over time to be performed leading to an increased understanding of the performance of the different services hosted. This is of course, is only of interest to the engineering department, however, the logs mere existence provides reassurance to the management that the tests are taking place and that any errors are being detected. Further to this the operational staff has a single document to inspect to check the progress of all interactive services at once.

The logging unit’s purpose within the testing system is to record all the events as they happen and the results retrieved by the testing application it’s self. The results are classified with a simple status codes that denotes the success or otherwise of a single test. Zero represents an error free test and full functionality, then 1 onwards relate to the different errors that may be encountered during the testing process. This status code is then recorded in the log with several other pieces of key information relevant to the failed test and reflecting the technical environment at that time. As is the case with the other units, this logging unit is independent and contains no information of the tests being performed. Therefore, any information that is recorded in the log is passed to the logging unit by a specific testing instance, leaving the logging unit to format the entry and manage the log file.
The functionality of the logging unit is contained within a single Java class which in turn used the file.io libraries to manage a text file. This log could be maintained in many different ways, recording the values as an xml document or within a database, however, none of these alternatives offer any great advantage over the simplicity of the text file in the context of this concept solution. It was, therefore, maintained that a single text file containing comma separated values would form the basis of the logging unit. The functionality contained within the logging class, allows local String variables to be set and then formatted as a new entry which is then appended to the log in chronological order. Other methods are included within this class that provide the ability to manage the log directly. These include; clearing the entire log, removing and editing entries and entering new entries. The same text file managed as part of the back-end testing system is also used by the front-end system when presenting the data to a user, this specific functionality is discussed in the chapters relating directly to the front-end system.

As mentioned previously there are many other values recorded along side the status code and it is these values that add to the value of the logging process. The values are; the time, date, service id, the value sent, the response from the host (if any) and the duration between sending and receiving values. These are all details that may lead the engineer to discovering what may have caused the fault. The time and date are important as it is this information that can be presented to the customers to whom the service was commissioned, allowing for accurate details of when the service encountered the error, which becomes very important in the event of any compensation claims brought by the relevant client. An segment of one of the log files is contained within Appendix G illustrating these values.

### 4.5.4 Email and Text

It is important that an engineer be alerted as soon as possible after the discovery of an error. To be able to perform this as an automated response as part of the testing application is therefore key to its functionality. Two of the most acceptable methods of achieving this were decided to be; email and text message, where an alert would be sent directly to an engineers mobile phone and inbox. After initial investigation it was discovered that implementing a practical text message service using Java was going to be considerably more complex, time consuming and costly[14]. It was, therefore, decided that to implement an email solution would suffice in providing an illustration of this alert functionality, allowing a text message service to be added at a later point if required.
Figure backend, shows the email component as a single unit and conceptually this is correct. It is, however, constructed as a single Java class containing several methods. These methods become available to the test unit upon a new instance of the email class being initiated. The email unit is designed to be independent and therefore, contains no specifics about the test in progress, these values are set using the access methods of the email class. A single call from a testing instance to the method; ‘postMail’ of the email class, containing the specific details of the test as arguments, is all that is required to trigger an email being sent. The arguments that are passed as part of this call relate to the test in progress and personalise the email accordingly. In the case of the ‘basic test the email address of the engineer responsible, the reference number of the test and the message were hard coded into the test class it’s self, the remaining values like the error status code are added dynamically as an error occurs.

The actual process of sending an email in Java using the Javax libraries available, is relatively straight forward. In an effort to reduce spam some SMTP hosts do require that the sending account be authenticated before they will forward any messages. This does increase the complexity of the code slightly but does not detract form the overall simplicity of the process as a whole. The main aspects related to sending an email in this fashion are concerned with establishing the session and building the message. These tasks are performed with the aid of several other classes form both the Java standard libraries and the Javax libraries, which manage the initiation of authenticators, properties, sessions and addresses, as well as, the message content it’s self. A copy of an email that was generated by the email class during the testing phase has been added to Appendix G.

4.5.5 Timer

The role of the timer unit is essentially; to halt execution at the end of a test and wait until the delay duration has expired before performing the next test, when the last test is performed the execution concludes and testing terminates. This is important as performing continual tests would in effect swamp the interactive service with requests and adversely affect its performance. This testing frequency and start time is set as one of the controlling parameters upon the initialisation of the test object of a specific service. The ‘Timer class’ is part of the Java.Util library, it allows a schedule of events to be built up and times, durations and delays to be associated with them. It can be built into any operation that needs the ability to recur in a controlled manor. The scheduling of these testing operations has been extended
to allow for a calendar of tests to be associated with each service, as would be require in a real-world implementation. This scheduling functionality is effectively contained within the same class as the timer functionality as these two operations make use of many of the same libraries. despite being shown as two separate units in figure 4.2.

The ability to schedule different tests for different services and initiate these automatically is a fundamental feature of the overall testing application, as many of the different services that require testing will be live at different times and require different durations of testing. It is therefore important that this functionality be represented within this conceptual architecture.

4.5.6 Database and Properties

The database unit as featured in figure 4.2 is concerned with the management and interaction of a number of tables held in a Postgres database. Each table relates to the different interactive services and the values that are needed to test it. The interaction between the database and the Java class is designed as per the JDBC standard; this allows Java classes to interact with a relational database manager via the use of the correct database driver. It is this driver that bridges the gap between the two different languages which often operate at different levels of abstraction.[31]. The properties file that is associated with the database unit contains all the access methods associated with the database and storage of user credentials. In providing these details in this way it allows the database implementation to be changed, with no disruption to the actual methods used by the testing application. It also allows for different databases to be used for different iTV services.

The functionality of the database class itself contains many different methods, all of which perform relatively simple tasks. Maintaining the database connection and performing queries are the two main sets of operations and these are used repeatedly throughout all the tests performed as no value used is cached during the testing process. The database maintains a simple structure of; a single table per iTV service, this does not claim in anyway to be normalised, however as there is little in the way of relationships between the values held and each value is used in an independent manner, this was not of great concern at this conceptual level.
4.5.7 GUI

As part of the concept solution a number of GUI (graphical user interfaces) have been implemented to show the processing in real-time. These are essentially for demonstration purposes; however, they do expose the details of the tests and the values being transmitted to an observer. This in addition to information printed to the command line, provides a comprehensive illustration of the status of the tests and distributed architecture. As shown in figure 4.2, the GUIs are located relative to the information that they display; i.e. the GUI as shown adjacent to the testing unit displays details of the tests; just prior to transmission, the actual test values, the iteration number, the delay duration and general simulated layout of the iTV service. The GUI adjacent to the RMI server shows the status of the transmission as it is received. The first of these GUIs is essentially a representation of what would be displayed on a television as the tests were conducted in a real-world implementation, in, a somewhat cruder fashion. However, being able to display the values at both sides of the distributed architecture, does go some way to proving the integrity of the simulation.

The GUIs are implemented using the ‘Java Swing’ framework for developing Graphical interfaces. These libraries are used as the basis of the GUI classes as implemented within this architecture. To allow the data to be passed from the testing application several methods have been implemented, that continually update the various fields of the Two GUIs. Each testing instance uses the functionality of a basic GUI and then extends this functionality according to the needs of each specific test. The Basic service for example, inherits the basic structure from the template GUI and then extends this to incorporate a field to hold the email address, that is collected from the database by the testing application and then sent to the RMI client (remote service). This value is displayed in the testing GUI and the received value displayed in a similar way at the remote side of the architecture.

4.5.8 Simulating the Distributed Platform

The simulator takes the form of a remote messaging implementation. Essentially it’s role of is to simulate the distributed element of the real-world architecture, this involves providing the testing application with an interface to test, as well as, representing aspects of iTV services functionality.

As described previously the RMI system is a remote messaging system that allows the methods of
distributed objects to be accessed as if they were local to the method of the calling object. [12] In choosing the JavaRMI implementation of this remote object system, a certain element of abstraction was able to be maintained throughout the testing process. This not only aided in the implementation but also purposefully reflected the same level of abstraction, that the middleware would afford a real-world implementation within the software architecture of a set-top box.

As illustrated in figure 4.2 the RMI architecture represented several iTV services. The RMI servers extend the interactive services functionality through various methods and the RMI client acts as the gateway to accessing this functionality for the testing application. Each RMI client contains the details specific to a single RMI server which in turn represents a single iTV service. Therefore the testing unit specific to that service uses the relevant RMI client as its means of communicating with that service. The functionality of the RMI client is only concerned with; packaging the effective iTV page, identifying the server, connecting to the server and transmitting this package to the server. It then passes any values returned from the server back to the testing application in way of an acknowledgement. This although simplistic implementation, captures some of the key elements present within the interactive return path system. There are although several weaknesses associated with this implementation, which are discussed separately within the evaluation section of this report. However from a pure conceptual viewpoint of functionality the RMI system implemented to represent these several iTV services provides an excellent framework for the testing application and also provides the flexibility to manipulate the testing environment and therefore effect the results received.

