NTNU
Innovation and Creativity

Digital Home

An architecture for easy administration and updates of services

Andreas Wigmostad Bjerkhaug
Øystein Ellingbø

Master of Science in Computer Science
Submission date: May 2006
Supervisor: Svein Erik Bratsberg, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

This master thesis is based on the report "Digital Home - Home Theatre Solution" written during the autumn of 2005. The report suggests a centralized architecture for a digital home, which uses a "nerve center" as the control unit, focusing on the client side of a digital home. This master thesis will focus on a service provider's role.

The goal of this report is to find an architecture of how a service provider can offer services through the customers' nerve centers. The architecture must be of such a manner as to enable the service provider to easily administrate and update the services offered. These updates should be as transparent as possible to the customers. Ideally no involvement of the customer should be necessary.

The architecture shall also provide the customers with remote access to their nerve center, giving them access to a selection of remote services. These remote services could be such as remote control of the customer's nerve center or streaming of media from the nerve center.

The architecture must take security into consideration. This applies both to the communication between the service provider and the customers' nerve centers and to protection of copyrighted material.

It might be feasible to extend an already existing digital home system. In this case, it is important that the system is as forward-looking and complete as possible.

It is desirable that the students create a prototype of the chosen architecture.

Assignment given: 2006-01-20
Supervisor: Svein Erik Bratsberg, IDI

Digital Home

-

An architecture for easy
administration and updates of services

-

Master Thesis
TDT4900 - Database Technology
and Distributed Systems
Spring 2006

Author:
Andreas Wigmostad Bjerkhaug
<bjerkhau@idi.ntnu.no>

Author:
Øystein Ellingbø
<ellingbo@idi.ntnu.no>

Supervisor:
Øyvind Strømme
<oyvind.stromme@accenture.com>

Subject Teacher:
Svein Erik Bratsberg
<svein.erik.bratsberg@ idi.ntnu.no>

Preface

This master thesis is written at the Department of Computer and Information Science at the Norwegian University of Science and Technology (NTNU), for the Database Technology and Distributed Systems group. The project is a co-operative project between NTNU and Accenture.

Professor Svein Erik Bratsberg was the subject teacher, and Øyvind Strømme our supervisor at Accenture.

The initiative for this study came from Accenture during May 2005. Accenture has provided the necessary equipment and has been responsible for the main counselling throughout the course of the project.

We would like to thank Professor Svein Eirik Bratsberg and Øyvind Strømme for advice throughout the project and helpful input during the writing process of this report. We would also like to thank HP Norway for supplying a Media Center PC and Eirik Solheim for technical advice about Media Center.

Trondheim, 31st May 2006

Andreas Wigmostad Bjerkhaug

Øystein Ellingbø

Abstract

In the last years digital home solutions have made their entry and are are now becoming mainstream. From a service provider's point of view, this creates an interesting opportunity. Today, if a service provider wishes to change the services offered, the flash memory of the customers' Set-Top Boxes must be updated or the Set-Top Boxes must perhaps be replaced for larger updates. Replacing the Set-Top Box is both costly and time consuming. As digital home solutions become more normal, it is possible to use such a system to offer services. This way adding, removing or updating a service can be done in software only. Our proposed architecture does this, and with minimal involvment of the customers. The architecture offers a generic interface, providing for third party development.

As the world gets more and more digitalized, the expectation of everything to be availabe from everywere gets more common. Our architecture lets a service provider offer their customers remote access to, and control of, their digital home.

We have based our system on Microsoft Media Center Edition. This was chosen after first studying the concept of an ideal digital home and then researching which existing digital home solution would bring us closest to this ideal situation.

Contents

List of Figures

List of Tables

Part I

Introduction

Chapter 1

Introduction

During the last decade, the world has become ever more digitalized and one could claim we are living in the digital age. More and more systems in all areas of society are being substituted by new digital systems. Many people feel that the next natural step is to digitalize the entire home. There have been many attempts at this endeavour with more or less successful results. Thus there are a myriad of different digital home solutions available today, although most of these systems are focusing on digitalizing only a small part of the household. To get a complete digital home, the user has to buy, install and make us of many different systems. Another side of this story is that these systems are often complex, both to install and to use. This means that only a few number of people really have the ability to fully enjoy these systems, and even for those few it will most probably take a lot of time and/or money to get the system up and running.

Digital homes are not common today. As the regular PCs will get the possibility to control the digital home, and the appliances of the home support some sort of communication protocol, fully digitalized homes might be seen more frequently. Once digital homes become more common, different kinds of service providers will probably want to offer their services through the digital home systems of their customers. Naturally they would want these services to be fully integrated with the existing digital home solutions, as well as being easily maintainable without involving the end user in a large extent. Once the home has been fully digitalized and different kinds of services are available through the system, the users would also probably want to be able to access these services as well as a full control of his digital home from anywhere in the world.

The rest of this introductory chapter is structured as follows. First the problem description and the main goals of the project are presented. Next

the methodology used in our project is described. At the end, the structure of the rest of the report is given.

To get a brief overview of this project, it could be wise to read this introduction and the conclusions before reading the rest of the report.

1.1 Problem description and goals

The main purpose of this project is to find an architecture that allows services from a service provider to be accessible on a customer's digital home system. The architecture must support easy adding, removal and update of services as well as the possibility for remote access. A prototype for the chosen architecture will be developed. The full problem description can be found in Appendix A.

From the problem description we have derived three goals for this project. These goals are enumareted from G1 to G3 and are listed below:

G1 - Find a base for the architecture: The first goal is to find a base for the architecture. This could either be an existing digital home system or it could be developed from scratch.

G2 - Find a service architecture: The second goal is to find the service architecture. This architecture must support both services at the customer's digital home system as well as remote access to services.

G3 - Develop a prototype of the architecture: The third goal is to develop a prototype which demonstrates the system.

The first goal of the project is to find a base for the architecture. To do this, different existing digital home solutions will be explored and evaluated. The solutions will also be compared with developing such a system from scratch. When choosing a base for the architecture, is is important that the chosen system is extensible as well as containing as much of the desired functionality of the digital home as possible. To be able to know what requirements of a digital home are and to properly evaluate different solutions, the concept of the ideal digital home must be explored. This will be done through exploration of different scenarios and will result in a set of requirements of the ideal digital home.

The second goal of the project is to find the service architecture. This architecture must support an interface for accessing services from a service provider on the chosen base system. It must be easy for the service provider

to add, update and remove services. All of these updates must be as transparent as possible to the user. Furthermore the chosen architecture must also support remote access to services. These services include both the kind of services accessible from the customer's digital home system as well as services designed to control the customer's digital home system remotely.

The last goal of this project is to develop a prototype of the chosen architecture. The main purpose of the prototype is not to be a complete commercial realization of the system, but rather to demonstratete the most important aspects of the architecture through some simple example services.

1.2 Methodology

Most of this report is based on the knowledge gained from an area study of digital homes performed in the project "Digital Home - Home Theatre Solution" from 2005 and from a further area study performed in the spring of 2006.

To test if the possible architectures we came up with for our system actually would work and to better be able to evaluate them, rapid developed prototypes containing the basics of the different architectures were created. These prototypes only contained the core functionality of the architectures.

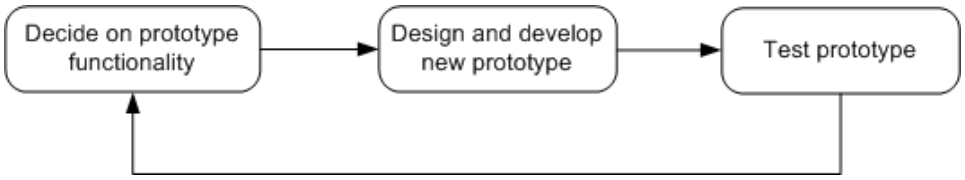

Figure 1.1: Prototype development model

We chose Microsoft Windows Media Center Edition as a basis for our system. This is a fairly new platform and developing for this platform was uncharted territory for us. Since we had little or no experience with the technologies needed to develop the prototype, we chose to develop the prototype using evolutionary prototyping. The essence of our prototype development method is shown in Figure 1.1. The finished prototype was developed through several iterations. Each iteration started by defining the functionality we wanted the prototype to contain after this iteration. Next the new functionality was added to the prototype design and the new version of the prototype was developed. At the end of each iteration the prototype was tested. If there were more functionality needed for the prototype, a new iteration began.

The design chapter of this report contains only the final prototype design.

To develop the prototype technologies new to us were needed. Thus some of the development period was spent studying and testing different technologies. The knowledge gained from these studies is presented in the report through brief descriptions of the different technologies.

1.3 Overview of the report

Chapter 2 and 3 address goal G1. Chapter 2 describes the ideal digital home through scenarios and requirements. Different existing digital home solutions are explored in Chapter 3 and then evaluated. The best existing solution is also compared with developing the system from scratch. At the end of Chapter 3 the chosen base solution, Microsoft Windows Media Center Edition, is explored.

Goal G2 is addressed by Chapter 4. This chapter presents an overview of our system and different possible architectures for that system. The different architectures are evaluated and a solution architecture is chosen. System requirements are presented in the end of Chapter 4.

Chapter 5 through 7 present the prototype developed in this project and address goal G3. Chapter 5 describes the design of the prototype. This includes both class descriptions and diagrams as well as sections discussing, among other tings, how server-client communication, updates and security are handled in the prototype. Chapter 6 presents different subjects regarding the development of the prototype. This includes technological choices made and descriptions of the most important technologies used in the prototype. Chapter 7 presents the result of the prototype development. This chapter includes screenshots and explanations of the finished prototype as well as a summary of the experiences we gained while developing for the chosen platform.

Chapter 8 presents some related work, our conclusions and future work.

Part II

Background information

Chapter 2

The Ideal Digital Home

We define the ideal digital home as a system where everything from heating and appliances to multimedia content is integrated as one system. This system will give the user extensive control of his home. In this chapter we look at the architecture and requirements of such a system. We will also look at what services a digital home should provide. We do this to gain a better understanding of important aspects to consider when creating our system.

First we discuss important aspects to consider when designing a digital home. These considerations form a possible architecture. Then two example scenarios are presented to shed more light on the architecture proposed. The chapter finishes by summarizing the requirements of the digital home as non-functional and functional requirements.

Keep in mind that this chapter might be a bit futuristic, although most of what we describe here is possible to create today. This chapter describes our vision of a digital home.

2.1 Important aspects

When creating a digital home there are many aspects to consider. In this section we start by exploring the ease of installation and use. We move on to make the architectural decision of a centralized or distributed solution. Then the functionality a digital home should provide is explored before we look into the aspect of remote access and security.

2.1.1 Ease of installation and use

A main focus is to create a system which is as easy as possible to install and use. The ideal digital home should not require the user to have deep technical skills. In addition, even if the user is technically skilled, the average person does not want to spend much time on installation and maintenance. The ease of installation and use are major selling points. No matter what great digital home you create, if it is not easy enough to install and use, it will most likely never be a commercial success.

A large step in the direction of creating a quick-to-install system is to use wireless network for the communication between all, or most of, the different devices. Having to run cables around your home to connect all the devices you want to include in your digital home would be a nightmare. Not using a wireless network will probably be enough to make the digital home a flop. Creating a wired digital home could be feasible when building a house from scratch. Wired network has advantages over wireless, especially when it comes to security. A good design choice would be to support both wired and wireless communication.

The GUI and interaction with the system should be constructed for the user to be in some distance to the screen. The user should be able to sit comfortably on his coach using a remote control to interact with the system, displayed on his TV screen. As mentioned, the ease of use is a major success criterion. The GUI should have easy menus and buttons with good affordance, providing quick access to the different functionality.

Today's "old-fashioned" solutions, such as light switches, VCRs and way of watching TV, require no or very little maintenance. People expect them to work, and normally the just do. A digital home will require maintenance. The average user does not want to spend time maintaining the system. Because of this, the system should give some service personnel the possibility to perform maintenance on the system without involving the user. This implies opening the system for persons outside the house and giving them administrator privileges. Security is a big issue here. The last thing you want do, is leaving backdoors open for hackers to exploit.

2.1.2 Centralized or distributed solution

The control unit of a digital home can either be a centralized or distributed system. Either way the user should have the experience of using one system, not many small systems controlling only parts of the digital home.

In a distributed solution you would have many control units collaborating.

A control unit would have to be an adequate computer to be able to handle all the different services the system should be able to provide. This would make a highly fault tolerant system. Even if one or more control units fail, most of the system would still be accessible. On the down side, the system would be very expensive. You would have to buy many control units, each in the price range of a personal computer. Because of the many control units, a distributed system would be time demanding to install.

A centralized solution would consist of a computer, a "nerve center", controlling the entire system. The user should be able to directly use the nerve center, connect using thin clients in other rooms or using other devices such as laptops or handheld devices. In contrast to the distributed solution, one nerve center and many thin clients are less expensive than many distributed control units. If the nerve center fails, the entire system will be unavailable. In a normal home this will probably not be critical. It is, however, possible to have an extra nerve center running as a backup. This backup nerve center would then be ready to take control if the main nerve center fails. This solution would increase the price of the system, but it would still be less expensive than the distributed solution. You could also reduce the impact of system failure by still having the old-fashioned controls (e.g. normal light switches).

Price and ease of installation are important aspects. Availability is also important, but for a normal home we do not consider it to be critical. Because of this, we will base our digital home on the centralized idea.

2.1.3 Functionality

The digital home should support a wide range of functionality. It should be able to control devices such as heating, appliances and alarm systems. Going on, it should also include telephony and handling of multimedia content such as television, music, radio, movies and your family photos. An important aspect when it comes to functionality is that if the system does not include a function, it should be easy to include this function as the need occurs.

Different people will have different needs. The system should be easily customizable to adapt to these needs. Smaller systems require less maintenance and are less likely to fail, thus only loading the system with the needed functionality would be feasible. In addition, some users might absolutely not want to have some of the functionality available. Not connecting the alarm system might be one of these things.

One aspect that could make or break the success of the digital home is that it gives the user a feeling of more, and new, functionality. It should absolutely

not feel like a limitation in comparison with today's functionality. As an example, a normal home today has multiple TVs. If the TV signal is received by cable or antenna, which channel you watch on one TV is independent of what other people in the house are watching. For the digital home to provide this independence, the nerve center must have multiple TV tuners.

2.1.4 Remote access and security

The digital home should support remote access both from within or immediately outside the house, and from remote locations. One consequence of a more and more digitalized world is the expectation of everything to be available from everywhere. Wireless network covers in and immediately outside the house. There are two main ways of gaining access from remote locations. One would be to have a direct openly accessible login functionality. The other is to connect through a service provider. Either way you open the system for the outside world, thus security will have to be a main area of focus when creating a digital home. For example it would not be acceptable if someone other than you gains access and turns off your alarm system remotely. If service personnel from the service provider are to be able to perform maintenance to the system, they need a login possibility. It would probably be a good idea to use this also for access from remote locations. Thus, connecting through a service provider seems feasible.