Due to the reliable nature of the TCP/IP protocol stack as used throughout many computer networks throughout the world, it is very unlikely that a remote message generated by the RMI client is not going to reach the RMI server running on another machine within the school of computing laboratories (University of Leeds). It was, therefore, realised that an element of unreliability, would have to be added in a controlled manor within the RMI server implementation itself. In doing so the rate of which errors generated throughout the testing process could be altered and the results observed. To achieve this added functionality of the RMI server, additional methods were added that maintained a Boolean variable that determined weather a simulated error would be generated. The remote method used to simulate the transmission of the interactive services values is its self a Boolean method and therefore, due to the way that Java works has to return a Boolean variable. If the test value is received then true is returned to
acknowledge this fact, however, if a error is generated then the value false is returned. This false value is ignored by the RMI client as though no acknowledgement had been received, and the test application left to time-out. The probability of an error being generated is set at 0.01 and is determined using a random function and a comparison between it and a fixed value. The time-out feature is incorporated within the RMI client and is set to a figure that reflects the speed of the LAN of the School of computing laboratory. Therefore rejecting any tests that take too long and those where no value being returned, these values in a real-world implementation would be indicative of the infrastructure of a real-world platform.

4.5.8.1 Packaging of the values

Finally the packaging of the interactive page’s values, with respect to the middleware environment of a set-top box architecture, is a concept that the RMI client and testing application have tried to simulate. Through using the ArrayList class of the Java standard library, the values were able to be packaged into a single object and passed from the client to the server. This somewhat abstract representation of the process of sending an iTV pages values in a single package, is reflective of the similar way that a HTML page posts the values of a HTML form as a reference-able value, within a single identifiable group. which is in turn similar in many ways to the way that an interactive application would send the values of a form to the remote host. In reality however, the middleware layer is merely packaging the values and passing it to a lower level of the software stack[10], but as an end-to-end conceptual representation this captures the essence of this approach.

4.5.9 Summary of the units developed

The preceding sections have presented the functionality of the various units developed as part of the conceptual architecture, for further pictorial representations of the developed system, a verity of schematic drawing have been included along with brief descriptions as part of Appendix H.

4.6 Front-End System

The front-end system as suggested relates to the user configuration element of the solution architecture. This is essentially related to the capability for a user to specify the test values used for each service and the presentation of the results in real time. It is an element of the architecture that is very important
to the management and engineering department of the service providing company as discussed in the problem definition chapter.

As is the case with many front-end systems it was decided to implement the user interface as a series of dynamic web pages. This again has the same motivations as the back-end system in that its role within the conceptual solution is to present the functionality that would be expected of such a testing application and to investigate the interaction needed to interface the front-end system with the results and data gathered by the back-end processes.

![Diagram](image)

**Figure 4.3:** The front-end architecture of the conceptual solution.

The front-end system was developed as a collection of web pages and therefore a suitable language for developing dynamic content had to be selected. It was decided that this should be JSP as it was a language that could share some of the functionality of the back-end classes to realise its own aims. JSP is a java based language and is very powerful all though somewhat complex[31] Essentially as per the diagram the same database and log files are used to share data between the front and back end systems. As the data is recorded by the back end system the front-end system can present this as part of a web page relating specifically to a single system.

The solution developed was implemented as a secure collection of Dynamic Web pages that allowed a user to log in, select a service and edit the test values associated with that service, these values were then recorded by the JSP pages to the database where they resided until the next test was carried out. This would then retrieve the new value from the database and use this as part of the test. From the same
initial page the user could select to view the error log which would be presented on screen in its raw form displaying the status of all the values recorded for the values mentioned in the previous chapters. However, if the user selected to graph the results that a chart was dynamically generated and presented on screen showing the transmission durations of the testing application for a specific service. This allowed the differences transmission times to be observed.

4.6.1 Using JSP Using Eclipse

The J2EE environment was used within the eclipse environment to allow all the dynamic content to be managed. This involved managing the web container that the JSP files were contained in and the raw HTML files. Being able to manage these in a single place was a great advantage to the project. As well as using JSP both Java script and CSS implemented as part of the web system this allowed for the best practise of web development to be observed as stated by the W3C.

This although brief section, shows some of the functionality that the web portal offered to support the testing application, this would of course be developed to be a main aspect of the implemented solution as it is through this that the user can interact and set up the system for new services and modify existing service.

All the functionality as relating to the web system stated in the aims of the project has been implemented, with the addition of the security aspect as this was thought to bring the web system closer to the level of service that would be required form such a system.
Chapter 5

Evaluation

5.1 Introduction

It is within this section that this development is evaluated. Using several topics as a framework for the evaluation process, these aspects can be presented in a systematic and justified way. As this project, in its nature, a more challenging solution to evaluate, the basis for the majority of the claims made will come from several meetings and discussions held with professionals of related fields. It is essentially, the role of this section to provide a critical assessment of the relative achievement of the aims and objectives previously set out by this project and the following sections will detail the findings of this process.

5.2 Framework for evaluation

In accordance with the aims and objectives as stated in chapter 1, one of the key objectives was to investigate the possibility of creating a solution to interactive TV application testing, through the generation of a concept solution. It was realised that this concept solution would also have to incorporate the development of a simulator to represent the distributed nature of the related platforms, with the reasons for this having been detailed in section 1.5. It is, therefore, key to the success of this project that the functionality of the problem domain be replicated in a suitable way allowing the concept solution to mirror the real-world environment.
In order to evaluate this project the following issues will be considered:

- The development process itself; whether it has been developed in an appropriate manner and observed best practice relevant to software development.

- The solution’s functionality, evaluated in its own right; does it work and does it do what is expected?

- The functionality with regard to an appropriate solution; is the functionality appropriate? Does it solve the problem and fit the task?

- Is the solution conceptually correct; how well does the concept replicate the real-world environment?

- Is the architecture developed appropriate to such a testing application? Will it support the tasks of the application?

- The overall strengths and weaknesses of the solution as a conceptual development.

To support the evaluation of the above issues a number of comments and opinions from several external professionals will be included. These external parties have differing experiences of the various areas that this project comes into contact with. These include; set-top box software development, distributed systems, interactive application and systems architecture and operations management.

5.3 The development process itself

The development process observed the methodology as presented in chapter 2. This methodology was constructed from the elements thought to be complimentary to the development of this project. Retrospectively, the adopted approach included both supportive and practical elements that allowed for the smooth and continual development of the conceptual solution.

There may well be aspects of any development process that can be criticised as not following the true guidelines set out by the methodology, however, the emphasis was identified as being on the agility and flexibility of the process and not on the heavy documentation and strictly procedural development processes. The core reason for this project needing to be so dynamic is due to the large amount of research that paralleled the majority of the development process, this related to the implementation of
each method and the best way to represent the real-world functionality it simulated.

Of course performing more research prior to beginning the implementation would indeed improve the integrity of the current implementation as it stands. However, as is pointed out in section 1.5, this was somewhat related to having to change tack mid way through semester one. Despite this it is an iterative process therefore, functionality is added in an iterative nature, gradually improving upon the implementation until such a stage is reached where the task is complete. Relating this to the development of this conceptual prototype solution, the development process may continue until the integrity of the very low level aspects of a specific delivery platform are realised. This was not however, the objective of this project as it stands and therefore the development process adopted suites and supports this project in reaching its current status.

From a more technical angle, the software elements of this project have being designed and implanted in accordance with the three tier architecture; presentation, application and data. [3] As is the normal approach for object oriented software design, this refers to both the front-end web based system and back-end testing application both of which have observed this approach of segmenting code according to its functionality. Commenting blocks of code throughout the various classes is also a best practice guideline that software developers are expected to observe and a feature that is present throughout this software. Relating to the commenting of code it is often expected that a standard format is observed throughout an entire project to provide consistency and easy reading for both the developer and any third parties, this is an aspect that could be improved as commenting in an application of this nature is vital to future iterations and extensions.[11]

5.4 The software functionality in its own right

Testing a piece of softwares functionality has been realised as an important aspect of many development methodologies and often features as a specific stage of the software engineering process. The testing process can refer to many different forms of testing and has many associated techniques to support them[3].