2.2 Possible digital home architecture

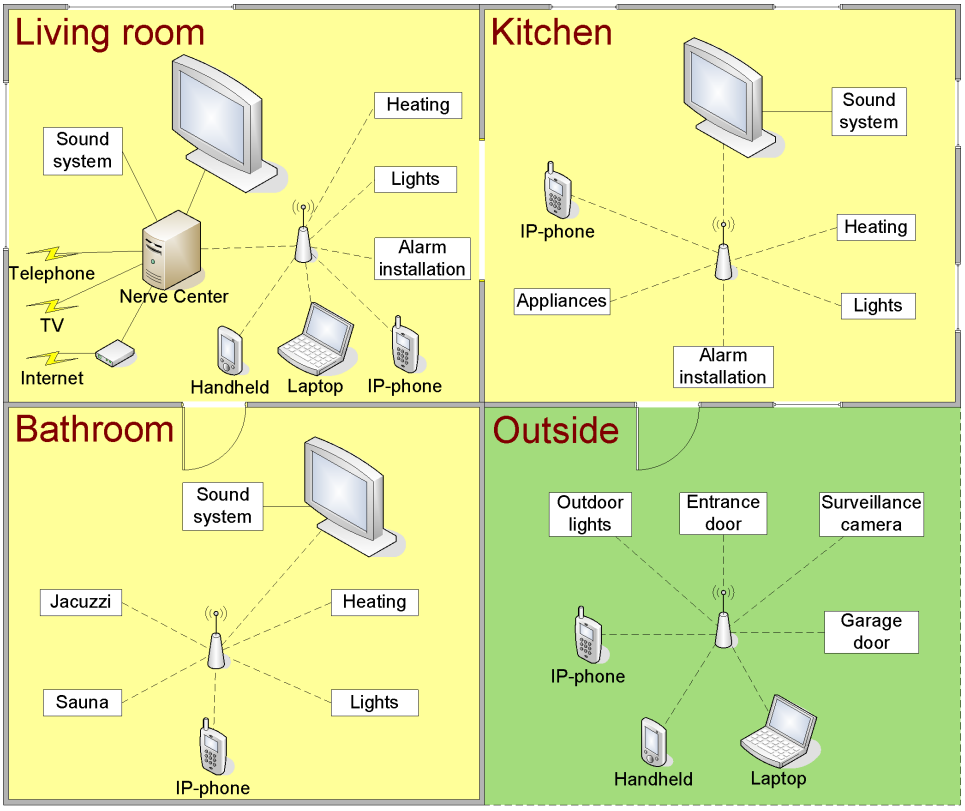

Figure 2.1: Sketch of a digital home

Based on the aspects considered in the last section, Figure 2.1 shows a sketch of a digital home. The main component of the system is the nerve center, currently located in the living room. The telephone line, TV signal and Internet connection are connected to the nerve center. The nerve center is also directly connected to the sound system in the living room and the TV. The nerve center controls the heating, lights, and alarm system in the entire house using a wireless network. It also controls other devices in the home, such as kitchen appliances, the Jacuzzi and sauna, the entrance and garage door and surveillance cameras.

In the other rooms of the house, you have a screen connected to the nerve center through the wireless network. In these rooms the sound system is connected to the screen.

The wireless network supports IP-phones around and immediately outside

the house. Using the wireless network it is possible to connect to the nerve center using a handheld device or a laptop.

It would also be possible to connect and control other devices than those in the sketch. For example you might want a gaming console in the kids' room. The connection of more devices should follow the same scheme as in the sketch, with wireless communication with the centralized nerve center.

2.3 Example scenarios

To give a better understanding of the ideal digital home we present two example scenarios. The first scenario gives an example of a typical day in your digital home. The next scenario shows how you can enjoy your digital home while travelling.

2.3.1 A day in your digital home

The day starts by waking up to your favourite radio channel. The bedroom lights were slowly dimmed up to 75% 10 minutes earlier. You have scheduled a standard time to be woken up each day of the week, so you did not have to adjust an alarm clock before going to bed yesterday. Your home automatically switches from night mode to morning mode based on your wake-up time. This might include turning up temperature and lights in specific rooms, turning off the alarm system and turning on the coffee maker. While eating breakfast you watch the latest news. You get a list of the news available, and can choose the headlights of interest.

When you leave for work, you switch the house to away mode, turning down lights and heating and activating the alarm system. While at work, your service provider's service personnel perform a weekly maintenance on your system (e.g. security patches and program updates). Back home again, dinner is ready. Using the plasma screen on the kitchen, you select "food ready". This sends a message to all other screens in your house, telling everybody dinner is served.

Yesterday you did not have the opportunity to watch your favourite TV show, but since you have selected this show to be one of your favourites, it was automatically recorded. While watching yesterday's show, your neighbour calls asking to borrow some tools. The show is really exciting and you do not want to pause it. Using your remote you activate the security camera motion notification system. When your neighbour approaches the garage, a message pops up in the corner of your TV, telling you there is movement

outside the garage. You open the garage door by a couple of clicks on the remote. If you like, you can display the surveillance camera's feed in a small window in the corner of your TV.

Later on you plan a romantic Jacuzzi with your wife. Using your PDA you ready the Jacuzzi and set the bathroom to romantic mode. The system notifies you when the Jacuzzi is ready. At the end of the day, before going to bed you switch the house to night mode, turning off the lights, down the temperature and activating the alarm system.

Figure 2.2 shows a use case diagram of this scenario.

Figure 2.2: Use Case: A day in your digital home

2.3.2 Enjoying your digital home while travelling

You are going on a business trip. When entering the airport, you remember you forgot to set your house to away mode. While waiting for the boarding to start, you use your mobile phone to connect to your service provider and set your home to away mode.

After a long day of meeting, you feel like relaxing in the hotel room. The movie you bought last week, using your service provider's multimedia-on-demand service, would be perfect. Using your laptop, you connect to your service provider and can browse all the media content on your nerve center. It is also possible to select a new movie from the service provider, which will be downloaded to your nerve center and streamed to your laptop. You connect the laptop to the hotel TV, lie down on the bed and watch the desired movie.

Figure 2.3 shows a use case diagram of this scenario.

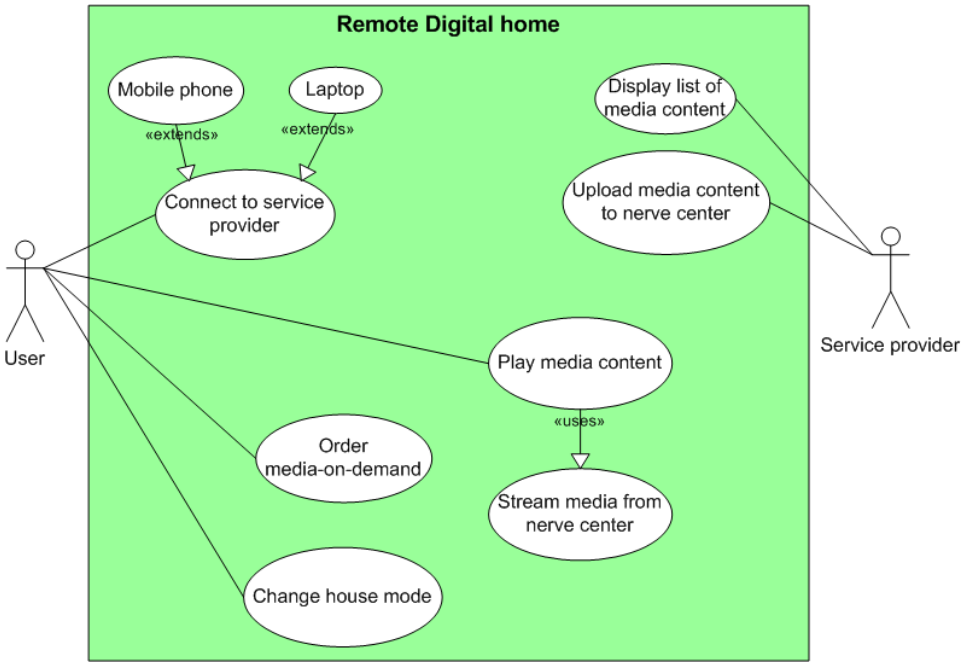

Figure 2.3: Use Case: Enjoying your digital home while travelling

2.4 Non-functional requirements of the ideal system

This section gives a description of the non-functional requirements of the digital home. These requirements are based upon our view of the ideal digital home. A non-functional requirement describes the constraints set by the environment and the external factors that influence the system. All Ideal Non-Functional Requirements are given a unique number, starting with "INFR".

INFR1 The system should have an easy to understand installation guide.

INFR2 Non-technicians should be able to install the system.

INFR3 The system should have good help functionality.

INFR4 The GUI should be easy, requiring few clicks to navigate the menus.

INFR5 The GUI should be build for 10-feet-interaction.

INFR6 The response time of the system should be low enough for the user not to get the feeling of waiting.

INFR7 The system should have a high up-time, but it is not critical if the system fails on occasions.

2.5 Functional requirements of the ideal system

This section presents the functional requirements of the ideal system. As with the non-functional requirements, these requirements are based upon our view of the ideal digital home. It starts out by presenting the general architecture and continues to introduce different intended areas of use. Finally requirements concerning the user and the service provider are presented. All Ideal Functional Requirements are given a unique number, starting with "IFR".

2.5.1 System architecture

There are many aspects concerning the system architecture. Here we summarize the main requirements of the proposed architecture.

IFR1 The main component of the system should be a computer acting as a nerve center, controlling the digital home.

IFR2 The nerve center should be connected to a screen for direct access.

IFR3 The nerve center must be connected to the Internet.

IFR4 The nerve center should have a high speed Internet connection. Otherwise many of the possible services will not be available.

IFR5 It should exist fixed devices in multiple rooms, connected to the nerve center for remote access.

IFR6 It should be possible to communicate with the nerve center using a laptop or handheld device.

IFR7 It should be possible to use wireless network for the communication between the nerve center and all connected devices.

2.5.2 Areas of use

The digital home covers many different applications and devices. The most central of those are described in the requirements below. The most important one being that the system must be made extensible.

IFR8 The user should be able to watch and record TV using the system. An Electronig Program Guide (EPG) should be available.

IFR9 The user should be able to watch DVD video and other video files stored on the nerve center or other computers in the household.

IFR10 The user should be able to use multimedia-on-demand services.

IFR11 The user should be able to listen to music, radio and other audio sources.

IFR12 The user should be able to browse the World Wide Web.

IFR13 The user should be able to check e-mail.

IFR14 The user should be able to use instant messaging applications.

IFR15 The user should be able to interact with his/her nerve center from wireless devices such as a laptop, PDA or mobile phone.

IFR16 It should be possible to communicate between the different screens connected to the system through an instant messaging application.

IFR17 The user should be able to receive incoming and create outgoing telephone calls. This includes phonevision if the system is equipped with one or more webcams.

IFR18 The user should be able to control and monitor the status of the security system.

IFR19 Other devices such as light switches, kitchen appliances etc should also be controllable, if supported by the appliances.

IFR20 The system must be made extensible so that future devices not listed here should be able to interact with the system.

2.5.3 The customer and a service provider

Requirements regarding the customer's use of the system and communication with a service provider are looked into in [11]. These requirements, modified to be more general, are repeated here.

IFR21 The system should be built to satisfy primarily two types of active actors: The customer and a service provider's service personnel.

IFR22 The customer must be able to interact with a service provider from the nerve center.

IFR23 Service personnel should be able to update software on nerve center devices remotely.

IFR24 Customers that already have a nerve center should not be required to buy a new nerve center customized by a service provider. The service provider should therefore offer a "package" that upgrades the customer's current nerve center to integrate it with the service provider's system.

Chapter 3

Exploration of existing digital home solutions

The concept of a digital home is not a new concept. There are many different digital home solutions commercially available today. The most sophisticated solutions available are often so complicated that very few people are able to use them. Thus the purpose of this chapter is to try to give an overview of the digital home market and find a suitable solution for our system to be built upon.

Chapter 2 presented our vision of a digital home. Even though most parts of this system are possible to create or already exist in some form, there does not exist any digital home solutions encompassing all of the ideal digital home. Most systems available today focus on multimedia functionality. Because of this, we look into using one of these systems as a base for our project.

The chapter starts out by presenting the digital home solutions of today. These are divided into software based and operating system based solutions, and a representative number of these solutions are presented. In the following section these solutions are evaluated and the best commercial off-the-shelf (COTS) solution is picked out. This solution is then compared with developing the system from scratch and a final solution is selected. Finally the selected solution is looked into in more detail, especially how the different requirements from the previous chapter can be realized.

3.1 Digital home solutions of today

There are a variety of different digital home solutions available today. This section tries to give an overview of the current situation, divided into software based and operating system based solutions as proposed by [11]. The software based solutions are stand-alone applications giving the user the possibility to control his or her digital home from a personal computer. The operating system based solutions are entire operating systems specially designed for controlling the digital home.

In addition we also briefly present so called embedded system solutions, i.e. hardware based systems that fulfil different parts of the requirements of the ideal system. A digital home based upon different embedded systems can be realized as of today, but the solution will be a mixture of many different systems (e.g one system for multimedia functionality and one for controlling household lighting). That way the user will have to deal with different systems for different functionality which is not how we want the system to be. Despite all this, some embedded systems are briefly presented just to give a brief overview of what is available today.

3.1.1 Software based solutions

One of the requirements of the digital home system is that it must be easy to use and to install for the average user. Most people have a personal computer in their home today, and most of these computers are running some version of the Microsoft Windows operating system. For this reason we limit our presentation here to applications designed for the Microsoft Windows operating system.

There are many applications available for controlling multimedia for home computers. There are for example multiple applications which allow the user to play media files, use TV tuners and listen to radio on his or her computer. However, not all of these are suitable to be used in a digital home setting. If the applications are to be used in such a setting, it is required that the user can make use of the system from a TV screen and preferably also through a remote control. According to [11] the following applications fall into this category: Snapstream Beyond software, Showshifter, SageTV and ORB.

Snapstream is a collection of different digital home software. Beyond TV is a Personal Video Recording (PVR) software that lets the user watch live TV and schedule recordings through an Electroninc Program Guide (EPG). A screenshot of the EPG accessible in Beyond TV is shown in Figure 3.1. Beyond Media lets the user watch DVD videos and both browse and watch

other media files. These programs can be combined with a remote control called Firefly and Beyond TV Link which makes it possible to connect to the system from other computers in the home network and make use of the features mentioned above [27].

Figure 3.1: Beyond TV EPG

Showshifter seems to have many of the same features as the software from Snapstream except from the TV Link part. Another main difference is that all the tools are included in one software package here, whereas when using Snapstream the user needs to buy several different software packages to get all the desired features of the system [26]. A screenshot from Showshifter is displayed in Figure 3.2.

Figure 3.2: Showshifter EPG

SageTV seems to be a solution with features very similar to those of Show-

shifter [25]. Although there are differences between them, the main functionality remains the same. The GUI of SageTV can be seen in Figure 3.3.

Figure 3.3: SageTV main menu

Another application usable in digital home settings is ORB. It allows the user to access media from his or her home computer remotely. This includes scheduling recordings, watching live TV, playing media files and so on. The user communicates with ORB through a web browser, and ORB is also possible to access on a PDA [22].

The applications examined so far are all commercially available solutions. There also exist open source solutions. Here especially one program stands out, Media Portal. This system seems to have much of the same functionality as the programs examined earlier in this section. In addition Media Portal also supports the use of RSS news feeds, reading and sending of emails, checking weather forecast and so on. Since Media Portal is an open source project, the system is also customizable for a programmer [17]. A screenshot from Media Portal is displayed in Figure 3.4.

Before starting to look at the operating system based solutions, a couple of other applications will be examined. These applications include features of home automation in addition to control of media. The programs examined in this part are Proximis Girder and a collection of applications from Meedio.

Proximis Girder is a system designed to manage the entire home of the user. This includes for example controlling programs for media playback, the lighting and the household alarm system. Girder lets the user make advanced scripts to control his or her home. This feature also makes Girder highly customizable and many pre-made plug-ins are ready for download. Girder makes use of a command based GUI which might be hard to read on a TV screen. In addition this GUI is rather complex, so the average user will probably have difficulties using this system [23]. The main window of

Figure 3.4: My music on Media Portal

Girder may be seen in Figure 3.5.

There are different applications available from Meedio that together make a digital home solution. Meedio has recently been purchased by Yahoo, and the programs presented here are the ones offered by Meedio in the time just prior to the purchase. It is likely that Yahoo will expand the functionality of the Meedio software, especially when it comes to online services. Meedio Essentials is the digital media center of the Meedio package. This program lets the user playback all kind of media files using a GUI suited for TV screens. In combination with Meedio TV, the user may also watch live TV and schedule recordings using an EPG through this system. Meedio HouseBot is a home automation program. The user may access the program on PCs, TV screens and PDAs, and control for example lights, appliances and A/V equipment [18]. Figure 3.6 shows the main menu of Meedio Essentials.

The applications presented in this chapter are only some of the many applications for multimedia in digital homes and home automation available on the market today. Our intention was not to present every possible solution, but to give an overview of the types of applications that are available. Although this section has focused on applications for Microsoft Windows, similar applications may also be found for other platforms.