One method of developing software is to take a test driven approach, this parallels the development of the actual functional software with additional code specifically implemented to check the correct op-
eration of software units. A unit in this context refers to the smallest possible piece of testable code. In doing this the number of bugs incurred by the system is reduced and exhaustive testing time is reduced significantly.[31] However, this was not an implementation that was adopted by this development process as the assurance of a bug free system was not of the utmost importance when compared to the functionality that the system illustrated. It is most likely that case that this implementation be regarded as a throw away prototype. In this case the functionality would be translated into an actual working implementation and it is this separate development that is most likely to adopt such a vigorous test driven development approach, as in this case, developing a system that is free of bugs is going to be significantly more important.

As touched on above, exhaustive testing is an alternative; however, this is said to never realise the full extent of the compromised elements of the system and as the name suggests is an exhaustive process.[31] This is generally what the testing phase of the development process consists of, along with checking the functionality of the system to ensure that it has realised all functional requirements as per the customers request, this process is referred to as acceptance testing.

It is these two testing methods that have been implemented during the development of this project and as per the project plan (Appendix I), taking place at the end of every iteration, as well as, a more comprehensive testing phase taking place at the end of the development. This process may not catch all the bugs, but as the onus is on the illustration of specific functionality, the project was more akin to this style of testing. It could therefore be said that this project did not observe industry best practise with regard to testing; however, it has a strong argument to support its case for not doing so.

The acceptance testing as used within this project; basically mapped the requirements identified as key to the applications functionality as shown in figure 1.1 to the actual abilities of the simulator. This use case driven development is often an inherent side affect of using an agile development methodology[4] and therefore acceptance testing becomes an almost invisible part of the development process its self. In completing this testing process, it can be said that; the functionality of the system developed does in fact demonstrate the functionality that it was designed to provide, i.e. each unit does provide the functionality expected of that unit.
5.5 Is the functionality appropriate to the solution?

In evaluating the appropriateness, of the functionality accredited to this testing application, previous experiences were drawn upon to allow a comprehensive list of requirements thought to be important to such a testing application. These as presented and discussed throughout the report, have also been presented to an operations manager of a company responsible for hosting such iTV services as those relating to this project. This has allowed these proposed functions to be confirmed as important features that should be included within the development of such a system.

The thoughts of this professional namely Time Wilson, have been included as part of Appendix B and highlight some of the following aspects; That the testing of these interactive services has unusually high priority due to the revenue that they provide, that it has being observed that tests were sometimes not being performed as per instructions and that tests were in no way documented despite a structured testing procedure being implemented. There are many other points raised in this letter that highlight the importance of testing these interactive systems and the advantages that the company would receive through this been an automated process.

When the architecture was presented to an engineer from the same company namely Peter, Starling, his remarks echoed many these points and added; that the having incorporated the ability to time events and to be alerted immediately upon the event of a fault occurring would offer significant advantage over the existing system. More of peters opinions have been recorded in Appendix D.

The solution as implemented; addresses many of these issues and also adds other functionality that relates to other divisions of the company structure. As presented in figure 1.1 the different functions of the application address different levels of the business hierarchy and the justification for these different functions are presented within chapter 1. Upon presenting the proposed functionality to a member of the University of Leeds research and teaching staff, namely Dr Karim Djemame, how specialises in distributed technologies, his comments supported the inclusion of these additional functions and a more comprehensive reporting system. There is also the functionality of the front-end system that allows teh plotting of graphs to represent the transmission times for the data sent over the distributed network. This as well as, providing the ability to modify test values specific to a interactive service and access the logs all form a single series of web pages full fills the requirements of the this side of the system.
In summary the functionality chosen and implemented as part of this solution does suit the nature of the application and has been observed as such by bodies external to the development of this project.

5.6 Is the solution conceptually correct?

This is the most complex of topics to evaluate and as was the case with the previous section to do so in a meaningful way requires the opinions of several third parties. Therefore the basic conceptual decisions will be presented followed by the thoughts of both Stuart Ingram, a software engineer for Pace Microsystems and Dr Kirim Djemame who was introduced in the previous section as a member of the University of Leeds Research department.

Certain aspects of this simulated architecture were implemented specifically to illustrate the inherent weaknesses of the architecture as well as those that were implemented to illustrate desirable functionality. This allowed a negative part of the implementation to be turned into a positive aspect of the demonstration. One of the main architectural decisions relating to this simulation of the return path communication was how specific should the representation be. Setting this level of abstraction relative to the likely real-world implementation was difficult to calculate. However, based on the fact that this aspect on its own is such a large task, achieving the current functionality will suffice. This may be identified as a possible area for expansion along with other specific pieces of simulation that may need tightening up on.

Both Stuart and Kirim Djemame identified finding the correct level of abstraction as been a key point to this projects value. In defence of the conceptual solution, the return path as implemented with the RMI messaging system may not replicate the exact functionality of a specific platform but does go a long way to highlighting many of the issues associated with communicating with a remote application. Issues like; connection oriented vs. Connectionless links i.e. TCP and UDP, how the service has to be identified prior to data being sent, reflected in the way that RMI uses a registry of available remote object servers.[12] There are many other justifications for why using RMI to replicate the distributed nature of the iTV systems is appropriate over alternative like, low level socket programming. Many of these are discussed within the architecture and background chapters of this report.
Kirim Djemame offered the opinion that using JavaRMI was a good choice and that it provided good functionality for this level of illustration. Stuart remained cynical that developing a simulator would not in fact provide an accurate enough representation of the various platforms used. Essentially there are so many new opportunities for more advanced return path implementations, as presented previously, like wireless links and ADSL connections, that simulating a specific platform at this stage could be considered to be short sighted.

Relating to the level of abstraction used to represent the whole testing system, this is limited by the fact that the set-top box hardware needed to be able to produce an actual onboard testing application was not available as detailed previously and that the exact nature of the implementation the at the testing application would be translated into is not known, therefore is has been decided that the most likely would be a middleware implementation with the testing application residing on a computer and the functionality been preformed with the aid of a development box. In essence the architecture may not represent the specifics of the various platforms but it does capture the majority of the issues that relate to their nature. Kirim Djemame commented on the architecture in its early stages as providing; a comprehensive conceptual representation of the architecture and its associated functionality.

5.7 Is the architecture developed appropriate to such a testing application?

This is again a section that is taken straight from the aims of the project and therefore, an aspect that should have been addresses. Upon presenting the designs for the supportive architecture to these various parties, all the comments received, were that this area of the project was justifiable. That the elements chosen to support specific aspects of functionality was sensible and that it was defiantly achievable within the environment of broadcast television. Both Kirim Djemame and Peter Starling both agreed that this aspect demonstrated a simple and implementable solution. It was also remarked by Stuart Ingram when the early designs were presented, that there was no problem with the supportive aspects of the design.

Based largely upon the these comments made early on in the development process, this area of the project has being the most fluent to develop and is justifiable as been a suitable supportive architecture for the functionality needed by the system. However, referring back to the front-end system, this
contained no conceptual features, all of the functionality performed by this system as documented in
the development section was as it would be in the real-world. The essential functionality of this system
was to provide the ability for a user to manipulate test results and check the progress of services. All of
which is implemented and tested as providing this functionality as per the project requirements. As this
section requires no conceptual abstraction less interest has been payed to it by the external parties.

5.8 Over all strengths and weaknesses

The overall strengths of the system are that it has the ability to act as a communication tool, that aids
with the identification of problems and obstacles that developing such a testing application regardless
of platform is likely to encounter. It does perform all of the functionality that was originally set to
achieve and it preforms this well. Regarding the discussion over the correct level of abstraction there
may be other issues that have limited any further progression into specific aspects of the simulation.
Obviously noting the time constraints of the overall project there is still the prolonged starting phase
which hampered the development getting off to a rapid start.
Chapter 6

Conclusion

6.1 Qualifying the final conclusions of the project as a whole

After following the analysis and development process from its conception and having developed a working conceptual solution. This report concludes by stating; that to develop a solution as presented within chapter 4, would not be a practical implementation of such a testing application as outlined in chapter 1. This is due to the complex nature of the interactive applications, realising that too much of the functionality was reliant on aspects that are not guaranteed to be implemented across all applications. Also as both the middleware market and return path technology looks poised for further radical development. To commence a real-world development process based on this existing architecture would produce a solution that very soon will be out of date. This, along with the need to further research the exact capabilities of the functionality that can be achieved form contemporary set-top box hardware seems to suggest that an alternative solution, implemented in a different area of this domain needs to be investigated.

As far as the requirements of this project are concerned, many if not all of the aims and objectives have been realised, with many different aspects of the broadcasting environment having been explored to gain an understanding of what capabilities are available from the interactive platforms as they stand today, as well as, getting an in-site in to the technological advances that are just around the corner.
As identified within the evaluation chapter there are many different ways that a solution to this problem could be addresses, with the outcome of this project hopefully providing sufficient groundwork for other projects to take a more detailed look at specific aspects of the real-world architecture.

A story of two tails; a conceptually appropriate architecture, however, implemented at such a level of abstraction, that some of the needed detail was not realised. Whether this is due to the constraints of time or ability, there is defiantly another iteration of the project remaining, to take this to a more detailed level where a solid assumption could be made, however so far the possibilities of a real-world implementation on the existing satellite platform are looks shady. Though, the UK and US cable platforms look to have many more sympathetic aspects relating to the testing architecture presented her. Watch this space!
Bibliography


Appendix A

Introduction

The information presented below is a personal account of my experiences relating to the development of my final year project. There are a number of sections each relating to different aspects of the FYP process, firstly an all over account of the final year project as a whole, essentially what was good and what was bad, secondly, several specific aspects of the project as I encountered them I believe to be important to future development projects and lastly a series of general observations made throughout the entire final year project experience that I will taken more care to adhere to, when beginning a similar development process in the future.

Overall experience of developing a project

I found the final year project as a whole, enjoyable, it was something that I could really get my teeth into explore some computing technology in my own way and at my own pace. As the project I undertook was of my own specification I was able to create the aims and objectives with relative freedom, incorporating the aspects that I wanted to investigate. It was a topic that I had wanted to look into for some time and I was pleased that I got the opportunity to do so as part of my final year project. It was not, however, a completely smooth journey from conception to implementation and it did contain a lot of hard work and motivation. Especially during the parts of the semester where the work load really stacks up.

As a whole, the process has been a valuable learning experience and I have had the opportunity to
work through some tough problems and generate some satisfying solutions. I feel that the solution I have developed does in fact satisfy all of the aims and objects set out in the mid-term report and I am pleased with the efforts I have made.

**Specific aspects of my project**

Throughout the process of specifying and developing the solution to my final year project a number of problems were encountered. These in the large, related to time management and on the whole were not entirely my fault, however, the lack of being prepared for such eventualities was my fault. I am referring to the issue of my attempt to work with Pace Microsystems and the time that this attempt cost me.

The main problem was maintaining timely correspondence with the contact I had at Pace. Sometimes several days could pass by before I would receive a reply to a previous email. This; as it was seen then, was not necessarily a problem as it was early in semester 1, however, when the cooperation was finally realised to be not realistic, a significant amount of time had been spent trying to set up meetings and discuss the project specifically. Leading to early semester 1 becoming mid-to-late semester 1, this meant that the project specification had to be altered and the aims and objectives adjusted. In all seriousness, with hindsight, it may have been more appropriate to have selected a completely new project, however, too much time had passed by this point.

This is the first and most significant issue that caused me problems. In realising this sooner and not becoming expectant of the hardware Pace would have provided, more options would have been available when things began to go south. Despite all this negativity, I did gain a lot of good information from the meeting at Pace regarding projects they were working on.

Basically, what I learnt from this was that building in contingencies is extremely valuable to any project and having time to be able to consider your options equally important.

The only other issue I had relates again to specifying my own project. In doing this it led to me having great difficulty in finding people within the department that has any experience of the topic that I had selected. This made every issue I had relating specifically broadcasting and interactive TV difficult to qualify. This was less of an issue that the prior as there is a reasonable amount of documentation avail-
able on the web relating to middleware and new iTV developments. However this is another issue that is worth bearing in mind when selecting FYPs.

**General advice for Final Year Projects as per my experience**

Lastly I will present a brief summary of the advice that I would like to have heard prior to starting my FYP to ensure that my project stayed on track throughout the two semester development period.

One of the most important pieces of advice that is always treated with a blasé attitude is to start early, making a good start early in semester one would really have been beneficial to my project in allowing me to manage my remaining time in semester two more appropriately. This in my case was inhibited by the slow correspondence between pace and myself, however I did not waist this time as I spent the majority of semester one researching the subject area. It would though, of been good to have been able to make a start on the actual implementation prior to the Christmas break. Second to this, building a good relationship with your supervisor is a good idea as they have been through this FYP process several times and are likely to offer solid advice. I was fortunate that my supervisor always had time to discuss the problems I had with Pace and advise accordingly. Another important point to observe early on in the FYP process is; when selecting you project, look for something that is not only contained within a subject of interest, but also a project that has functional and measurable outputs. This makes the whole project more meaningful and a lot easier to document within the report. If specifying your own project be very wary and only select a topic that you have some existing knowledge of or can spend the additional time researching. Also in specifying your minimum requirements regardless of project ensure that you really can achieve something and that your are not going to find your self up against the wall in the second semester when it is too late.

All of these points are not only issues that I have realised to be important to the success of a project but that will also make the whole experience a lot more enjoyable.
Appendix B

Notes from Tim Wilson

Contained within this appendix is an email and letter from Tim Wilson, an ex-employee of YooMedia Plc. Tim’s role within the company was as the transmission manager, looking after the day to day running of the transmission area and its staff. This also encompassed both the area where the manual testing of interactive TV application was performed and the staff who performed it.

The contained documents refer to Tim’s thoughts on various aspects pertaining to the development of an automated implementation of this testing process.
Hi James,

Please find my response to your proposal to develop an automated interactive testing app.

Kind Regards,

Tim Wilson
Dear James,

Re: Interactive Application Testing.

Further to your proposal for automated interactive games test software, here is a brief summary of how and why this tool would be useful.

The interactive applications require frequent testing to ensure they were working correctly. The applications create a direct revenue source for the Company and so the working status of the application gives the testing an unusually high priority compared to the actual environment it’s operated within.

The Company asked highly skilled Transmission Controllers to step through the application once every ten minutes. This created a great deal of low morale and subsequent high turn over of staff. Owing to the simplistic but 24/7 timing of the tests, the company could not warrant a dedicated member of staff to carry out the tests. However, the tests were seen as menial and diluted the concentration and diligence away from the actual job in hand; looking after the conventional Broadcasting.

The Company found that some staff did not test the applications frequently enough, sometimes at all. This was put down to prioritisation but also due to a lack of interest and forgetfulness by the operator. This caused the Company to a) enter into disciplinary procedures with repeat offenders and b) create a time checking system that the Operator had to complete on each test.

The down side of checking system is that works to your lowest level of employee; it’s not generally needed across the department and the better staff don’t need to complete such a process. This therefore has a negative impact on moral and reduces the perceived skills and career projection of the role.

I strongly feel that at the time of my management of the Transmission area, if an automated system was designed and installed, it would have been a major benefit to the company. For an example, today there would be a higher skills base in the department. The department would have not gone through as much low level administration, management time would not have been spent dealing with the fall out this created and most importantly, the Company would have saved considerable amounts of lost revenue owing to undetected down time.

Tim Wilson - 07771 534 895
Studio and Transmission Manager
Appendix C

Stuart Ingram

Stuart Ingram Head of testing department for Pace Micro based at Saltaire. Presented here are the emails exchanged between Stuart and myself, the original brief that was sent to Stuart. Outlined below is a summary of the notes taken during the meeting I attended at 16:00 Monday 20th November.

General Notes:

Main concern was with the return path implementation, how was this going to be implemented. Was it going to be accurate?

After it was realised that the original brief was not going to be realised, Stuart was very supportive of the alternative that I had conceived.
Issues over accuracy of the separate components were his main concern, however the conceptual architecture presented to him, he thought was to be of a sound nature.

Functionally very well specified.

He advised me to think of some lower level details. TCP architecture etc.
The initial aim of the project is to develop a system that can replace a manual process that is in use by at least one broadcast satellite centre; the process being that of continually checking that the interactive services are available to consumers, and that any interactive forms present within the application are both transmitting and receiving their content. Although this sounds like a trivial matter for the companies that are responsible for the content, it is of great importance as these companies only exist as a proxy between the commercial company and the consumers. There is often a specified agreement between the two companies detailing the amount of agreeable down time, if any, and what the penalties are for any breach of this agreement.

The system that I had imagined would solve such a problem will involve some sort of application running on a set top box (either a simulated STB or a Development box) that is able to take values from a user via a web front-end and then complete the interactive form for which these values have been entered. This process of entering values may be duplicated for several different services for which the host company is responsible. An example of this could be an information request service running during a commercial at a specific time of day and a registration form active during a specific program on a different channel. The best use would be for testing services that are available twenty four hours a day on numerous quiz channels; it would generally take a while for these forms to be spotted if they were not becoming active or not transmitting the information in the correct manner. Also this sort of implementation would be able to run repeatedly for a given duration and at a given delay.

My test system would be able to pick up the values from a central store and complete these forms every time the service was programmed to become active and then send a test transmission of the data and record the response. It could be possible to add additional features to the system, for example timing the duration between transmitting the form and getting a response, and then plotting these results to show deviations in the time taken for the transaction to complete, which could be used to prove that the system is working reliably.