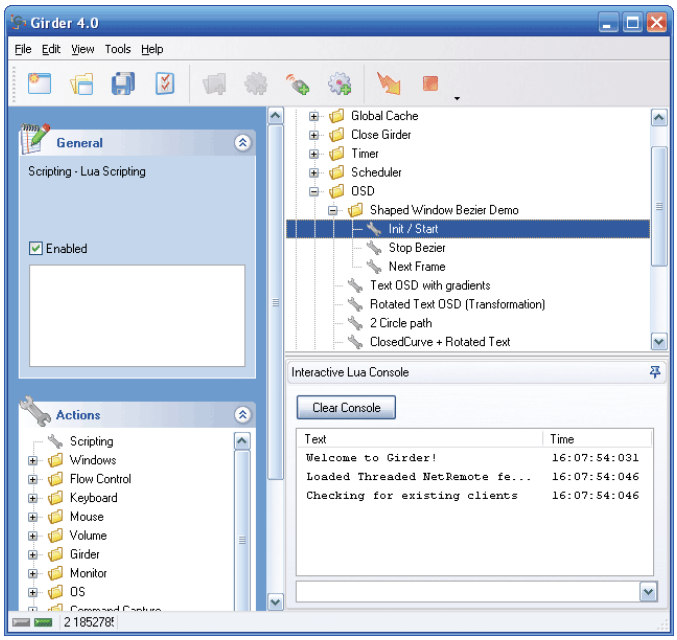

Figure 3.5: Girder main window

Figure 3.6: Meedio Essentials main window

3.1.2 Operating system based solutions

The only operating system based solution for digital homes today which is widely commercially available is Microsoft Windows XP Media Center Edition (MCE). This section will give a brief overview of the functionality of the newest version, i.e. the 2005 version, of this operating system.

Microsoft Windows XP Media Center Edition 2005 is really a normal version of Windows XP with a Media Center application called eshell running on top of it. The main menu of the Media Center is displayed in Figure 3.7. Like many of the solutions presented in the last section, MCE lets the user watch live TV, record TV shows through an EPG and play different kinds of media files (audio, video, pictures). The user also have the option to copy multimedia to a CD or a DVD. [38].

Figure 3.7: Main menu of MCE

In addition to all of these features, users of MCE will get access to different online spotlights. These are different online services, for example multimedia-on-demand, which are provided by a third party. One example of this is the newly released spotlight from the Norwegian Broadcast Corporation (NRK). Through this spotlight the user gains access to a huge archive of videos and twelve different radio stations, and all of this is free of charge [30]. A screenshot of this service is shown in Figure 3.8.

Microsoft has released a SDK which allows programmers to develop applications, called add-ins, for MCE using .NET and HTML. This makes it fairly easy to add new features to MCE. There are already many different add-ins available, for example add-ins which let users check weather forecast and use

Figure 3.8: NRK online spotlight for MCE

Microsoft Outlook through MCE.

MCE is as the name implies focused on controlling the multimedia of the household, it is a home theatre solution. There are no features for home automation embedded into MCE. However, there exists an application for MCE developed by Embedded Automation called mControl which adds home automation features to MCE [3]. This program lets the user control surveillance cameras and many other devices such as lamps, heaters and so on.

The home theatre PC (HTPC) itself is connected to either a monitor or a TV screen. In addition it is possible to connect up to five different TV screens to the HTPC through so called extenders. The MCE can be used independently at each of these 5 screens in addition to the HTPC itself. This means that up to six users may access content and play media files at the same time. However MCE only supports two hardware TV tuners as of today, so only two different channels may be in use at any given time. Figure 3.9 shows how this could look like with a HTPC in the living room and an TV screen connected to an extender in the bedroom.

MCE is only available in an OEM version, i.e. it has to be sold with some hardware. The idea from Microsoft is that this operating system should only be sold together with special Media Center PCs. The reason for this is that MCE have specific hardware requirements of the system. There are multiple Media Center PCs available on the marked today. These range from the ones looking like a traditional PC to the ones designed to match your home hi-fi system.

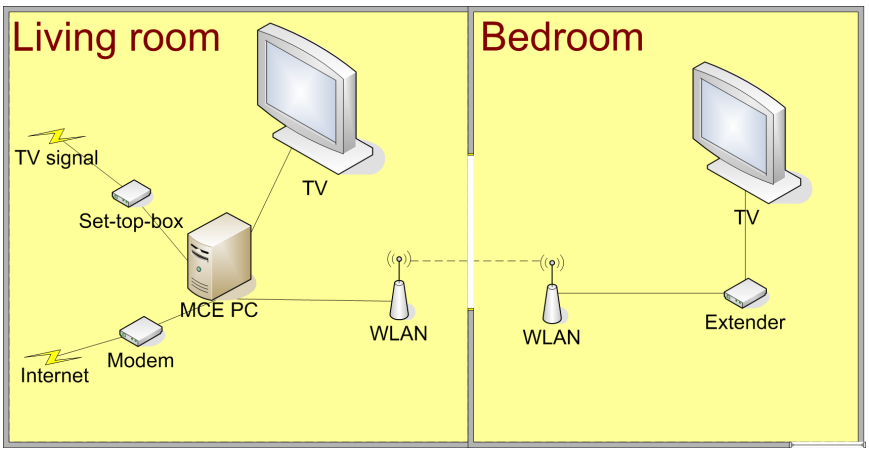

Figure 3.9: System with one MCE and one extender

3.1.3 Embedded system solutions

In the previous sections we have discussed different software based and operating system based solutions. All of these require the user to have a PC available for use. There also exist special embedded systems which by themselves feature some of the requirements of the ideal digital home, and together might realize the entire digital home. This section does not present many systems in detail, but rather tries to give a brief overview of what is available. We start out by presenting systems for multimedia purposes. At the end we present some home automation solutions.

At the multimedia front there exist a myriad of different embedded system solutions. These range from the simplest DVD players and recorders to the more advanced products such as hard drive based personal video recorders (PVR). These PVR machines let the user make digital copies of TV shows and store these on a hard drive. The recorded shows can on some PVRs be shared with other computers in the home network through the FTP protocol. Some PVRs also feature an electronig program guide (EPG) which can help the user program recordings more easily.

There also exist many embedded system solutions for the home automation part of the digital home. This includes different control centers for controlling lighting, heating, home security and so on. For example, Leviton [12] offers different lighting control systems that work with the existing AC wiring of houses. Other companies offer similar products to control other appliances of the household.

This was just a very brief overview of what functionality one can get from

using some embedded system solutions. These systems provide good solutions to the special features they offer. The problem is that if using these systems for controlling the digital home, the user will have to access different systems for controlling different parts of the house. As described in Chapter 2, we want the user to be able to control everything from one system which is easy to use. Thus we will not discuss any embedded system solutions any further.

3.2 Evaluation and choice of solution

The chapter so far has presented many different applications for digital homes. This section will select which solution that will be used in this project. The evalution criteria and the evaluation of the different solutions are first prestented. Finally one solution will be selected. This selection will be based upon the evaluation and external reviews. Before leaving this subject alltogether, the selected solution will be compared with development from scratch.

The evaluations presented in this chapter are purely based on information gathered from the different manufacturers. The systems have not been tested first-hand. In addition a user review of some of these systems is briefly presented in Section 3.2.3. The reason why the systems have been evaluated in this manner is mainly because we found the information available on the different systems more than adequate to judge how suitable the different systems are for our project. Also the acquisition of all of these systems would involve a substantial investment cost, and it was deemed unnecessary to invest in the systems to give a proper evaluation.

The purpose of the evaluation is to find a base which can be extended with our system. In addition we want the chosen system to posess as much of the functionality of the ideal digital home as possible. For this reason, the systems from Snapstream, SageTV and Showshifter are not evaluated since they are not extensible. ORB is not evaluated either, since this system is not extensible and only represents a limited version of the remote access functionality we desire for our system.

3.2.1 Evaluation criteria

Prior to performing an evaluation, the criteria on which to base the evaluation need to be carefully defined. This section will present a total of 9 criteria, each of which is given a number from C1 to C9. During the evalua-

tion, each solution will get a score from 1 through 5 on each of these criteria. An exception to this is criterion C9. This criterion will only get a score between 1 and 3. The reason for this is that this criterion is considered slightly less important than the other ones.

C1 - TV/radio functionality It must be possible to use the system to watch TV. This includes both live TV and recorded TV. In addition the system must have the possibility to schedule recordings, preferably through an EPG. It must also be possible to use system to play radio.

C2 - Play media It must be possible to play and/or show media files on the system. These media files must include audio, video and pictures. The system should also support multimedia-on-demand services through the Internet. Support for DVD video and CD audio should also be present.

C3 - Home automation It must be possible to use the system for home automation. This includes features like controlling lighting, heating, home security and so on.

C4 - Remote control It must be possible to control the system through a remote control.

C5 - Connect to the system through remote devices It must be possible to gain access to all or some of the system through remote devices using the Local Area Network.

C6 - Use the system at multiple screens It must be possible make use of the system at multiple screens simultaneously. This should be done without connecting a PC to each screen (i.e. we want a thin client solution).

C7 - Ease of use The system must be easy to use, although it is often hard to define what makes a system easy to use. It is not easy to rate the user-friendliness of a system, especially without testing the system first-hand. Since this evaluation is purely based on information from the manufacturers of the different systems, this criterion is mostly based on whether or not the GUIs of the systems are designed to be used on a TV screen and from a distance. In addition, if a system has a very complicated GUI, as observed from screenshots, this will result in a lower score.

C8 - Extensibility Chapter 4 through 7 describes the service provider functionality which we have designed and developed in this project. To be able to integrate this into the base system, the system must be extensible. This criterion describes to what extent the system is extensible.

C9 - Cost This criterion is meant to measure the cost of the different systems. The difference in price between the different systems is not that large, so this criterion will give a score of 3 if the system is free and a score of 1 if the system is a commercial system.

3.2.2 Evaluation of the solutions

This section presents evaluations of the different solutions based upon the evaluation criteria defined in the previous section.

Evaluation of Media Portal

Media Portal is evaluated below, and the evaluation is summarized in Table 3.1.

C1 - TV/radio functionality - Score: 5 Media Portal supports both watching of live TV, scheduling recordings through an EPG and the use of radio.

C2 - Play media - Score: 4 Media Portal supports all kinds of media files specified except multimedia-on-demand.

C3 - Home automation - Score: 1 Media Portal does not support home automation features.

C4 - Remote control - Score: 5 Several remote controls are supported by Media Portal.

C5 - Connect to the system through remote devices - Score: 1 Media Portal does not support this feature.

C6 - Use the system at multiple screens - Score: 1 The multi-screen feature is not supported by Media Portal.

C7 - Ease of use - Score: 5 Media Portal is designed to be watched on a TV screen and from a distance. Furthermore the GUI seems to be easy to use.

C8 - Extensibility - Score: 5 Media Portal is extensible via plug-ins. Furthermore Media Portal is an open source application, so the program itself should be possible to customize.

C9 - Cost - Score: 3 Media Portal is an open source project and is thus free.

Criterion	**Score**
C1 - TV/radio functionality	5
C2 - Play media	4
C3 - Home automation	1
C4 - Remote control	5
C5 - Connect through PDA/mobile phone	1
C6 - Use the system at multiple screens	1
C7 - Ease of use	5
C8 - Extensibility	5
C9 - Cost	3
Total	**30**

Table 3.1: Evaluation of Media Portal

Evaluation of Proximis Girder

Proximis Girder is evaluated below, and the evaluation is summarized in Table 3.2.

C1 - TV/radio functionality - Score: 2 These features are only available through 3rd party applications which are controllable through Girder.

C2 - Play media - Score: 2 These features are only available through 3rd party applications which are controllable through Girder.

C3 - Home automation - Score: 5 Girder supports most home automation features available today.

C4 - Remote control - Score: 5 Several remote controls are supported by Girder.

C5 - Connect to the system through remote devices - Score: 1 This feature is not supported by Girder.

C6 - Use the system at multiple screens - Score: 1 The multi-screen feature is not supported by Girder.

C7 - Ease of use - Score: 1 Girder is not designed to be used from a distance and on a TV screen. Furthermore the Girder GUI is complicated and not suitable for everyone.

C8 - Extensibility - Score: 5 Girder can be extended through a variety of plug-ins.

C9 - Cost - Score: 1 Girder is a commercial system and thus involves an investment cost.

Criterion	Score
C1 - TV/radio functionality	2
C2 - Play media	2
C3 - Home automation	5
C4 - Remote control	5
C5 - Connect through PDA/mobile phone	1
C6 - Use the system at multiple screens	1
C7 - Ease of use	1
C8 - Extensibility	5
C9 - Cost	1
Total	**23**

Table 3.2: Evaluation of Proximis Girder

Evaluation of Meedio software

The software from Meedio is evaluated below, and the evaluation is summarized in Table 3.3.

C1 - TV/radio functionality - Score: 4 Both watching live TV and scheduling recordings through an EPG is available from Meedio Essentials and Meedio TV. However the radio functionality is not supported.

C2 - Play media - Score: 4 Meedio Essentials supports the playback of all media content specified in this test. Multimedia-on-demand is not supported.

C3 - Home automation - Score: 5 Home automation is supported through the Meedio HouseBot application.

C4 - Remote control - Score: 5 The Meedio software supports the use of remote controls.

C5 - Connect to the system through remote devices - Score: 3 It is possible to connect to Meedio HouseBot through a PDA.

C6 - Use the system at multiple screens - Score: 1 The multi-screen feature is not supported by Meedio.

C7 - Ease of use - Score: 5 The GUI of the software from Meedio is designed for use on TV screens and to be used from a distance. The GUI also seems easy to use.

C8 - Extensibility - Score: 5 The Meedio package can be further extended through different add-ins.

C9 - Cost - Score: 1 The software from Meedio are commercial systems and thus involve an investment cost.

Criterion	**Score**
C1 - TV/radio functionality	4
C2 - Play media	4
C3 - Home automation	5
C4 - Remote control	5
C5 - Connect through PDA/mobile phone	3
C6 - Use the system at multiple screens	1
C7 - Ease of use	5
C8 - Extensibility	5
C9 - Cost	1
Total	**33**

Table 3.3: Evaluation of Meedio software

Evaluation of Microsoft Windows Media Center Edition

Microsoft Windows Media Center Edition is evaluated below, and the evaluation is summarized in Table 3.4.

C1 - TV/radio functionality - Score: 5 MCE supports both watching of live TV and recorded TV as well as the ability to listen to radio. Schedule TV recordings is also available, and it is possible to do this through an EPG.

C2 - Play media - Score: 5 MCE supports all audio and video files the PC have codecs for as well as letting the user watch pictures. MCE also supports various multimedia-on-demand services through online spotlights.

C3 - Home automation - Score: 3 MCE does not support home automation features by itself, but there exists for example an add-in called mControl which adds home different home automation features to the MCE.

C4 - Remote control - Score: 5 MCE uses a special remote control supplied with all Media Center PCs.

C5 - Connect to the system through remote devices - Score: 1 MCE does not support this feature as of today.

C6 - Use the system at multiple screens - Score: 5 MCE supports this feature by the use of up to 5 extenders on one system.

C7 - Ease of use - Score: 5 MCE is designed to be used on a TV screen and from a distance. Furthermore the GUI is intuitive and easy to use.

C8 - Extensibility - Score: 5 New features can be added to MCE either through already existing add-ins or developed through the MCE SDK.

C9 - Cost - Score: 1 MCE is a commercial system and thus involves investment costs.

Criterion	**Score**
C1 - TV/radio functionality	5
C2 - Play media	5
C3 - Home automation	3
C4 - Remote control	5
C5 - Connect through PDA/mobile phone	1
C6 - Use the system at multiple screens	5
C7 - Ease of use	5
C8 - Extensibility	5
C9 - Cost	1
Total	**35**

Table 3.4: Evaluation of Microsoft Windows Media Center Edition

3.2.3 Choice of best COTS solution

A multitude of different solutions have now been briefly presented and then evaluated according to the given evaluation criteria. This section will summarize the evaluation and present one of the solutions as most suited for our system. This section will also present an external review of some of these systems. Finally one solution will be selected based upon both the evaluation and the external review.

The evaluation is summarized in Table 3.5. As can be seen from this table, Microsoft Windows Media Center got the highest score with the software from Meedio and Media Portal following close behind.