The results for each service could then be viewed on a webpage which would display the status, and if there were any errors these could also be sent via email or text message to alert a duty engineer allowing a near-instant notification of failure.

I am aware that there are some commercially available solutions that may perform a similar task, however I have not had any contact with these and do not know the extent to which they would solve the stated problem.

This problem comes from a real environment where I have observed this as an issue for companies hosting interactive form services delivered over satellite infrastructure.
In Summery
My current understanding of the problem suggests that the system will involve a web based front end that allows a user to enter values and see any results that have been generated. These values, and results, will be stored in a database which will interface (through the use of XML documents) with software developed to upload these values to an interactive television form and monitor the transmission of these forms as well as recording any errors that may be received from the box.
Stuart

I have attached the brief as you requested I hope that is serves the purpose if you need any more information then please let me know and I will fill in any gaps.

I hope it reflects where the problem came from and my maybe simplistic solution to solving it.

Many thanks once again for you time

James Hudspeth
Stuart

I hope you got my email with the proposal attachment I know that you were away this week and may not have had a chance to read it but if there are anymore details that you need then I can send these on to you just let me know.

I was wondering if I could possibly arrange to come and discuss the proposal with you and get any ideas that you may have had when reading through it. I was hoping that this would be possible over the coming week (possibly Friday) as I have a report to draft on the future direction of my project detailing more specific aspects of the technology that I am likely to be using.

If this next week is not practical for you then just let me know when would be best for you.

Hope to hear from you soon

Many thanks

James Hudspeth

scs4rh@comp.leeds.ac.uk
Alternative email address
Hi James;

Apologies for not getting back to you sooner; I didn't get chance to read your proposal until over the weekend as I was working away in Ireland.

I am back in Dublin this week too so unfortunately I won't be able to meet with you, however I am free on Monday morning (20th November) if that time is suitable for you? Would 10am be ok? If you let me know I can block some time out in my calendar.

As for the proposal it seemed fairly sound, however there are a couple of things that I would like to discuss with you and depending on your ultimate aim for your project, perhaps propose a slight change to the idea to tailor it to a real life, current issue that is a hot topic for the industry at the moment and which uses very similar principles and technologies.

Thanks;

Stuart.

-----Original Message-----
From: james hudspeth [mailto:hudspeth_jr@hotmail.com]
Sent: 10 November 2006 17:54
To: stuart.ingram@pacemicro.com
Cc: markw@comp.leeds.ac.uk
Subject: project proposal

Stuart
I hope you got my email with the proposal attachment I know that you were away this week and may not have had a chance to read it but if there are anymore details that you need then I can send these on to you just let me know.
I was wondering if I could possibly arrange to come and discuss the proposal with you and get any ideas that you may have had when reading through it. I was hoping that this would be possible over the coming week (possibly Friday) as I have a report to draft on the future direction of my project detailing more specific aspects of the technology that I am likely to be using.
If this next week is not practical for you then just let me know when would be best for you.
Hope to hear from you soon
Many thanks
James Hudspeth
scs4jrh@comp.leeds.ac.uk
Alternative email address

Click here to report this email as spam.
Hi James;

Would 4.30pm on Monday be ok for you?

If you can let me know I'll put it in my calendar.

Stuart.

-----Original Message-----
From: J Hudspeth [mailto:scs4jrh@comp.leeds.ac.uk]
Sent: 15 November 2006 12:36
To: stuart.ingram@pacemicro.com
Subject: project proposal (meeting)

Stuart

Sorry it has taken me a couple of days to get back to you

I have unfortunately got a lab session scheduled for 10:00 on that Monday morning and the earliest i would be able to get a way form university would be 15:00.Therefore if you would still like to meet on Monday i would be able to be with you in Saltaire at 16:00.

If Monday is no longer possible for you then i would be able to be in Saltaire on Wednesday from 13:00 onwards if this is any better.

many thanks
James
Stuart

Thank you again for your time on Monday it was very interesting to get a glimpse of what goes on at Pace in Saltaire. However I am sorry to say that it does not seem to be possible for me to get involved with the project you mentioned to me when we met last week. Some of the reasons that have come to light I discussed with you on Monday despite hoping that some of these could have been negotiated I have been advised that to get involved with a relatively new project (with regard to the university timetable) at this stage in the final year project calendar would be not recommended and that I should stay on course with the project I outlined to you. This of course will have to undergo a few amendments before it is fully feasible but I am hopeful that these will be simple enough.

Thanks again for your time, I took many valuable thoughts away with me and am still thinking about solutions to your problem.

James Hudspeth

PS - If I have any queries at a theoretical level I hope that you would not mind me dropping you a quick email in the future.
Appendix D

Peter Starling

Notes reflecting the comments and opinions of Peter Starling, Head of Engineering at YooMedia, Wapping, an interactive solutions company.

Outlined below are a few comments that were offered during a phone conversation held between Peter and myself regarding the architecture that I had previously sent to him. The conversation took place at 13:00 March 20th 2007.

General observations made by Peter:

The functionality that is documented to be implemented was thought to be very valuable, especially the reporting and alert systems.
Some good management functionality.

The overall architecture is practical and not beyond the means of the company to implement.

On a more technical level, the functionality provided by the set-top box should be confirmed to be available, as some of the functionality relies heavily on this being available.
An overall comprehensive conceptual design, highlighting some very interesting issues.

Would need to be considered in lower level detail before it could be thought to be commissioned.
Appendix E

Development Box

The enclosed information relates to an example OCAP development set-top box available for purchase from ADB. The STB’s functionality is presented along with a section relating to the nature of ADB’s business. Also includes to the rear an article presenting a move by Pace MicroSystems to implement OCAP middleware on their new set-top box range.
About Advanced Digital Broadcast

Advanced Digital Broadcast (ADB) provides a diverse range of products to the worldwide digital television industry. We supply digital set-top boxes across all television transmission platforms including cable, IPTV, satellite and terrestrial.

We deliver technically superior products, service and support, in as short a lead time as possible. Our products are universally recognised as amongst the industry’s most technologically advanced and reliable both for current and future generation broadcast technology.

Our customers include digital television operators, consumer electronics manufacturers and distributors of digital set-top boxes. Key to our success is our close collaboration with our customers as well as technology partners and industry regulatory bodies. We are an expert in system integration, incorporating the industry’s leading middleware, conditional access and hardware technologies.

Our business combines the technical excellence of the broadcast industry's leading engineers with a highly-experienced management team who have been operating within the international digital television industry since its inception. Our experience, proven ability to develop and deliver products ahead of the competition, along with our spirited and enthusiastic business culture, have guaranteed our business an enviable reputation as one of the most advanced and trustworthy partners of choice for new technology.
Set-top box features and capabilities include:

- Advanced hardware design
  Based on Broadcom BCM7111 processor with integrated DOCSIS® modem, 250 MHz RISC MIPS 32 microprocessor and 32 bit graphics capability
- Robust Functionality
  Firmware optimized for full processor capability, including 32 bit graphics and robust MPEG and DOCSIS® software implementation
- DOCSIS®
  Two way support via integrated DOCSIS® modem and Ethernet port
- Linux OS
  Linux STB OS with integrated IP stacks
- OCAP™ middleware
  Seamless integration of OCAP™-based middleware from Osmosys

PC software tools include:

- Application Upload
  A software upload tool to enable off-network uploading of application software from a PC via set-top box RS 232 port
- Application Debugging
  A software debugging tool to provide output messages that aid in application software debugging
- Set-top box Firmware Upgrade
  A firmware upgrade tool to permit future set-top box firmware versions to be downloaded via STB RS 232 port

ADB-3200C-DEV
OCAP™ development set-top box kit

ADB's 3200C-DEV OCAP™ development set-top box kit is a high performance, OCAP™-based digital product that can be used both for application software testing and for end to end OCAP™ software lab testing.
set-top-box technical specification

System
- Digital Front End: SCTE compliant,
- Output signal: NTSC
- DOCSIS® 1.1 modem platform

Software
- Linux OS
- OCAP™-based middleware

CATV Tuner
- Frequency range: 54 – 862 MHz
- Modulation type: QAM 64; QAM 256

Video
- MPEG-2 Main Level @ Main Profile
- Formats: 4:3, 16:9 with Pan & Scan and Letterbox
- Graphics: 720x480

Audio
- Dolby™ AC-3 pass through S/PDIF and down mix.
- MPEG-1 Layer 1 and 2

System Resources
- Main Processor: Broadcom BCM7111 250 MHz
- MIPS 32
- 32 bit graphics
- Dual frontend for broadcast and data