Solution	**Score**
Microsoft Windows Media Center Edition	35
Meedio software	33
Media Portal	30
Proximis Girder	23

Table 3.5: Evaluation summary

The main difference between the top two candidates is that MCE has got the wanted multi-screen feature whereas the software from Meedio supports home automation in itself and partly the connection through PDA. In our opinion it is much harder to extend the Meedio software with multi-screen features than extend MCE with PDA/mobile phone compatibility and to add a home automation add-in. Therefore MCE is here chosen as the winner of the evaluation.

Eirik Solheim presents in his blog a review [29] of some of the systems presented in this chapter. The following systems are a part of this review:

1. Showshifter
2. Meedio + Beyond TV
3. Microsoft Windows Media Center Edition.

The review also presents the combination of MyHTPC and Snapstream PVS, but since these are the predecessors of Meedio and Beyond TV respectively, these will not be mentioned further here.

Solheim started out by using Showshifter. His main reason to change to the combination of Meedio and Beyond TV was that the creators of Showshifter were slow on updates and he really wanted hardware encoding support and EPG (which is now included in Showshifter). He also claims Showshifter still seems to be slow on updates. The author of the blog seemed quite pleased with the combination of Meedio and Beyond TV. He also briefly tested Beyond Media, but found this both a bit limited and not very user friendly.

Finally Solheim tested Microsoft Windows Media Center Edition. Although this system also have some disadvantages such as it requires more powerful hardware, is more expensive and lacks a streaming server and a web GUI, he finds this system most satisfactory. The installation process was described to be very easy and the system was running stable on his PC. The system

was described to be a bit limited compared to Meedio, but since there exists a myriad of available add-ins to MCE this was not considered a big problem. To quote the author of the review: "MCE is by far the easiest and most complete HTPC front end I have tried".

Based upon both the evaluation performed in this report and the review by Solheim, we select Microsoft Windows Media Center as the most suitable COTS solution to base this project upon.

3.2.4 Comparison with development from scratch

A comparison between developing a digital home system based upon MCE and developing from scratch is found in [11]. This section is based upon that comparison.

We will first examine the advantages and disadvantages of basing our implementation upon MCE. The advantages of developing on MCE rather than from scratch are as follows:

- A lot of the wanted functionality is already implemented.
- New functionality can be added from already existing add-ins.
- New functionality can easily be developed using the MCE SDK.
- Known brand, good support from Microsoft?
- Hardware components especially designed for this system are commercially available (PCs, extenders etc).
- Supported multimedia-on-demand services exist for MCE.

The disadvantages can be summarized in the following way:

- The architecture of the system may become an obstacle.
- The source code is not publicly available, thus it may contains some features which can not be removed/customized or bottlenecks when considering some functionality.
- One have to install an entire operating system instead of just a software package to use the system.
- Limited options when choosing development language and platform.

- One have to spend time to get to know the SDK.
- Hardware limitations, e.g. MCE only supports two tuners and five extenders as of today.

If we want to develop everything from scratch, we will have other possibilities and other problems. The advantages of developing the solution from scratch are listed below:

- Can choose developing platform and language freely and thus it is possible to keep the solution platform independent.
- No need to install an entire operating system when installing the final product.
- No unwanted functionality.
- Full control over the software.
- Full access to the source code, the system can be optimized to our needs.

There are of course disadvantages as well when developing from scratch. Some of these are presented below:

- Have to develop everything ourselves, may take more time.
- No framework to lean on.
- Higher risk, if it fails the loss will be greater.
- No specially designed hardware exists for this solution.
- Hard or impossible to integrate with many existing multimedia-on-demand services.

3.2.5 Final choice of solution

So far in this chapter many COTS solutions have been evaluated and Microsoft Windows Media Center Edition has been selected as the most suitable base for this project. In the previous section, advantages and disadvantages between developing on MCE and from scratch was explored. Now we will finally select the solution we are going to use and briefly explain the reason behind that choice.

In this project we develop an architecture to let a customer access services from a service provider both locally as well through remote access. The development of existing digital home features is not our main focus, so it is preferable to extend an existing digital home system rather than developing everything from scratch. For this reason we consider developing everything from scratch to be a bit waste of time and effort, and we will develop our system on MCE.

3.3 Exploration of the chosen solution

Microsoft Windows Media Center Edition has now been chosen as the foundation of our solution. This section presents how some of the functionality of the ideal digital home is realized in MCE. We also look at add-ins that can offer some of the functionality that MCE by itself is lacking. The areas we look into are multi-user environment, home theater functionality, home automation functionality and remote connection. Contrary to the previous section, this section is based both on literature and on actually testing the system first-hand.

At the end of this section we try to give an overview on how well an MCE based solution fulfil the requirements for the ideal digital home presented in Chapter 2. We do this to see how far away we now are from realizing the ideal digital home.

3.3.1 Multi-user environment

As mentioned when describing MCE earlier, it is possible for up to six users to use the system at any given time. One person can use the MCE PC directly while five other persons are using the MCE at other TV screens in the household. This is possible due to extenders which are communicating with the MCE and can be connected to a TV screen.

The extenders, used to connect more TV screens to the MCE PC, are small hardware boxes which can connect to the HTPC either through wired or wireless LAN. A special version of the remote desktop protocol (RDP) is used by the extenders to communicate with the MCE PC. There are currently two different extenders available, one from Linksys [13] and one from HP [7] (which can be seen in Figure 3.10). These extenders are currently only available in the US. It is also possible to use an Xbox as an extender, but this requires the user to buy a special software package from Microsoft (which is also only available in the US).

Figure 3.10: HP x5400 Media Center Extender

The newly released Xbox 360 has built-in extender functionality, and it is possible to buy a special MCE remote control to this console [43]. Xbox 360 is the first available extender in Europe, and in time many homes will feature one of these. The result of this is of course that the users does not have to invest in seperate extender boxes if they already are in posession of one or more Xbox 360 consoles. Furthermore the Xbox 360 is not much more expensive than a dedicated extender box, and it features a lot more functionality than just the extender capability described here. A picture of an Xbox 360 console can be seen in Figure 3.11.

Figure 3.11: Xbox 360

The hardware requirements of your MCE PC increases with the number of extenders you want to connect to it. Table 3.6 presents the CPU speed and memory size of the MCE PC for different numbers of extenders as recommended by Microsoft [16].

Evidently it requires a rather powerful computer to support from three to five extenders. In addition the bandwidth of the home network also limits the number of possible extenders. This is especially true when using WLAN,

Number of extenders	CPU Speed	Memory
1	2.8 GHz	256 MB
2	2.8 GHz	512 MB
3-5	3.4 GHz	1 GB

Table 3.6: MCE PC hardware requirements for different numbers of extenders.

where the highest bandwidth of today is 108 Mb/s. As computer hardware and network evolves, future versions of MCE will hopefully support more than five extenders.

Since extenders can connect to the MCE PC through a wireless LAN, some security issues need to be confronted. To prevent anyone from connecting to your MCE PC through your wireless LAN, there are special setup procedures for extenders. The procedures are different for different kinds of extenders. However, in all of them you have to specify at the MCE that the extender is to be allowed to connect to the MCE PC. For example when connecting an Xbox 360 unit, you have to enter a 8-digit code on your MCE which you get from the setup process at the extender [16].

3.3.2 Multimedia functionality

This section looks at the multimedia functionality of MCE. Since this is what MCE really is designed for, it has typically good support for most of the requirements of this category. Here we will present how MCE handles such media as TV, video, audio and pictures.

MCE has support for both watching live TV, scheduling recordings of TV shows and watching recorded TV through the menu option "My TV". As of today, MCE supports only two tuners. That means that only two different channels may be used at any given time. Each of these channels may be watched at any number of extenders. In addition MCE can record from either or both of these channels. If a user tries to switch channel when all the tuners are in use, the user will first see a list of the different tuners and what they are showing. From this list, the user can stop all activity on one of the tuners and use this for his or her purposes. Although MCE normally does not support more than two tuners, a user on the forums of "The Green Button" [33] claims that it is possible to change Windows registry values to enable more tuners. As mentioned at the start of this section, it is also possible to schedule recordings of TV shows. This scheduling of recordings can either be done by selecting channel and date/time or by selecting the

show you want to record in the electronic program guide. It is also possible to use the EPG to choose witch show to watch.

In addition to watching TV, it is possible to watch other kinds of video using MCE through the "My Video" menu. Here it is possible to watch video files stored at the MCE PC or any other computer in the home network. MCE supports the playback of all kinds of video files as long as the proper codec is installed. It is also possible to access different video-on-demand services through the "Online Spotlights" menu. Of the current available online spotlights in Norway, there are video-on-demand services like the ones offered by NRK, TV2 and SF Norge. Finally it is also possible to playback DVD videos on MCE by entering the menu "Play DVDs". However, this feature is only available at the MCE itself and not through any of the extenders.

MCE lets the user playback audio files. This is done from the "My Music" menu. Here you can access all audio files stored both on the MCE PC and at other computers in the home network. It is possible to both playback audio CDs and copy them onto the MCE PC. When entering the "My Music" menu, all albums accessible through the MCE is browsable through icons displaying the album cover. To give the user more flexibility and to help the user navigate faster through possible thousands of audio files, MCE supports both a search function as well as the creation of playlists. There exist also audio-on-demand online spotlights. These include both services which the user can buy regular songs as well as a karaoke service. Figure 3.12 shows MCE playing one of the sample audio tracks which is included with MCE.

Figure 3.12: Screenshot of My Music.

Pictures on the MCE PC and other computers in the home network may

be browsed through the "My Pictures" menu. Here pictures can be sorted into different categories, and slideshows of some or all the categories can be displayed. It is possible to playback audio while watching slideshows as well as printing pictures directly from the MCE software.

In addition to all of this it is possible to copy any of the multimedia content to a CD or DVD as well as listening to radio through the "My Radio" menu.

3.3.3 Home automation functionality

MCE does not support any home automation functionality by itself, but there exist add-ins which makes this possible. This section will first discusses two of the most common home automation protocols called X10 and INSTEON. At the end of this section, an MCE add-in called mControl will be presented. This add-in utilizes both the X10 and the INSTEON protocol to provide home automation features to an MCE PC.

The X10 protocol [35] was developed in 1975 by Pico Electronics and is perhaps still the most widely used protocol for controlling household devices and appliances remotely. The already existing AC wiring of the household is used to send digital data between X10 units and X10 controllers. The protocol supports up to 256 different units in one system, each identified by a house code from A to P and a unit code from 1 to 16. The house codes represent different zones in one household whereas the unit codes represent different units within one zone. Commands such as "Turn on", "Turn off", "All lights on", "All units off" are supported by the protocol. In addition one need to select which zone and possibly which unit one wants to give a command to. As an example, to turn off all lights in zone D in the house one would give the following commands: "Select house code D", "All lights off".

INSTEON [8] is a home automation network developed by Smarthome. INSTEON is first and foremost a wireless home-control networking product. It also supports sending information through the AC wiring of the household, but wireless communication is preferred. The signals transmitted in the X10 protocol are sensitive to changes in the voltage and current in the wirings. This is avoided by using wireless communications instead. The reason INSTEON also supports communication through the AC wiring is twofold. Firstly it works as a backup network in case of wireless interference. Secondly it allows INSTEON to be backwards compatible with X10. Units in an INSTEON network are not given codes like in an X10 network. Instead each INSTEON device is given a unique ID to identify it.

Embedded Automation has developed a home automation application called

mControl [3] which is designed for MCE. mControl supports devices and controllers for both INSTEON and the X10 protocol. In addition it also supports two different surveillance cameras: D-Link 2100+ and 5300W. mControl lets the user divide his or her home into different zones and add different units to the different zones. When selecting a zone, the user will get a list of all devices in the zone with status information and the possibility to control each unit. In Figure 3.13 a screenshot of mControl is presented.

Figure 3.13: Screenshot of mControl.

3.3.4 Remote access

When thinking about remote access in a digital home situation we actually think at two different kinds of remote access. The first one lets the user connect to the system from other devices on the same LAN as the MCE PC. The second kind of remote access is connections to the system through the Internet. Although MCE does not support any of these (except the connection through hardware extenders on the LAN), we will briefly discuss how such features should be implemented.

The first type of remote access has perhaps the easiest solution. Devices such as laptops (or other computers in the household), PDAs and mobile phones should all be able to connect to the system using the same protocol as the hardware based extenders. However, Microsoft has not released the details about this protocol or any software based extenders as of today. Although rumors has it that the software based extender is going to be released with some versions of Windows Vista, this information is highly unofficial and

the rumors have not been confirmed by Microsoft yet. This type of remote access will not be further investigated in this project.

The second kind of remote access can be achieved by using different kinds of third party application. For instance, one can use TvOnTime [34] for scheduling recordings and ORB [22] for accessing media remotely. However, this requires the user to access different systems for different functionality. The user should be able to have access to all functionality from one single system. Providing a remote access inteface through the web page of a service provider is a part of the main focus of this project and will be more thoroughly explored later in this report.

3.3.5 Summary of requirement fulfilment by MCE

In this section we will look at the main requirements of the ideal system described in Chapter 2 and discuss how MCE fulfils these requirements. The main reason for this is to see how far away we are from realizing the digital home by using MCE as a basis for development. At the end of this section, we try to explain which areas which will be the main focus for our master theses.

Non-functional requirements MCE is both easy to use and to install and also provide good help functionality. The GUI is also designed for 10-feet-interaction. The system have relatively fast response times and the uptime is the as for the normal version of Microsoft Windows XP. New functionality can be added easily to MCE both through already existing add-ins and self-made add-ins developed using the MCE SDK. Thus MCE is extensible.

System architecture The MCE digital home system consists of a nerve center, the MCE PC, which could be connected to a screen for direct access. Furthermore other screens in other rooms can be connected to the nerve center through both wired and wireless LAN. However, as of today only 5 extra screens are supported. It is not possible to communicate with the system through laptops or handheld devices as of today.

Areas of use Most areas of use are covered earlier in this section. The multimedia functionality is very well supported by MCE. Home automation features are not part of MCE itself, but there exist add-ins which provide home automation functionality. This, however, requires the devices and appliances of the household to support the X10 or the INSTEON protocol. Remote control of the nerve center from other

devices than extenders is currently not possible. It is as of today not possible to use MCE to browse the web, check email or make telephone calls, but there exist add-ins that support some of these functions and it is possible to develop new add-ins that support the rest. Regarding instant messaging applications, currently only MSN Messenger is supported directly.

The customer and a service provider This functionality does not exist.

To summarize this chapter, we see that MCE provides much of the functionality required by the ideal digital home and more functionality can be added through either already existing or self-made add-ins. The area which MCE covers the least is the customer and a service provider. Actually MCE does not cover this area at all. This functionality is what we want to provide. If successfull, it will fill a large gap between today's situation and the ideal digital home.

The focus of the rest of this report is to find an architecture which provides a service provider with easy administration and updates of services, accessible both through their customers' Media Centers and by remote access. We will also develop a prototype demonstrating this architecture.

Part III

Finding and developing an architecture

Chapter 4

Architecture and requirements

In Chapter 2 we explored the ideal digital home, and in Chapter 3 we used this to find which existing system would bring us closest to the ideal digital home and at the same time be a good basis for our system.

The main goal of this master thesis is to find an architecture and create a system that lets a service provider easily update their services and change the range of services offered. All this should require as little involvement of the customers as possible. The architecture should also enable the service provider to offer their customers remote access to, and control of, their Windows Media Center PC (MCE PC). This is what our system will add to Windows Media Center.

In this chapter we give a high level description of the desired system, and use this to find the best possible system architecture. At the end, this is summarized as system requirements.

4.1 High level system description

This section presents a high level system description, starting with the paramount requirements of the system. Next a textual system description is given which also explains the paramount requirements in more detail. At the end of this section, a scenario of use of the system is presented along with use case diagrams describing the scenario.

4.1.1 Paramount requirements

The main functionality of the system can be summarized into a number of paramount requirements. Some of these requirements are based upon the requirements of the ideal digital home stated in Section 2.5.3. They are all presented in one list of paramount requirements, as shown below. Each Paramount Requirement is given a unique number starting with "PR".

PR1 The system should be built to satisfy primarily two types of active actors: The customer and a service provider's service personnel.

PR2 The customer must be able to interact with a service provider from his/her Media Center.

PR3 Service personnel must be able to update software on the customers' MCE PCs remotely.

PR4 Updates must be as transparent as possible to the customer.

PR5 The software should be easily installable on any given MCE PC. The service provider must therefore offer a package that upgrades the customers' current MCE PC to integrate it with the service provider's system.

PR6 The customer must be able to interact with his/her Media Center from other devices through the service provider.

4.1.2 Textual system description

The system should offer the customers of a service provider access to different services through the customers' MCE PCs. The services accessible for the customers can for instance include multimedia-on-demand, computer games, download of different kinds of software and so on. The service provider can offer different packages to its customers and let the customers subscribe to the services they prefer.

A system such as this will consist of two different kinds of services. The first type is the ones mentioned above. These are called user-invoked services. In addition, the system also needs different background services. These services are for instance responsible for file transfer and updates.

It should be possible for the customer to change which services he or she subscribes to. In addition services are naturally in need of updates as time goes by. This means that the service provider should be able to add, remove

and update services in some manner. These operations may also require the customer to gain access to new updates to the client version of the software to be able to continue to access the services provided by the service provider. All of this should be made as transparent to the customer as possible. Thus the system should be designed in such a way that it needs a minimum of updates of the client software. If the client software needs to be updated, these updates should be downloaded and installed automatically and interfere as little as possible with the customer's use of the system.

The service provider could offer its customers MCE PCs which are customized with the service provider software. However, some customers might already be in possession of a MCE PC and do not wish to invest in a new one. Therefore it is important that the service provider software can be easily obtained and installed on the customer's MCE PC.

The part of the system described thus far involves both user-invoked and background services, but all access to services happens through the customer's Media Center. Another way to let the customer gain access to services is through a web interface accessible on different remote devices. Such remote devices include standard PCs, laptops, PDAs and mobile phones. The idea is to let the customer log into his or her account at a web page supplied by the service provider. At this web page the customer may edit his or her account information and subscription data as well as accessing different services provided by the service provider. These services include communication with the customer's Media Center, and can for instance be scheduling and deleting recordings, ordering downloads, gaining access to media on the MCE PC and so on.

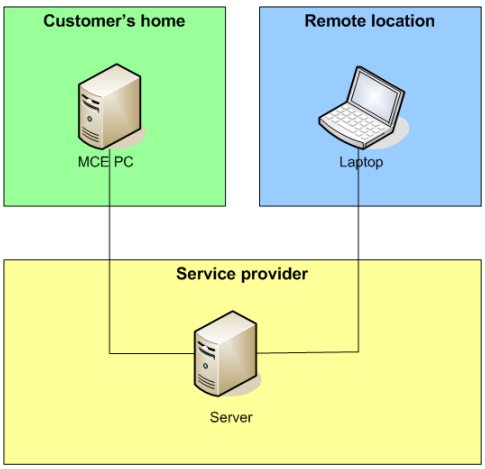

Figure 4.1: System architecture overview

A figure showing the main idea of the system architecture is shown in Figure 4.1.

The next section describes the use of this system through a scenario.

4.1.3 Scenario of use

You are looking forward to a relaxing night at home watching your favourite TV show through your Media Center. Although the show does not air anymore at any of your local TV stations, you have the option to watch the show through the video-on-demand service you subscribe to. After having watched the show for a couple of minutes, you discover that your bandwidth does not allow you to watch the show with the desired quality.

You enter your subscription data and add a service which allows you to download different TV shows to your MCE PC. The change in subscription is confirmed by entering your customer id and password. While the show is downloading, you try out one of the new games you got access to through your subscription. However fun the game may be, you want to explore the list of third party add-ins you can download to check if any new ones have been added to the list since your last visit. You find an add-in for displaying weather conditions at your Media Center, and select the add-in for download and install. A moment later a pop-up-box informs you that the add-in is installed. While testing the new add-in, another pop-up-box informs you that your TV show has finished downloading and is ready to be watched.

The next day at work, you discover that you have to work later than first expected. You enter the service provider's web page through your web browser and log in using your customer id and password. From the status screen on top of the page, you see that your MCE PC is connected to the service provider's server. You schedule a recording of the TV show you were planning to watch. While you are logged in to the service provider's web page, you also select the next episode of the show you watched last night to be downloaded to your MCE PC.

Finally on your way home from work, you want something to do to pass the time on the long buss trip. You log in to the service provider's web page using your mobile phone and selects music from your MCE PC you want to be streamed to your mobile phone while travelling.

Figure 4.2 shows the use case diagram for this scenario. The diagram shows the customer on the left side with the different actions which are available to him or her in this scenario. The service provider is also shown with the actions they can perform on the customer's MCE PC. The diagram shows

that the service provider can collect anonymous data and log statistics. This is only done if allowed by the customer. The use of remote access services is shown in Figure 4.3. This use case diagram shows the different remote services available to the customer in this scenario. It also shows how different parts of the remote access interface are connected with the service provider's server and the MCE PC.

Figure 4.2: Use Case: The customer's use of the local system

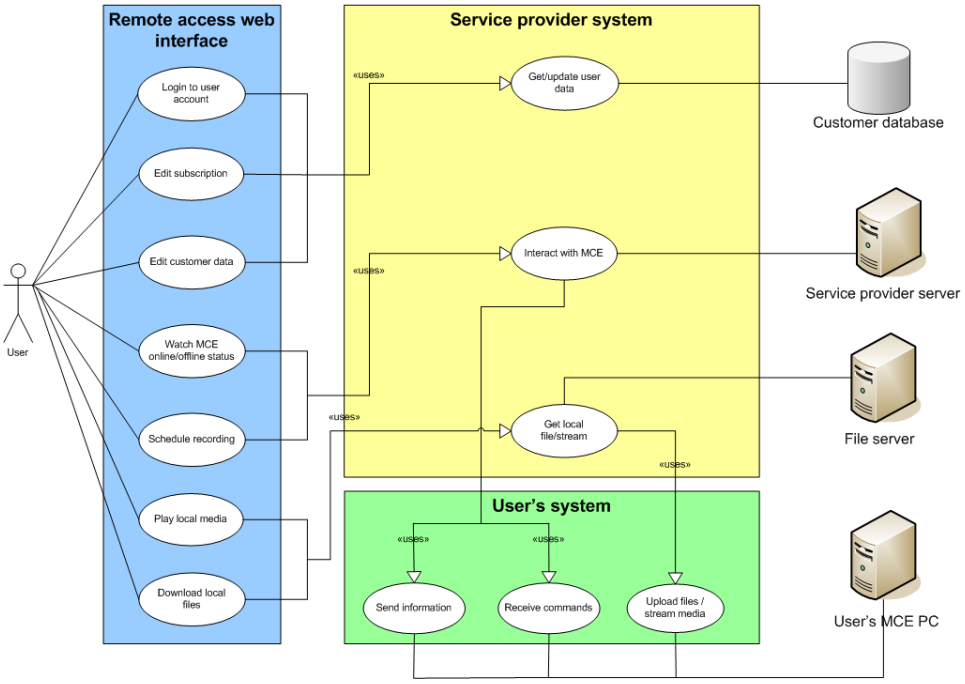

Figure 4.3: Use Case: The customer's use of remote access

4.2 Core system architecture

The core of our system is how a customer accesses services both from his Media Center and remotely. This section explores two possible architectures for this core functionality. The first is to create all user-invoked services as local Media Center add-ins. These add-ins communicates with the service provider by sending XML documents back and forth. The second alternative is to run the user-invoked services as web pages on the service provider's servers. This solution will also require locally running add-ins, but much less of them. Both solutions require some background running services. It is two ways of achieving this, which we will look into at the end of this section. Before we look into how to run background services, the communication module designed for the core system is explored and the differences between the communication modules for the two architectures are discussed.

4.2.1 Add-in solution

The add-in solution is your standard client-server architecture. All services a service provider provides will have to be installed as add-ins on each cus-

tomer's MCE PC. These services communicate with the server through a secure connection and update their content this way. To update the actual services, an update has to be downloaded and installed. This update process is handled automatically without involvement form the customer. If a customer should get access to new services, these will be downloaded and installed on the MCE PC. In the same way, if a customer no longer shall have access to a service, it will have to be uninstalled. At this point it is important that the customer in no way can block the removal of a service.

The Media Center will have a service provider menu in its main menu. This menu contains a list of the different services the customer has access to. Different customers can have access to different services. Since the menu is locally stored, to add or remove services the menu has to be updated.

In addition to the services accessible from the service provider menu, there will have to be a collection of background running add-ins installed on the MCE PC. These add-ins, or background services, are needed to handle such as installation, updates and removal of services. Remote access services will also require background services. The background services will also need updates, but probably less frequently than the services accessible from the service provider menu.

The server side of this architecture authenticates the customers and handles requests from the services running on the customers' Media Centers. To handle these requests, a collection of matching service handlers will have to be running on the server.

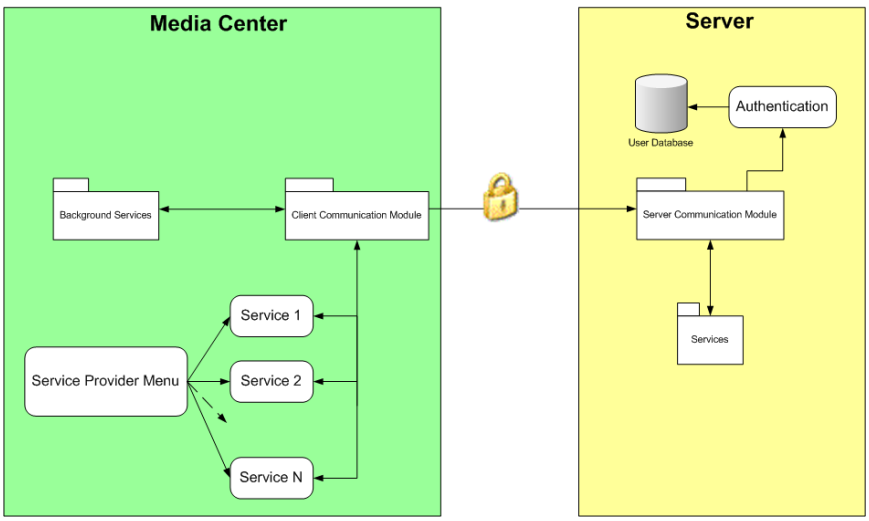

Figure 4.4: Local add-in architecture

Figure 4.4 shows the architecture of the add-in solution. The left hand side of the figure shows the client, a MCE PC. The Media Center has a service provider menu, which lists the different services the customer has available. The different services use the communication module to communicate with the server and update their content. The communication module will be described in Section 4.2.3. The Media Center part also consists of a collection of background services, such as an update manager. The right hand side of the figure shows the server. A communication module handles the communication with the MCE PC. Using a user database, the server authenticates the customer. Depending on the request a customer makes, different services on the server are invoked.

To support remote access, some background services are required on the MCE PC. The service provider handles the remote access, thus the server must be extended. A model of this is shown in Figure 4.5. How the server handles services accessed form a customer's Media Center remains unchanged. To handle remote access, the server will have to have a remote access menu. The customer is authenticated using the user database. Once authenticated, the customer gains access to a selection of remote services. Which services are available can vary between different customers. The remote services can either just need information form the server, or communicate with the MCE PC. Communication with the customer's MCE PC goes through the communication module.

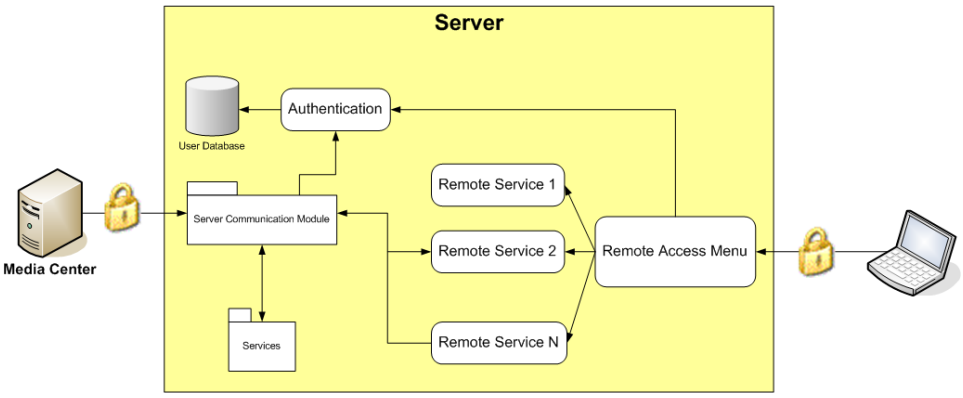

Figure 4.5: Local add-in architecture, remote access

4.2.2 Web solution

The web solution is a modification of the add-in solution. The main difference is that the services accessed from the Media Center are moved to the server. Actually, the whole service provider menu is located at the server. There will still be a service provider menu in the main menu of the Media Center, but this is just a link to the menu at the server. When a customer accesses the service provider menu, his identity is checked against the user database and a secure connection is acquired. The service menu is then built according to which services the customer should have access to. The server contains all services offered by the service provider, and each customer will have access to a selection of these services. To change which services a customer has access to, no installation or uninstallation is required at the customer's MCE PC. It is merely a question of changing the customer's profile in the user database. To update a service, the server must be updated. After updating the service this one place, all customers accessing the service will be using the updated version. As with the add-in solution, the web solution will require some background services running on the MCE PC. To modify a background service, updates must be downloaded and installed.

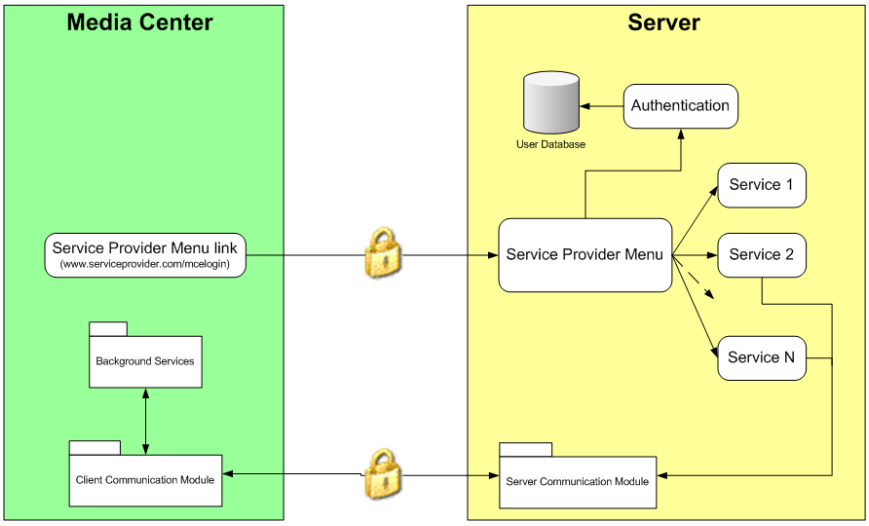

Figure 4.6: Web solution architecture

Figure 4.6 is a model of the web solution. As we see on the left hand side, the service provider menu is replaced with a link to the menu on the server. The Media Center still has a communication module and background services, but these are less comprehensive than in the add-in solution. At the server side, we now see the service provider menu. Some of the services offered will need to send data to or receive data from the Media Center. Because of this,

we still need the communication module. Since the services are located at the server, there is no need for the service handlers found at the server side of the add-in solution.

Remote access will require background services on the customer's MCE PC. Figure 4.7 shows a model of how the server will have to be extended to support remote access. As with the add-in solution, the server needs a remote access menu and the login authenticates the customer using the user database. A menu of available services is then built according to the customer's user profile. This menu consists of different remote services, but can in addition use some of the services accessible from the customer's Media Center. This is possible since these services are located at the server, not the MCE PC itself.