(DOCSIS®) transmission
- Memory:
  - Flash - 32MB
  - DDRAM - 128MB

Indicators
- Multi-color LEDs
- Power - Green/On, Red/Standby
- Remote
- Message
- Upstream
- Downstream

Panel Connectors
- Cable In F-type connector
- Composite Video output with OSD - 1xRCA
- Stereo Audio Variable - 2xRCA
- S-video output
- S/PDIF/Optical
- Ethernet interface
- RS 232 port/9-pin D-sub
- Power in connector

General Data
- Operating temperature: 41-104°F
- Relative humidity: 95% maximum
- Supply voltage: 110V AC, 47-63 Hz

Other Features
- FCC and UL approved

Accessories Included
- External power pack
- Remote control
- Baseband A/V Cable

DOCSIS® and OCAP™ are trademarks of Cable Television Laboratories, Inc.
ADB Inc. reserves the right to change, modify or enhance this specification at any time without prior written notice.
ADB Inc. is a member of ADB Group (www.adbholdings.com).
© 2006 Advanced Digital Broadcast
June 15, 2006

Pace Implements CableLabs' OCAP DVR Specification

--Has Formed OCAP Partnership with Alticast

UK set-top box manufacturer, Pace Micro Technology, says that it has developed a DVR that is based on CableLabs' OCAP DVR Specification (OC-SP-OCAP-I02), and that it will demo the box at the upcoming SCTE Cable-Tec Expo (takes place in Denver, June 21st-23rd). The company says that it will use the OCAP DVR specification in all its current and future OCAP DVR's, starting with its flagship, dual-tuner Tahoe TDC775 HD-DVR. To further its OCAP efforts, Pace Micro has partnered with South Korean OCAP specialist, Alticast, the first company to deploy OCAP commercially. As part of the partnership, Pace has integrated Alticast's OCAP middleware solution, AltiCaptor OCAP 1.0, into its cable set-top box line. Pace has also participated in a number of CableLabs' multi-vendor interoperability testing events over the past few months, including one this month, during which it says it brought its products to test with DVR-enabled EPG's, DOCSIS 2.0 Advanced DSG signalling, and various other OCAP applications that are under development. "Pace and Alticast have tracked the CableLabs specifications to a 'T' by implementing all the specifications and engineering change notices to date, which marks a major milestone," Pace Americas' VP of technology, Chris Dinallo, said in a prepared statement. "In addition, we uniquely designed our OCAP platform to function on different headend configurations, and in a manner that saves MSO's capital expense, time, and risk. We are on track to have a leading high-definition DVR OCAP-based platform ready for operators to make the transition to OCAP."
Appendix F

New Media Marketing

Enclosed is an article entitled DTT, cable and DSL offer future growth for middleware. The part of interest is the table presented, that illustrates the distribution of the middleware implementations by platform.

http://www.nds.com/pdfs/Middleware_NMM_300704.pdf
NMM guide to digital-television middleware in Europe

DTT, cable and DSL offer future growth for middleware

By Alexandra Wales

There is still a lot to play for in the market for interactive-television middleware in Europe with many digital-television platforms still to commit to a particular system.

Most cable and digital-terrestrial platforms have still to take the plunge into interactive television. And the emergence of television-over-DSL networks opens up another market for middleware.

Until now, satellite networks have accounted for the bulk of middleware deployments.

According to New Media Markets research, the installed base of set-top boxes that contain middleware among major operators totals 30 million. Of this, 67 per cent is accounted for by satellite, 19 per cent by cable, 13 per cent by digital terrestrial and one per cent by television-over-DSL platforms.

With Kabel Deutschland, Europe's biggest cable operator, still to choose its middleware platform, there is still much to play for in this part of the market and in the digital-terrestrial sector, which is still very much in its infancy.

It is in digital-terrestrial television that the multimedia home platform (MHP) open-standard middleware, already adopted in Italy, Germany and Finland, is likely to find its biggest initial market.

The emergence of television delivered over DSL opens another potential market for middleware providers.

John Tinsman, a director in the office of the chief technology officer at OpenTV, still the leading middleware player, says that the company is "actively pursuing" interests in the TV-over-DSL market.

Analyst David Mercer, at Strategy Analytics, believes that interactivity will come later rather than sooner in this sector: "I'm not sure that telcos really see interactive television as part of their portfolio at the moment, but it will be on the roadmap for the future.

Another key issue in the sector is MHP – the open-standard developed by the Digital Video Broadcasting Project (DVB). MHP has been a long time coming and it has still made little impact. Full-scale deployment has been "just around the corner" for several years now. At present there are fewer than 250,000 installed set-top boxes using MHP.

But the platform has been given a huge boost by the decision of the Italian government to subsidise set-top boxes that include MHP (NMM March 26, 2004). There are expected to be as many as 700,000 MHP-enabled boxes in Italy by the end of the year.

The main obstacle to wider deployment of MHP remains its relatively high cost. Over the last few years the trend has been for operators to deploy low-specification forms of middleware as a way to keep down costs.

Peter MacAvock, executive director of the DVB, says that prices for MHP-enabled set-top boxes will fall and that they "are a viable alternative to proprietary systems."

He adds: "Operators are anxious to ensure that they have platforms that can upgrade to MHP, but because of many issues, mainly cost, some are reluctant to invest."

The open standard is not necessarily a threat to the proprietary providers, as they can devise MHP-compliant versions of their own middleware and lever their existing customer relationships to provide migration to MHP. NDS, for instance, has contracts to provide MHP-compliant versions of MediaHighway to cable operators in South Korea, where MHP has been mandated.

Market share

OpenTV still dominates the worldwide market for digital-television middleware deployments, holding about 59 per cent of the market, according to the media consultancy Datasax. NDS follows with 27 per cent, then Liberate Technologies with seven per cent. Microsoft TV has 0.3 per cent, with the remaining seven per cent split between small middleware providers and in-house systems.

In Europe, according to New Media Markets research, among the leading operators included in our survey, OpenTV has a 41-per-cent market share, followed by NDS with 30 per cent and Liberate with 10 per cent.

OpenTV accepts that its market share is unlikely to increase. Tinsman says: "We don't want to be fighting for a bigger slice of the market. We want to see it go forward and each have our share."

The company, since May 2002 controlled by Liberty Media (NMM May 17, 2002), has also sought growth through diversification into related areas. In 2001, it bought interactive television company Static 2358 and last year it acquired Betting Corp, a developer and operator of betting and gaming technology.

Last September's €60 million acquisition of MediaHighway from Thomson Multimedia has made NDS, the leading player in the conditional-access market, a strong rival to OpenTV, both in Europe and elsewhere.

NDS's main task at the moment is the continuing integration of MediaHighway into its business. It will then be able to offer customers an integrated package combining MediaHighway with its VideoGuard conditional-access system.

The company hopes to secure new contracts, including those for MHP-compliant versions of MediaHighway, for digital-terrestrial television services across Europe.

Guillaume de Saint Marc, director of product marketing for middleware at NDS, says: "The main challenge for middleware companies in Europe is not MHP, because that is something that we are mastering quite well. The challenge is to manage products and offer not only advanced products but broadband and hybrid solutions combining free-to-air DTT, MHP, broadband services and PVRs."

Saint Marc believes that the middleware issue is critical for existing operators that are running platforms formed by the merger of former competitors. At present both Sky Italia and Spain's Digital Plus are using MediaHighway and OpenTV on different networks of set-top boxes. At some point, the operators will move to a single middleware system, although firm decisions have not yet been taken on which technology.
<table>
<thead>
<tr>
<th>Company/body</th>
<th>Middleware</th>
<th>Key customers</th>
<th>Country</th>
<th>Platform</th>
<th>Installed set-top boxes</th>
<th>Main shareholders</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenTV</td>
<td>OpenTV</td>
<td>BSkyB</td>
<td>UK</td>
<td>Satellite</td>
<td>7.27 million</td>
<td>Liberty Media (46%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Digital Plus (former Canal Satellite subs)</td>
<td>Spain</td>
<td>Satellite</td>
<td>approx 1.25 million</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TPS</td>
<td>France</td>
<td>Satellite</td>
<td>1.25 million</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sky Italia</td>
<td>Italy</td>
<td>Satellite</td>
<td>700,000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Viasat</td>
<td>Scandinavia</td>
<td>Satellite</td>
<td>600,000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Neos</td>
<td>France</td>
<td>Cable</td>
<td>442,991</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Boer</td>
<td>Sweden</td>
<td>DTT</td>
<td>250,000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Telia Stofa</td>
<td>Denmark</td>
<td>Cable</td>
<td>200,000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>France Télécom Cable</td>
<td>France</td>
<td>Cable</td>
<td>176,232</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ComHem</td>
<td>Sweden</td>
<td>Cable</td>
<td>140,000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cablecom</td>
<td>Switzerland</td>
<td>Cable</td>
<td>80,000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TeleDanmark</td>
<td>Denmark</td>
<td>Cable</td>
<td>68,000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chorus</td>
<td>Ireland</td>
<td>Cable</td>
<td>45,000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Casema</td>
<td>Netherlands</td>
<td>Cable</td>
<td>25,630</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Essent</td>
<td>Netherlands</td>
<td>Cable</td>
<td>25,000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UPC</td>
<td>France</td>
<td>Cable</td>
<td>22,200</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Primacom</td>
<td>Germany</td>
<td>Cable</td>
<td>10,986</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Est Vidéocommunication</td>
<td>France</td>
<td>Cable</td>
<td>7,723</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Multikabel</td>
<td>Netherlands</td>
<td>Cable</td>
<td>6,000</td>
<td></td>
</tr>
<tr>
<td>NDS</td>
<td>MediaHighway</td>
<td>Canalsatellite</td>
<td>France</td>
<td>Satellite</td>
<td>3.1 million</td>
<td>News Corporation (77.8%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sky Italia</td>
<td>Italy</td>
<td>Satellite</td>
<td>1,900,000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Digital Plus (former Canal Satellite subs)</td>
<td>Spain</td>
<td>Satellite</td>
<td>1.5 million</td>
<td>Financial institutions and private investors (22.2%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Canal Digital</td>
<td>Scandinavia</td>
<td>Satellite</td>
<td>950,000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Canal Digitaal</td>
<td>Netherlands</td>
<td>Cable</td>
<td>600,000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ITV Digital (boxes still in market)</td>
<td>UK</td>
<td>DTT</td>
<td>500,000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NC Numéricable</td>
<td>France</td>
<td>Cable</td>
<td>400,000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Be TV (formerly Canal Plus Benelux)</td>
<td>Belgium</td>
<td>Cable</td>
<td>160,000</td>
<td></td>
</tr>
<tr>
<td>Liberate Technologies</td>
<td>DTV Navigator</td>
<td>NTL</td>
<td>UK</td>
<td>Cable</td>
<td>1.43 million</td>
<td>Financial institutions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Telewest</td>
<td>UK</td>
<td>Cable</td>
<td>1.29 million</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ono</td>
<td>Spain</td>
<td>Cable</td>
<td>approx 75,000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UPC</td>
<td>Netherlands</td>
<td>Cable</td>
<td>53,500</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UPC</td>
<td>Norway</td>
<td>Cable</td>
<td>33,100</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UPC</td>
<td>Austria</td>
<td>Cable</td>
<td>26,600</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UPC</td>
<td>Sweden</td>
<td>Cable</td>
<td>26,300</td>
<td></td>
</tr>
<tr>
<td>Microsoft TV</td>
<td>Microsoft TV Advanced</td>
<td>TV Cabo</td>
<td>Portugal</td>
<td>Cable</td>
<td>approx 50,000 gradually being swapped out</td>
<td>Microsoft Corporation (100%)</td>
</tr>
<tr>
<td>Alcatel/iMagic TV</td>
<td>iMagicTV</td>
<td>KIT (Kingston Interactive Television)</td>
<td>UK</td>
<td>DSL</td>
<td>5,500</td>
<td>Alcatel (100%)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SesameTV</td>
<td>Monaco</td>
<td>DSL</td>
<td>300</td>
<td></td>
</tr>
<tr>
<td>Open-standard middleware</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DVB</td>
<td>MHP</td>
<td>Medienanstalt Berlin-Brandenburg (MAAB), T-Systems, television stations</td>
<td>Germany</td>
<td>Satellite</td>
<td>200,000</td>
<td>Industry body</td>
</tr>
<tr>
<td></td>
<td>MHP</td>
<td>Digita</td>
<td>Finland</td>
<td>DTT</td>
<td>20,000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MHP</td>
<td>Televisión de Catalunya</td>
<td>Spain</td>
<td>DTT</td>
<td>Trial 100 homes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MHP</td>
<td>VRT, Belgacom</td>
<td>Belgium</td>
<td>DTT</td>
<td>Trial</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MHP</td>
<td>TV2/Nord</td>
<td>Denmark</td>
<td>DTT</td>
<td>40,000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MHP</td>
<td>Mediaset, RAI, La7</td>
<td>Italy</td>
<td>DTT</td>
<td>700,000 by end of year</td>
<td></td>
</tr>
</tbody>
</table>

Notes: Some Freesat adapters do not include middleware, subscriber figures for end Q1/Q4, *Estimates from Ofcom Sources: New Media Markets and Companies

JULY 30, 2004
Appendix G

System Output

The two items contained with this appendix are; firstly 3 pages of log file output and secondly an error alert as would be sent to the engineer responsible for the given interactive application that has failed.
log.txt
An error has occurred with service; Basic Service. The error reference: 1 was returned.
Please check website http://localhost:8080/portal/index.html:8080
Appendix H

Schematics and Deployment diagram

The three drawings attached give a more visual representation of the architecture. These are, in order of appearance;

The original conceptual diagram illustrating all the functionality that the system will provide. A pictorial representation of the usecases presented as part of chapter 1.

The second is a deployment diagram for the actual infrastructure developed. showing the testing server, the engineering terminal and the remote service implemented on a remote PC.