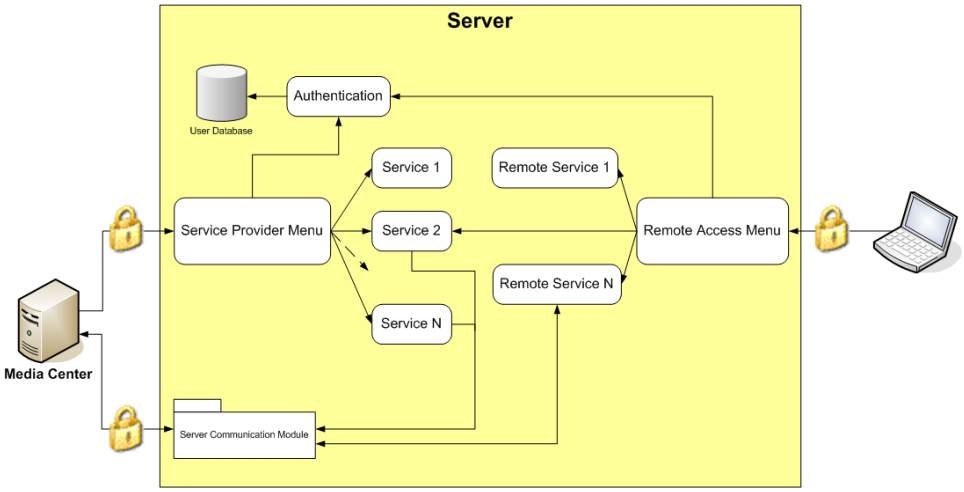

Figure 4.7: Web solution architecture, remote access

4.2.3 Communication Modules

So far we have proposed two possible architectures. In both cases the MCE PC and the server needs to communicate, thus both architectures include communication modules. These communication modules look the same for both architectures and client- and server-side, but the content differs. We will now describe the communication module and explain the differences between the add-in solution and the web solution.

The communication module holds a secure connection. To communicate, it sends and receives XML documents. When an XML document is received, a parse manager distributes the XML documents to the correct service parser.

This way, to add or remove a service you have a single place to hook or unhook it. In addition to communicating using XML documents, some services will want to download or upload data. In this case, the services will use a file transfer manager.

In the web solution all services accessed from the service provider menu resides at the server, thus some of the services do not need parsers neither at the MCE PC nor at the server. The services requiring parser will need much simpler parsers than in the ass-in solution. This gives us the difference between the two solutions' communication modules. The web solution has a less extensive communication module.

The communication module is shown in Figure 4.8. A connection manager accepts messages from the server or MCE PC and passes them on to the parse manager which in turn passes it on to the correct service parser. If a service, be it user-invoked or background, wants to send a message it passes it to the connection manager which handles the connection. To start a download or upload, the file transfer manager is used.

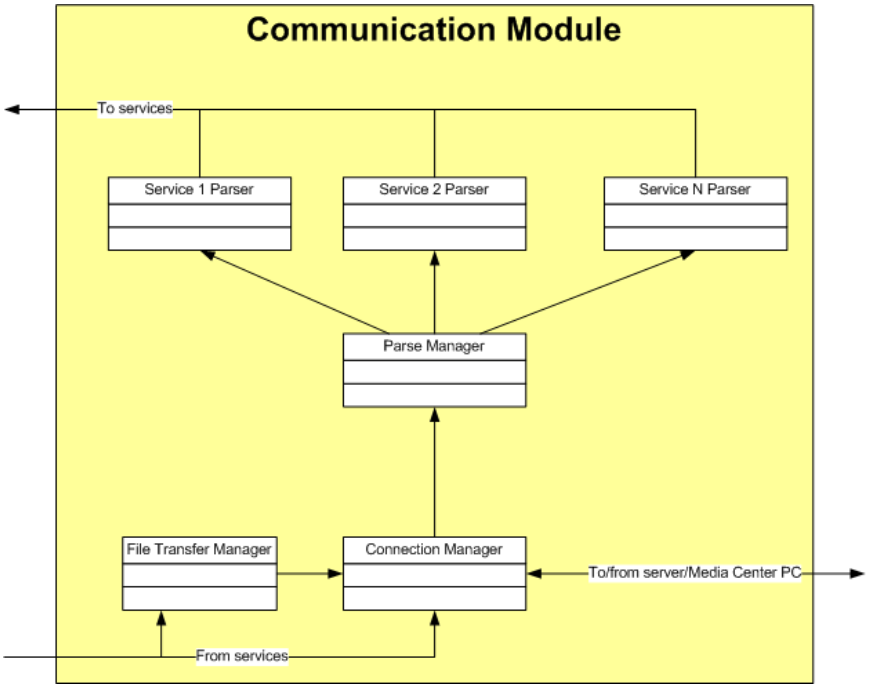

Figure 4.8: Communication Module

4.2.4 Running the background services

Both the add-in and web solution require background services. In example, the communication module must be run as a background service. It is two ways of achieving this. Either you implement the background services as background running Media Center add-ins, or you create them as Windows services. The resulting difference is at what time the background services are running. Implemented as Media Center add-ins, the background services run as long as Media Center is running. As Windows services they run as long as the MCE PC is turned on.

This choice has a notable affect on the functionality of the system. If the background services run as Windows services, some services might be available even if the Media Center is not running. For example, file transfer could be active. If one wants to make sure a service is running even if the customer closes his Media Center, it should be implemented as a Windows service. Implemented as Media Center add-ins, the customer has the opportunity to close all connection to the service provider by simply closing his Media Center. Some services will require the Media Center to be running. These would not be affected of whether the background services are Windows services or Media Center add-ins.

It is, of course, possible to use a mix of Media Center add-ins and Windows services.

4.3 Choice of architecture

We have proposed two possible architectures for the core system, the add-in solution presented in Section 4.2.1 and the web solution presented in Section 4.2.2. The web solution is a modification of the add-in solution, and thus both solutions have some shared qualities. They do, however, differ in a couple of areas. These differences will determine the choice of architecture.

With the add-in solution, the service provider menu and user-invoked services are stored locally. This makes the menu loading independent of the customer's internet connection. In addition, the service provider might offer services that can run locally at the customers MCE PC. A game service might be feasible to offer this way. Once installed, such a service will not use any bandwidth. When many customers download a little less data, the requirement of the service provider's bandwidth is reduced. For the customer, services such as multimedia-on-demand will decide the need for bandwidth. In comparison with the web solution, the add-in solution can reduce the service provider's need for bandwidth. The impact for the customers is small,

but might result in a bit quicker menu navigation.

Both solutions use the same communication channel between their communication modules. The security aspect of this communication is the same in both solutions. In addition the web solution sets up an HTTPS connection when a customer accesses user-invoked services. It is probably feasible to use certificates to create this secure connection. Because the service provider menu is a normal web site, it can actually be loaded in any web browser on any computer provided you have the right certificate. If someone wishes to illegally gain access to the system, they would have two possible entry points with the web solution, while only one with the add-in solution. Because of this, the web solution will require more attention to security.

The web solution makes it possible to reuse the user-invoked services as services offered by remote access. This will not be feasible for all user-invoked services, but there might be user-invoked services the service provider also wishes to offer by remote access.

When it comes to updates, the web solution has advantages. To change the selection of user-invoked services a customer has access to, it is merely a question of changing the customer's profile. No updates are required at the customer's MCE PC. Updates to the services only require the server to be updated and again no updates to the customer's MCE PC are required. In both solutions the updates to the background services will require updates to the customer's MCE PC. The add-in solution's background services include handlers for the user-invoked services. Thus the background services will require more frequent updates with the add-in solution than with the web solution.

Add-in solution	**Web solution**
Less bandwidth dependent	Possible to reuse user-invoked services in remote access
Easier to keep secure	Less updates to the customer's MCE PC

Table 4.1: Advantages of the two possible core architectures

Table 4.1 summarizes the advantages of both architectures. The add-in solution is less bandwidth dependent, but this is of minimal impact for the customers. The services run will put orders of magnitude more demand on bandwidth than what is saved, and this demand will be equal with both solutions. With internet connections only getting better, the saving of bandwidth with the add-in solution is of moderate impact. It is less work keeping the add-in solution secure. However, running a secure web site is not a new

discipline, thus this should not be a deciding factor. With the web solution it is possible to reuse user-invoked services in remote access. While this is a nice feature, but it would not be too much work to recreate a user-invoked service as a remote access service if this reuse was not possible. The most important aspect of the architecture is the ease of updating services and changing the selection of services offered. At this area the web solution is by far the best architecture. Based on this, our core architecture will be the web solution.

4.4 Remote access of local files

Section 4.2.2 presented the architecture of the web solution and how this architecture would be extended to support remote access. To offer a full range of remote services, accessing files on a customer's MCE PC is required. The Media Center PC could be considered as a multimedia server of the home. As discovered in Chapter 2, the ideal digital home should meet the customers' expectation of everything to be available from everywhere. Providing the customers with remote access to their local files brings us closer to the ideal digital home.

There are mainly three different architectures for accessing local files. The first alternative is to always retrieve the desired file from the MCE PC when a customer requests it. Alternatively all shareable files are uploaded to the service provider in advance and stored on their server. A third alternative is a hybrid of these solutions. We will now take a closer look at these alternatives to find which is best suited for our system.

Retrieving the files a customer request when he requests them is a very simple and straight-forward architecture. When a customer requests a file, this file is retrieved from his MCE PC and passed on to the customer. No storing of the file is performed by the service provider. As long as the customer has an ok broadband internet connection, the wait should be within acceptable limits. Audio and video files can be streamed, thus the customer does not need to wait for the entire file to be transferred. For slower internet connections, the delay of transferring files can be a problem. Another important thing to notice with this architecture is the need for the customer's MCE PC to be online. If the MCE PC is offline, no files can be retrieved.

The second alternative is to upload and store all files a customer wishes to have access to remotely at the service provider's server. This upload should be performed automatically, uploading all files from some predefined folders (My Pictures, My Music etc.). When files are added or removed, this should be reflected at the server, but not necessarily immediately. It would also

be possible to remove the local copy, limiting the customers' need for local storage capacity. When a customer requests a file, it is retrieved from the service provider's server, thus this solution does not rely on the internet connection to the customer's MCE PC. The customer's MCE PC does not even need to be online. For a service provider, this architecture is a lot more complicated than the straight-forward on-demand retrieval. Firs of all, it requires a large amount of storage capacity. It is, however, possible to take advantage of the fact that many customers probably will have some of the same files. For example, if two customers have the same song in the same quality, the service provider does not necessarily need to store two copies of it. Second, DRM is an issue. The service provider must decide whether or not to verify the legality of the customers' media files. Fast [4] is working on a product called mDisk, which among others include such functionality as described in this paragraph. We will describe mDisk closer in Section 8.1.

A third alternative is a hybrid of the two architectures described so far. Files are retrieved on-demand, but stored at the service provider's server when they are retrieved. When a customer request a file, the file is only retrieved from the customer's MCE PC if it is not already on the server. This way, popular files are stored at the service provider's server and are accessible even if the customer's MCE PC is offline. This solution require some amount of storage capacity, but the size of the "popular cache" each customer has available can be much less than what is offered with the complete server storage solution.

For our system we find the hybrid solution to be most fit. This solution is fairly straight-forward, but still offers quick retrieval of popular files. The complete server storage is a great solution with a lot of potential. However, it is a big task, too big for us to take on in this master theses.

4.5 System requirements

Based on the architectural choices made so far in this chapter, this section presents what we view to be the requirements of the system. The requirements are divided into non-functional- and functional requirements. In addition, three example services and their requirements are presented.

In the next chapter we will design a prototype. When designing this prototype some of the requirements stated here will be disregarded. We will look into these simplifications at the end of this section.

4.5.1 Non-functional requirements

Constraints set by the environment and external factors influencing the system are presented here. These non-functional requirements are given unique numbers starting with "NFR", and are listed below.

NFR1 Windows Media Center must be used as the operating system in the nerve center.

NFR2 The GUI of Media Center add-ins must be created for 10-feet-interaction.

NFR3 All GUI must be easy to navigate.

NFR4 The services must have a low response time.

NFR5 Copy protected media must be subject to DRM.

NFR6 The service provider's internet connection must be of such a manner as to offer good quality-of-service to all its customers.

NFR7 The service provider must be able to store the most recent files a customer requests through remote access.

4.5.2 Functional requirements

This section presents the functional requirements. We start of by stating the requirements applying to the system as a whole. We then move on to requirements concerning user-invoked services, background services and finally remote access.

Each functional requirement is labelled with "FR" followed by a unique number.

General

The general functional requirements apply to the system as a whole. These requirements must always be fulfilled, and should be kept in mind when any part of the system is reviewed. The general requirements are as follows:

FR1 All services the customer subscribes to must be available to the customer.

FR2 The customer must not by any means be able to access services he/she does not subscribe to.

FR3 The service provider must be able to easily update services.

FR4 The service provider must be able to easily add services.

FR5 The service provider must be able to easily remove services.

FR6 The services must be up to date at any given time.

FR7 All menus must be up to date at any given time.

FR8 The customer should be able to view his/her account information.

FR9 The customer should be able to change his/her account information manually.

FR10 All communication between the service provider and the customers must be secure.

User-invoked services

User-invoked services are services the customer actively accesses from his Media Center. In addition to the general requirements, user-invoked services imply a set of extra requirements. These requirements are listed below.

FR11 The main service menu must be available from the Media Center main menu.

FR12 The customer must be able to select and access services from the main service menu.

FR13 The customer must be uniquely identified through the use of X.509 certificates.

Background services

Background services are services automatically running on the customer's MCE PC. These services operate without the customer's explicit control. The requirements implied by the background services are as follows:

FR14 Background services must consist of one or more MCE background add-ins (i.e. add-ins that launch when the Media Center launches).

FR15 Background services can also consist of one or more Windows service which communicates with the MCE add-ins.

FR16 A secure connection for communication with the server must be established and maintained by the background services.

FR17 The background services must be able to create secure temporary connections with the server if needed.

FR18 The background services must be able to receive and process messages from the server.

FR19 The background services must be able to receive files from the server.

FR20 The background services must be able to upload files to the server.

FR21 The background services must be able to detect and install updates to the background service software.

Remote access

Remote access concerns services a customer can access from locations outside his home. Remote access gives us these requirements:

FR22 The service provider must maintain a secure web site for the customers to log in to.

FR23 The web site must provide the online/offline status of the customer's Media Center.

FR24 The web site must provide a remote services menu.

FR25 The customer must be able to select and access services from the remote services menu.

FR26 Services requiring communication with the customer's Media Center must not be accessible if the Media Center is offline.

4.5.3 Example services

To illustrate the functionality and possibilities of our system, a few examples are needed. We have chosen three example services which will be described and their requirements stated. The first two are picture download and picture upload. We chose to use pictures because they are quick to transfer and make a good illustration. The services could just as well have transferred or streamed music or video. The third example service is a TV-scheduling service.

Download pictures to Media Center

To illustrate the possibility of multimedia-on-demand services, we have chosen a picture download service. In the real world, a video or music service would be more attractive, but pictures are great for a quick demonstration.

Picture download will be a user-invoked service, accessible from the customer's Media Center. The customer shall be provided with a list of pictures, and can select to download any of them. The selected pictures will be stored in the "My Pictures" folder of the customer's MCE PC. In addition, picture download shall also be available through remote access. This is to illustrate how user-invoked services can be reused as remote access services. To do this, we only need to create a suitable GUI for remote access as well as the GUI fitted for MCE.

The requirements of the picture download service are presented below.

FR27 The picture download service must have a list of available pictures.

FR28 A thumbnail of the selected picture should be displayed.

FR29 If selected for download, the picture must be sent to the customer's MCE PC.

FR30 Downloaded pictures should be stored in the "My Pictures" folder.

FR31 If a picture of the same name as the downloaded picture already exists, the downloaded picture must be given a new and unused name.

FR32 The picture download service should be available both from the customer's Media Center and by remote access.

Upload pictures from Media Center, remote access

One great feature of remote access, is the possibility of accessing files from your MCE PC. To show this feature, we have selected to create a picture upload service. This service shall retrieve the list of pictures in the "My Pictures" folder of the customer's MCE PC. Pictures selected for upload will be retrieved, stored on the service provider's server and displayed in the web browser used for remote access.

As with download, we chose to use pictures to give a quick demonstration. As said introductorily, we could just as well have streamed music or video. This is where the power of this service is really exposed.