The last of the drawings is very much a conceptual representation of the hardware involved showing the computers and the storage used to implement the system in its conceptual form.
<table>
<thead>
<tr>
<th>ID</th>
<th>Task Name</th>
<th>Start</th>
<th>Finish</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mid Term Report</td>
<td>Sun 19/11/06</td>
<td>Fri 08/12/06</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Exam Revision</td>
<td>Sat 06/12/06</td>
<td>Fri 19/01/07</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Reading</td>
<td>Thu 05/10/06</td>
<td>Fri 09/03/07</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Contact with paceMicro</td>
<td>Wed 18/10/06</td>
<td>Mon 27/11/06</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Contact with Kerim + Nick</td>
<td>Thu 12/10/06</td>
<td>Wed 29/11/06</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Define Project Specification</td>
<td>Tue 12/09/06</td>
<td>Fri 08/12/06</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Iteration 1 Section 1</td>
<td>Mon 22/01/07</td>
<td>Sun 04/02/07</td>
<td>Web Portal</td>
</tr>
<tr>
<td>8</td>
<td>Iteration 2 Section 1</td>
<td>Fri 02/02/07</td>
<td>Tue 13/02/07</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Final iteration Section 1</td>
<td>Mon 26/02/07</td>
<td>Tue 06/03/07</td>
<td>Extra Features</td>
</tr>
<tr>
<td>10</td>
<td>Testing</td>
<td>Mon 05/03/07</td>
<td>Tue 13/03/07</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Evaluation</td>
<td>Mon 05/03/07</td>
<td>Tue 13/03/07</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Iteration 1 Section 2</td>
<td>Mon 29/01/07</td>
<td>Fri 02/02/07</td>
<td>Database</td>
</tr>
<tr>
<td>13</td>
<td>Iteration 2 Section 2</td>
<td>Fri 02/02/07</td>
<td>Wed 07/02/07</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Final iteration Section 2</td>
<td>Mon 05/03/07</td>
<td>Wed 07/03/07</td>
<td>Extra Features</td>
</tr>
<tr>
<td>15</td>
<td>Testing</td>
<td>Tue 06/03/07</td>
<td>Sat 10/03/07</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Evaluation</td>
<td>Tue 06/03/07</td>
<td>Fri 09/03/07</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Iteration 1 Section 3</td>
<td>Mon 22/01/07</td>
<td>Tue 13/02/07</td>
<td>Simulator</td>
</tr>
<tr>
<td>18</td>
<td>Iteration 2 Section 3</td>
<td>Fri 09/02/07</td>
<td>Fri 23/02/07</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Final iteration Section 3</td>
<td>Mon 05/03/07</td>
<td>Fri 16/03/07</td>
<td>Extra Features</td>
</tr>
<tr>
<td>20</td>
<td>Testing</td>
<td>Wed 14/03/07</td>
<td>Mon 26/03/07</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Evaluation</td>
<td>Wed 14/03/07</td>
<td>Mon 26/03/07</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Iteration 1 Section 4</td>
<td>Mon 05/02/07</td>
<td>Tue 27/02/07</td>
<td>Test application</td>
</tr>
<tr>
<td>23</td>
<td>Iteration 2 Section 4</td>
<td>Fri 23/02/07</td>
<td>Fri 09/03/07</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Final iteration Section 4</td>
<td>Fri 16/03/07</td>
<td>Fri 30/03/07</td>
<td>Extra Features</td>
</tr>
<tr>
<td>25</td>
<td>Testing</td>
<td>Fri 30/03/07</td>
<td>Mon 09/04/07</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Evaluation</td>
<td>Fri 30/03/07</td>
<td>Mon 09/04/07</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Testing</td>
<td>Wed 04/04/07</td>
<td>Mon 16/04/07</td>
<td>Final</td>
</tr>
<tr>
<td>28</td>
<td>Evaluation</td>
<td>Wed 04/04/07</td>
<td>Mon 23/04/07</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Final Report</td>
<td>Mon 12/03/07</td>
<td>Thu 26/04/07</td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>Task Name</td>
<td>Start</td>
<td>Finish</td>
<td>Information</td>
</tr>
<tr>
<td>----</td>
<td>--------------------------------</td>
<td>----------------</td>
<td>----------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>1</td>
<td>Mid Term Report</td>
<td>Sun 19/11/06</td>
<td>Fri 08/12/06</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Exam Revision</td>
<td>Sat 06/12/06</td>
<td>Fri 19/01/07</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Reading</td>
<td>Thu 05/10/06</td>
<td>Fri 06/03/07</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Contact with paceMicro</td>
<td>Wed 18/10/06</td>
<td>Mon 27/11/06</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Contact with Kerim + Nick</td>
<td>Thu 12/10/06</td>
<td>Wed 29/11/06</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Define Project Specification</td>
<td>Tue 12/09/06</td>
<td>Fri 06/12/06</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Iteration 1 Section 1</td>
<td>Mon 22/01/07</td>
<td>Sun 04/02/07</td>
<td>Web Portal</td>
</tr>
<tr>
<td>8</td>
<td>Iteration 2 Section 1</td>
<td>Fri 02/02/07</td>
<td>Tue 13/02/07</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Final iteration Section 1</td>
<td>Mon 26/02/07</td>
<td>Tue 06/03/07</td>
<td>Extra Features</td>
</tr>
<tr>
<td>10</td>
<td>Testing</td>
<td>Mon 05/03/07</td>
<td>Tue 13/03/07</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Evaluation</td>
<td>Mon 05/03/07</td>
<td>Tue 13/03/07</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Iteration 1 Section 2</td>
<td>Mon 28/01/07</td>
<td>Fri 02/02/07</td>
<td>Database</td>
</tr>
<tr>
<td>13</td>
<td>Iteration 2 Section 2</td>
<td>Fri 02/02/07</td>
<td>Wed 07/02/07</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Final iteration Section 2</td>
<td>Mon 05/03/07</td>
<td>Wed 07/03/07</td>
<td>Extra Features</td>
</tr>
<tr>
<td>15</td>
<td>Testing</td>
<td>Tue 06/03/07</td>
<td>Sat 10/03/07</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Evaluation</td>
<td>Tue 06/03/07</td>
<td>Fri 06/03/07</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Iteration 1 Section 3</td>
<td>Mon 22/01/07</td>
<td>Tue 13/02/07</td>
<td>Simulator</td>
</tr>
<tr>
<td>18</td>
<td>Iteration 2 Section 3</td>
<td>Fri 09/02/07</td>
<td>Fri 23/02/07</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Final iteration Section 3</td>
<td>Mon 05/03/07</td>
<td>Fri 16/03/07</td>
<td>Extra Features</td>
</tr>
<tr>
<td>20</td>
<td>Testing</td>
<td>Wed 14/03/07</td>
<td>Mon 26/03/07</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Evaluation</td>
<td>Wed 14/03/07</td>
<td>Mon 26/03/07</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Iteration 1 Section 4</td>
<td>Mon 05/02/07</td>
<td>Tue 27/02/07</td>
<td>Test application</td>
</tr>
<tr>
<td>23</td>
<td>Iteration 2 Section 4</td>
<td>Fri 23/02/07</td>
<td>Fri 06/03/07</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Final iteration Section 4</td>
<td>Fri 16/03/07</td>
<td>Fri 30/03/07</td>
<td>Extra Features</td>
</tr>
<tr>
<td>25</td>
<td>Testing</td>
<td>Fri 30/03/07</td>
<td>Mon 09/04/07</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Evaluation</td>
<td>Fri 30/03/07</td>
<td>Mon 09/04/07</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Testing</td>
<td>Wed 04/04/07</td>
<td>Mon 16/04/07</td>
<td>Final</td>
</tr>
<tr>
<td>28</td>
<td>Evaluation</td>
<td>Wed 04/04/07</td>
<td>Mon 23/04/07</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Final Report</td>
<td>Mon 12/03/07</td>
<td>Thu 26/04/07</td>
<td></td>
</tr>
</tbody>
</table>
Appendix J

Dr Karim Djemame

A member of the research and development team specialising in distributed technologies.

Below are a number of comments recorded from a meeting held at 11am on Wednesday 20th of November 2006

General Notes:

Liked the functionality, some very nice ideas. Defiantly appropriate to the problem in hand.

In Kirim’s opinion keeping the level of abstraction conceptual will offer the best potential for developing a system that represents in some way the functionality specified.

The choice of languages and style of development is appropriate to the project. Overall good design with the potential to add some extra features and have a play with some distributed systems in the process.

Don’t go too far off track trying to make it too realistic but try to encapsulate some areas of realism.
Appendix K

Middleware

A collection of information on middleware implementations as used by set-top boxes throughout the interactive domain.

Multimedia Home Platform
www.mhp.org

NDS Middleware - MediaHighway
http://www.nds.com/middleware/middleware.html
The Multimedia Home Platform (MHP) defines a generic interface between interactive digital applications and the terminals on which those applications execute. It enables digital content providers to address all types of terminals ranging from low-end to high-end set top boxes, integrated digital TV sets and multimedia PCs. The MHP extends the existing, successful DVB open standards for broadcast and interactive services in all transmission networks.

**MHP architecture**

The architecture of the MHP is defined in terms of three layers: resources, system software and applications. Typical MHP resources are MPEG processing, I/O devices, CPU, memory and a graphics system. The system software uses the available resources in order to provide an abstract view of the platform to the applications. Implementations include an application manager (also known as a "navigator") to control the MHP and the applications running on it.

**MHP Profiles**

The MHP specification provides a consistent set of features and functions required for the Enhanced Broadcast, Interactive Broadcast and Internet Access profiles. The Enhanced Broadcast profile is intended for broadcast (one way) services, while the Interactive Broadcast profile supports additional interactive services. The Internet Access profile allows MHP to use the worldwide communication network provided by the Internet.

The MHP Standard is available on the DVB CD-ROM and from www.mhp.org or www.etsi.org
MHP Test Suite

The MHP Test Suite, an collection of over 10,000 tests, allows for verification of compliance to the MHP standard. Passing the Test Suite is a validation of the interoperability of an MHP implementation and guarantees the proper execution of MHP applications.

As a mark of quality, the right to use the MHP logo is only granted to those MHP implementation that pass the Test Suite.

How to get the MHP Test Suite?

The MHP Test Suite is available from the European Telecommunications Standards Institute (ETSI). Send ETSI the signed Test Application Licence and pay the administrative fee of 1,000€, and you will receive the latest version of the Test Suite. The source code and a test harness are available at no extra cost.

When your implementation has successfully passed the MHP Test Suite, you can choose to obtain a license of Intellectual Property Rights essential to the MHP specification from ETSI.

How to get the MHP logo?

The MHP logo is distributed and managed by the DVB Project Office. Only after successfully passing the MHP Test Suite and receiving the Certificate of Completion of Conformance Testing from ETSI are you able to apply for the right to use the MHP logo.

Send the signed MHP Mark License Agreement to the DVB Project Office and pay the royalty of 10,000€, and you will receive the MHP logo and design guidelines.

Contacts

MHP Test Suite Custodian
ETSI
Ms. Gina Ebenezersson
650, Route des Lucioles
F-06921 Sophia Antipolis
France
T +33 4 9294 4216
F +33 4 9294 4270
E ebenezersson@etsi.org

DVB Project Office
Mrs. Eva Melamed
17A Ancienne Route
CH-1218 Grand Saconnex (GE)
Switzerland.
T +41 22 717 2719
F +41 22 717 2727
E melamed@dvb.org

Questions?

Want to know more about MHP? The DVB Project Office is at your service. Ask us at
dvb@dvb.org

or find your answer on
www.mhp.org

You can also discuss MHP matters in an international forum on
forum.mhp.org

© Alticast

WP-02, April 2003
This screen shot is taken from:
http://www.nds.com/middleware/middleware.html

It is also where the attached paper detailing the NDS middleware development solution is available.