The picture upload service poses the following requirements:

FR33 The picture upload service must retrieve a list of available pictures on the customer's MCE PC.

FR34 The list of pictures should consist of all pictures in the "My Pictures" folder.

FR35 When selected for upload, the picture must be uploaded to the service provider's server.

FR36 When the picture is uploaded, it should be displayed in the customer's web browser.

FR37 If a picture selected for upload already exists on the service provider's server, this picture is used and no upload is performed.

Schedule TV-recording, remote access

Remote access can be used to control your Media Center. To show this, we will create a service that lets a customer schedule TV-recordings remotely. To do this the customer must be provided with an Electronic Program Guide (EPG). When programs are selected for recording, the customer's Media Center is told to add the recording to its schedule. The requirements of this service are listed below.

FR38 An EPG of the channels the customer subscribes to must be available.

FR39 The customer must be able to browse the EPG.

FR40 The customer must be able to select programs he/she wishes to record.

FR41 The customer must be able to select if he/she want to schedule one single show or a series.

FR42 Schedule requests must be sent to the customer's Media Center.

FR43 The customer's Media Center must add the schedule request to its scheduled recordings list.

4.5.4 Prototype simplification

In the following chapter we will design a prototype. The goal of this prototype is to test and verify the architecture proposed, not to create a perfect commercial system. Because of this, a few simplifications are made when

creating the prototype. These simplifications do not affect the demonstrational power of the prototype. It will be fairly straight forward to alter the prototype at a later stage if one does not want to perform these simplifications.

The list below shows which simplifications are made to the functional requirements. No simplifications are made to the non-functional requirements.

FR8 *Regarding viewing of account information.* It would be trivial to add account information. Since it does not affect the architecture in any way, we choose not to implement this functionality.

FR9 *Regarding changing of account information.* This requirement is dropped due to the same reason as for FR8.

FR10 *Regarding secure communication.* The background services will not use secure communication. For quicker implementation we use Sockets in stead of SecureSockets.

FR16 *Regarding connection to the server.* The requirement is only partially met, since the communication is not secure.

FR17 *Regarding temporary connections to the server.* The requirement is only partially met following the same reason as for FR16.

Chapter 5

Designing the prototype

The architecture presented in Chapter 4 contains server software and client software as well as services represented as web pages. This chapter presents the design of the prototype. First the package and class descriptions of both the server and client are given along with package and class diagrams. Next the client-server communication is explored. Following this the file transfer protocol used in the prototype is described.

The updates of both single services and the client software as a whole are important aspects of this project and are presented in Section 5.5. Next the top-level design of the service web pages is given. The last section of this chapter presents some security issues faced by this project and how these are handled in the prototype.

The server software is developed in Java, the client software is developed in C# and the service web pages uses JSP. For a justification for these and other technological choices, see Section 6.1.

5.1 Server design

This section describes the design of the server software. First an overview of the server design is given and the different packages are described. Following this overview is a more detailed textual description of the different classes of the different packages, as well as class diagrams for the different packages.

The class diagrams presented in this section are meant to give an overview of the system rather than to give a complete description of all classes with every attributes and operations. Thus only the most important attributes and methods of each class is shown in the diagrams, and some utility classes

are not shown at all.

5.1.1 Server package descriptions

The server will be implemented in Java, and the classes of the server are divided into different packages. The package diagram of the server can be seen in Figure 5.1, and the packages of the server are described below:

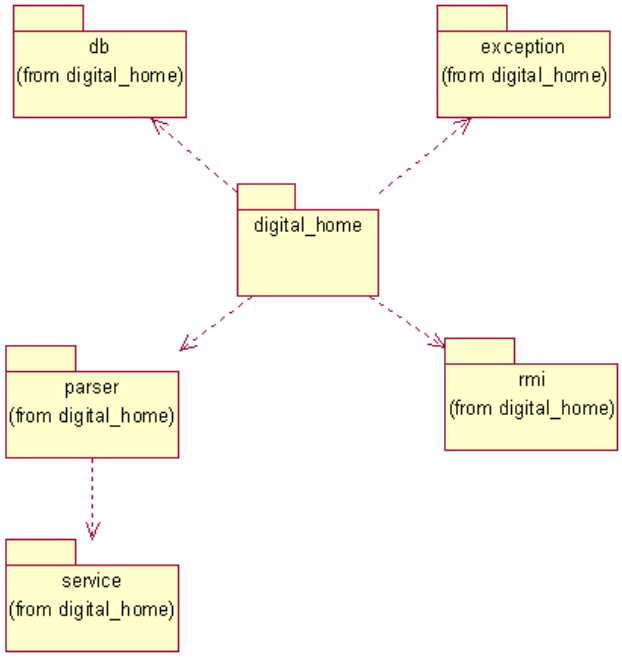

Figure 5.1: Package diagram for the server

digital_home This is the main package of the server software. This package contains the main classes for the server logic as well as classes needed for client-server communication and file transfer.

digital_home.parser This package contains all classes involved in parsing of XML documents. This includes a parse manager, a general XML service parser and all the specific service parsers.

digital_home.service This package contains all classes belonging to specific services.

digital_home.rmi This package contains the class and interface used at the server for RMI, as well as all classes used by JSP services to make RMI calls to the server.

digital_home.db This package contains the classes used as a substitute for a real service provider's database.

digital_home.exception This package contains the different exceptions used in this project. This package is not explored in the following sections.

5.1.2 Server class descriptions: digital_home

The package digital_home is the main package of the server. The most important classes of this package are shown in the class diagram in Figure 5.2. Some utility classes are left out. A description of all the classes in the root package of the server is given below:

Connection This is an abstract class containing everything that is common for all kinds of connections used by the server. This class implements Runnable since most connections are going to listen for data in a new thread.

ConnectionManager The ConnectionManager manages all connections used for XML message communication between the server and the different connected clients. The ConnectionManager listens for connecting clients and creates new XMLConnections as clients connect.

DownloadConnection A DownloadConnection is used by the server when a client wishes to download files from the server. When a client intializes a download, a new DownloadConnection with that client is created. The DownloadConnection contains a queue which allows the client to queue new files for download while waiting for the current download to finish.

EOFInputStream This is a utility class created by the staff of TTM4100 Communication - Services and Networks in 2002. This class is used to detect an EOF for the incoming stream of the XMLConnection. This class is not shown in Figure 5.2.

FileInfoTokenizer This is a utility class that takes as input a complete path of a file and gives lets the user retrieve the filename, extension and path element of the complete path. This class is not shown in Figure 5.2.

FileTransferManager The FileTransferManager manages all active DownloadConnections and UploadConnections to all clients. The FileTransferManager is also active in the file transfer process of both download and upload.

NetworkVariables This is a utility class containing static network variables such as URLs, port numbers etc. This class is not shown in Figure 5.2.

Server This is the main class of the server. The Server class contains the main start-up logic of the server system as well as acting as a connection point for all other classes.

ServerGUI This is a class extending a JFrame acting as the GUI of the server. The GUI contains lists of all active non-authenticated connections, authenticated connections and download connections. The GUI also contains both a log window and a menu. The menu allows among other things the administrator of the system to send out client updates.

UploadConnection An UploadConnection is used by the server when a client needs to upload files to the server.

UploadWaitController UploadWaitControllers are used to make the RMI call which initializes a file upload from the client to the server to wait until the file transfer is complete before returning.

XMLConnection An XMLConnection is the main connection between a client and the server. All communication except file transfer goes through here.

XMLVariables This is a utility class containing static variables used for creating and parsing XML documents. When creating new services, this class can be extended with the new variables containing information used to create and parse XML documents for that service.

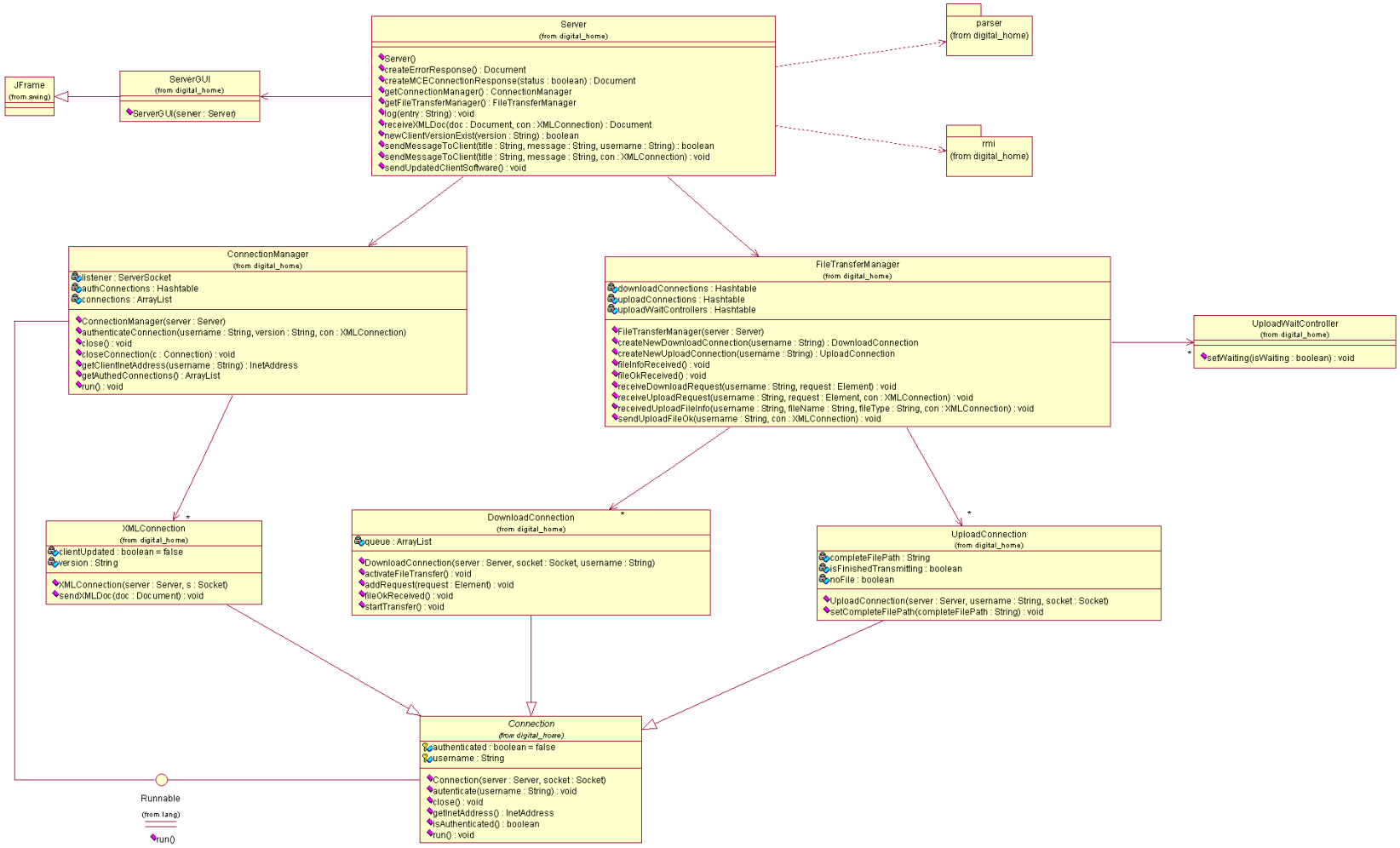

Figure 5.2: Class diagram for package digital_home

5.1.3 Server class descriptions: digital_home.parser

The digital_home.parser package contains all classes involved in parsing XML documents. A class diagram of this package can be seen in Figure 5.3. This diagram shows the parse manager along with the general parser class and a couple of actual parsers. The rest of the current parsers are not shown in the diagram since all parsers basically have the same structure. A description of the classes in this package is given below:

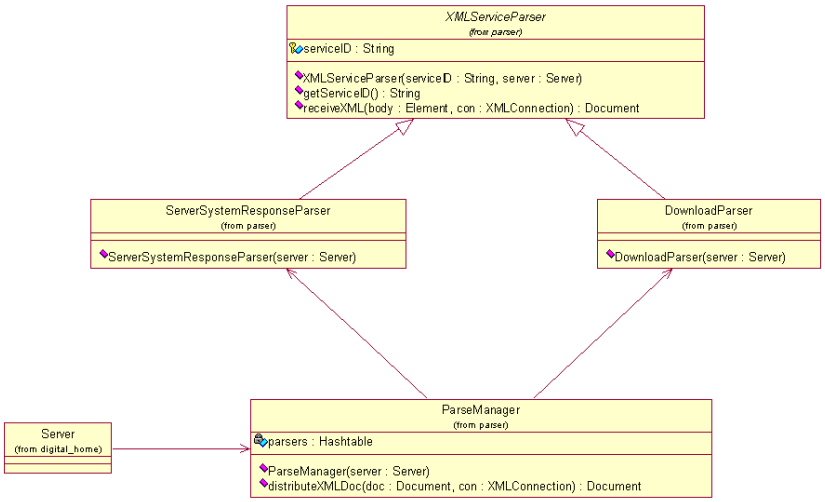

Figure 5.3: Class diagram for package digital_home.parser

DownloadParser This is the parser used for parsing the messages in the protocol used for downloading files from the server to a client.

FileListParser This parser is used for the service which retrieves different kind of file lists from a client.

ParseManager This is the main class in the digital_home.parser package. Referances to all parsers are kept here. This class receives all incoming XML messages and passes them along to the appropriate parser.

ScehduleTVRecordingParser This is the parser used by the service which allows users to schedule TV recordings through remote access.

ServerSystemResponseParser This is the parser used for all server system response messages.

UploadParser This is the parser used for parsing the messages in the protocol used for uploading files from a client to the server.

XMLServiceParser This is an abstract class defining that which is common for all parsers.

5.1.4 Server class descriptions: digital_home.service

The digital_home.service package contains all service specific classes. The only classes in this package at the present time are classes used by the service which retrieves different kind of file lists from a client. If special classes are needed for different services at a later time, they should be added to this package. No class diagram is shown for this package. A description of the classes in this package is given below:

FileListManager This class manages all active file list requests.

FileListService This class is used to make the RMI call which initialized the file list request to not return until the file list is received at the server.

5.1.5 Server class descriptions: digital_home.rmi

The digital_home.rmi package contains all classes used for RMI communication between the server and the service JSP web pages. A class diagram of the server side classes is shown in Figure 5.4. A description of the classes in this package is given below:

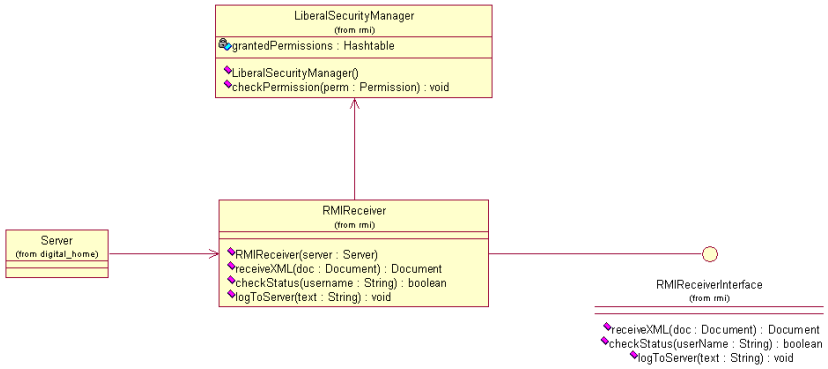

Figure 5.4: Class diagram for package digital_home.rmi

LiberalSecurityManager This class is created by the staff of TDT4190 Distributed Systems in 2003. This class is used to set a more liberal

security manager for RMI communication than the default security manager used in Java RMI.

RMIPictureService This class is used to create request for download of pictures from the server to a client's MCE PC.

RMIReceiver This class implements the RMIReceiverInterface. This is the RMI class of the server which have methods to check the online status of a client's MCE as well as receiving XML documents from a service.

RMIReceiverInterface This interface describes the remote methods available at the server by the class RMIReceiver.

RMIRemoteCommon This class has methods for creating different types of RMI requests which are common for several services. As of today, this class contains methods for cheching the online status of a client's MCE and a method for retrieving a list of files (any kind of files) from the client's MCE PC.

RMIScheduleRecording This is the class used by the TV recording scheduler to request a scheduling.

RMIUploadPictures This is the class used by the upload picture service to request an upload of a picture from the client's MCE PC to the server.

5.1.6 Server class descriptions: digital_home.db

The digital_home.db package contains the class simulating a service provider's database. The contents of these classes can be switched to contain access to a real database. To make sure all services work after such a change is made, the UserDatabase and the User class must stille exist along with the getUser() method of the UserDatabase and the checkForService() and checkForRemoteService() method of the User class. The class diagram of the database classes can be found in Figure 5.5. A description of the classes in this package is given below:

Service This class represents a service containing the ID of the service, the name of the service and the URL to the service JSP web page.

User This class represents a user in the database. There are methods available at the User class to check if the user is subscribing to different services.

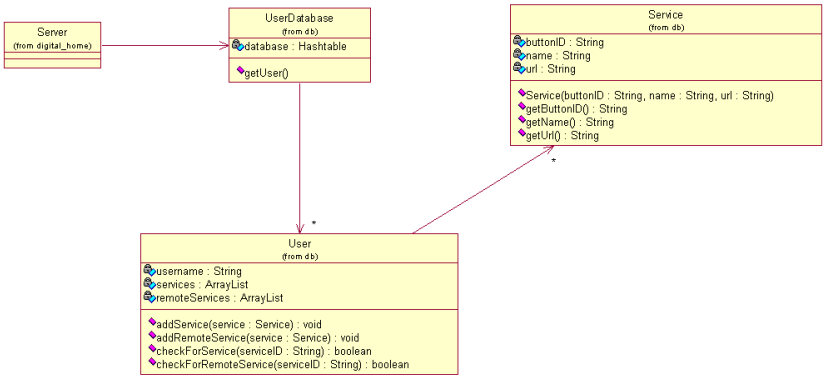

Figure 5.5: Class diagram for package digital_home.db

UserDatabase This is the main class of the database. The most important method of this class as of today is the getUser() method which retrieves the User object of a user with a given username.

5.2 Client design

This section describes the design of the client side of the system. This system have to be developed in .NET, and as stated earlier we have chosen the C# language to be used in this development. First in this section, the different namespaces (C# version of package) the client software is devided into are described. Next, a description of the different classes of the different namespaces is given as well as a class diagram showing the most important classes of the client.

As with the server, the class diagram of the client is only meant to give an overview of the most important aspects of the client design. This means that some classes, attributes and operations are not shown in the diagram.

5.2.1 Client namespace descriptions

The client software is divided into three different namespaces: MediaCenter.DigitalHome, MediaCenter.DigitalHome.parser and MediaCenter.DigitalHome.service. An UML package diagram is used to show how these namespaces are connected with each other. The package diagram can be seen in Figure 5.6. A description of the different namespaces is given below:

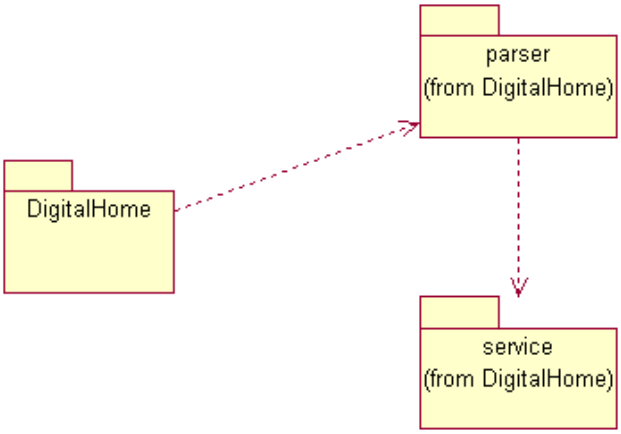

Figure 5.6: Namespace diagram for the client

MediaCenter.DigitalHome As with the server, this is the main namespace of the client. This namespace include the main logic of the client and all classes handling communication with the server.

MediaCenter.DigitalHome.parser This namespace contains all classes involved in the parsing of XML documents.

MediaCenter.DigitalHome.service This namespace includes classes used by different services.

5.2.2 Client class descriptions: MediaCenter.DigitalHome

As described earlier, this is the main namespace of the client software. In general, the client contains much fewer classes than the server and is less complicated. For this reason, there is only given one class diagram for the client containing all important classes from all namespaces. This diagram is shown in Figure 5.7. The classes at the client are a lot similar to those of the server and a description of them is given below:

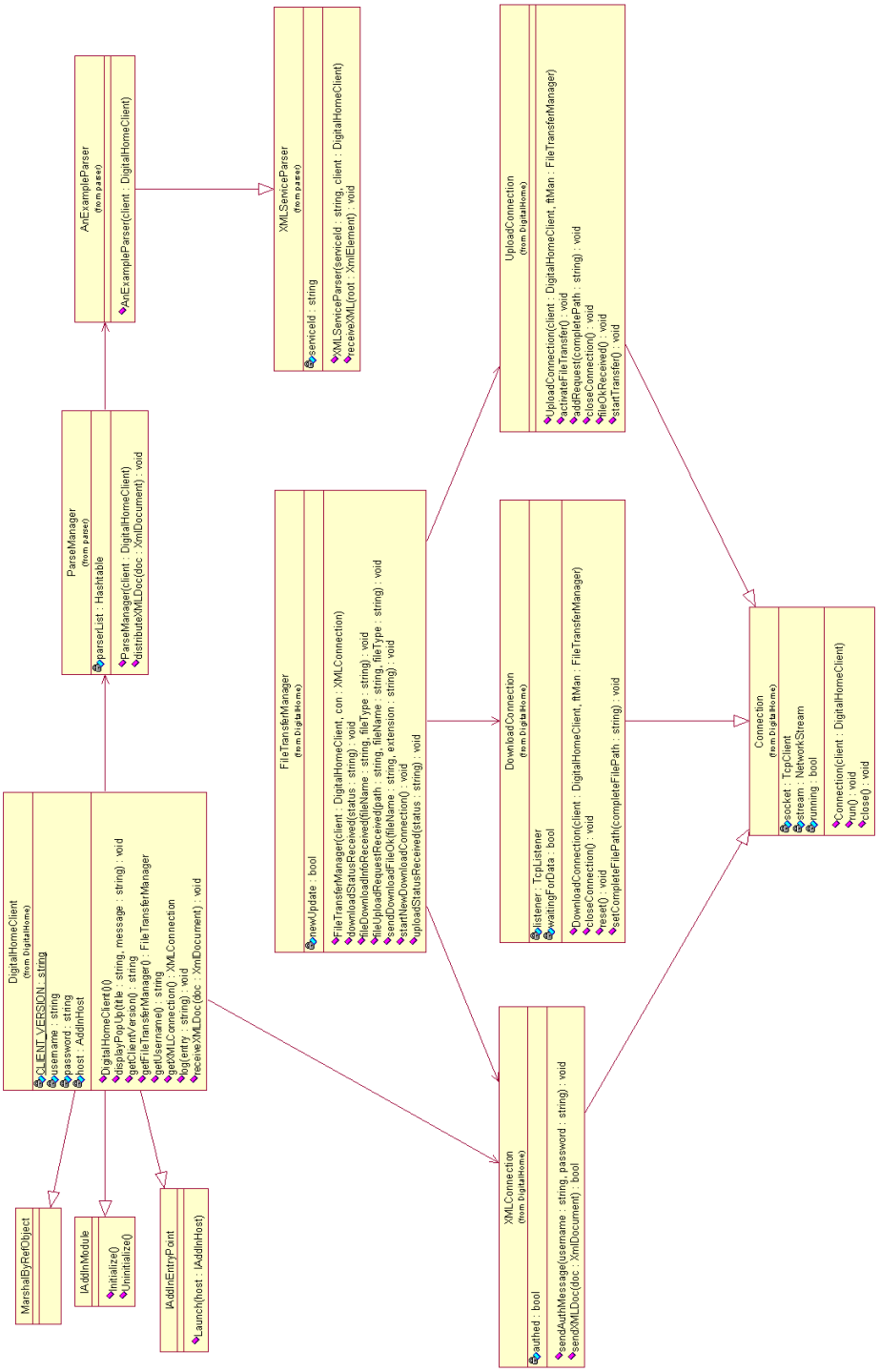

Figure 5.7: Class diagram for the client

Connection This is an abstract class containing everything that is common for all kinds of connections used by the server.

DigitalHomeClient This is the main class of the client. The client is developed as a MCE add-in. For this reason, this class extends MarshalByRefObject, IAddInModule and IAddInEntryPoint which contains methods called when the add-in is initiliazed and unitialized as well as the launch code of the add-in.

DownloadConnection The client contains one download connection. This connection waits for the server to connect to it when a download to the client is requested.

FileInfoTokenizer This is a utility class that takes as input a complete path of a file and gives lets the user retrieve the filename, extension and path element of the complete path. This class is not shown in the class diagram.

FileTransferManager The FileTransferManager manages the DownloadConnections and the UploadConnection of the client. The FileTransferManager is also active in the file transfer process of both download and upload. Updates to the client software is also done by the FileTransferManager.

FileTypeResolver This is a utility class used to resolve the file type of downloaded files. The FileTypeResolver has methods which checks if the extension given as input is classified as different types of files (e.g. picture, audio etc.). This class is not shown in the class diagram.

NetworkVariables This is a utility class containing static network variables such as URLs, port numbers etc. This class is not shown in the class diagram.

UploadConnection The client contains one download connection. This connection waits for the server to connect to it when an upload from the client is requested. The UploadConnection class also contains a queue of files that awaits to be uploaded to the server.

XMLConnection An XMLConnection is the main connection between a client and the server. All communication except file transfer goes through here.

XMLVariables This is a utility class containing static variables used for creating and parsing XML documents. When creating new services, this class can be extended with the new variables containing information used to create and parse XML documents for that service.

5.2.3 Client class descriptions: MediaCenter.DigitalHome.parser

The MediaCenter.DigitalHome.parser namespace contains classes used for parsing XML documents at the client. The main class in this namespace is the ParseManager class. The ParseManager along with the XMLServiceParser class and an example parser is shown in the class diagram in Figure 5.7. The rest are parsers for different kinds of services. A description of the classes in this namespace is given below:

FileDownloadParser This is the parser used for downloading files from the server to the client.

FileListParser This is the parser used to send file lists to the server.

FileUploadParser This is the parser used for uploading files to the server.

MessageToUserParser This class parses messages which the server wants to be displayed as a pop-up box at the client's Media Center.

ParseManager This is the main class in the this namespace. Referances to all parsers are kept here. This class receives all incoming XML messages and passes them along to the appropriate parser.

ScheduleTVRecordingParser This is the parser for the TV recording schedule service.

ServerSystemResponseParser This is the parser for server system response messages.

XMLServiceParser This is an abstract class defining that which is common for all parsers.

5.2.4 Client class descriptions: MediaCenter.DigitalHome.service

The MediaCenter.DigitalHome.service namespace contains classes that are service specific. At the present time, only one service exists in this namespace, but new services should be added into this namespace as needed. The service class in this namespace is not shown in any class diagram. A description of the classes in this namespace is given below:

ScheduleTVRecording This is the service which performs the scheduling of TV recordings requestet through remote access.

5.3 Client-server communication

This section describes how the client and the server communicate in our proposed architecture through the sending of XML documents. The idea is to have the client software establish a connection with the server when the client's Media Center is started. This connection is of the type XML-Connection and lets the client and the server communicate by sending XML documents back and forth. As long as the client's Media Center is running, this connection is uphold.

When the client first establishes a connection with the server, the server has no way of knowing which client connected. Thus the XMLConnection is treated as not authenticated at the server. All incoming messages to a non-authenticated XMLConnection is disregarded with one exception. The client can authenticate itself by sending a authentication message including username and password. If such a message is received at the server, authentication of the client's connection is performed and a response with the result of the authentication is sent back to the client.

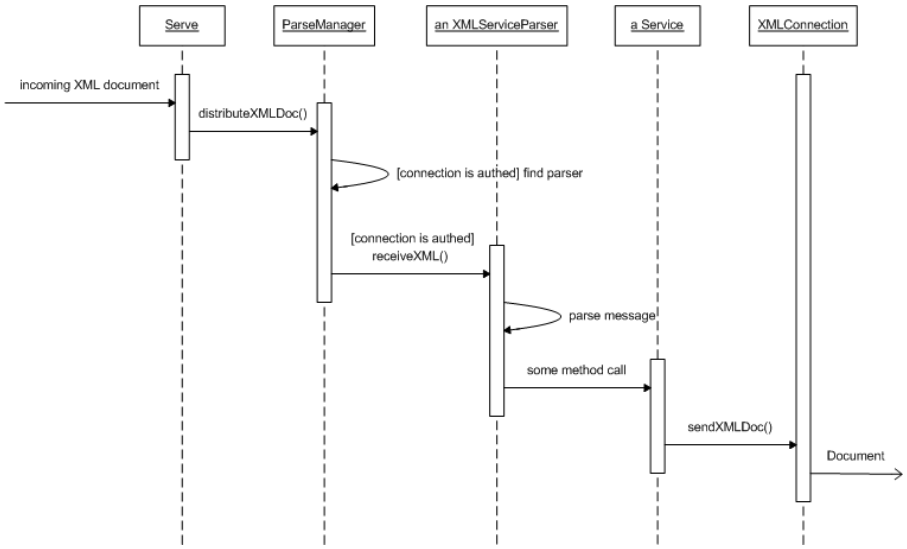

Figure 5.8: Sequence diagram of server XML communication

The rest of this section is dedicated to show how the XML message transfer is performed at the server and the client. We will start by looking at the reception of XML messages at the server side. A sequence diagram for this is shown in Figure 5.8. The process starts by a message source calling the receiveXMLDoc() method on the Server. The message source may either be an XMLConnection or the RMIReceiver. The XML document is then forwarded

to the ParseManager. The ParseManager checks if the XMLConnection is authenticated. If it is not, and the message is not an authentication message as described previous, the message is disregarded. If the connection is authenticated, the message root is parsed and the message body is forwarded to the appropriate parser. The parser then parses the message and for the most cases this will result in some method call on the service to which the parser belongs. This service will perform actions as described in the message and may optionally send a response back to the client.

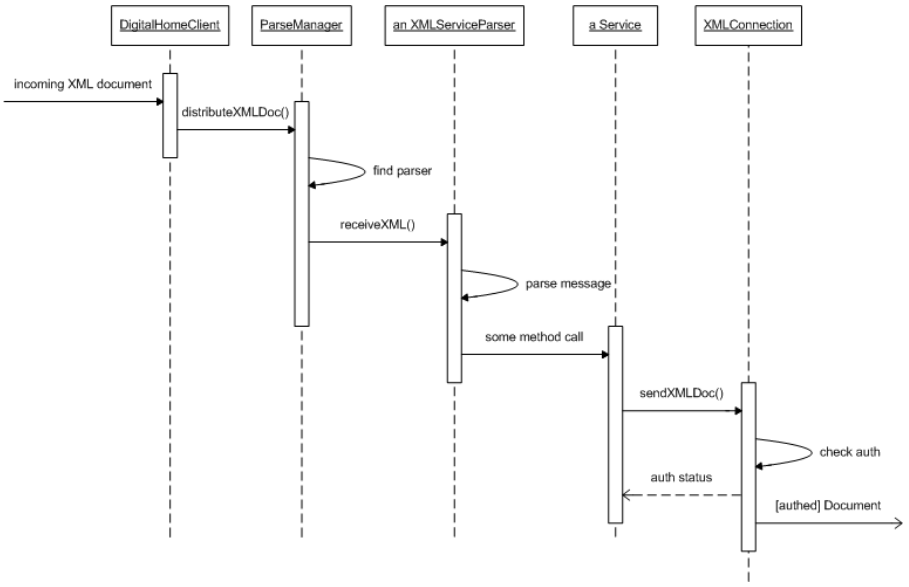

Figure 5.9: Sequence diagram of client XML communication

The reception of XML messages at the client side is quite similar to that of the server side. A sequence diagram for the client side reception is shown in Figure 5.9. The client does not send out any type of XML messages except from authentication messages if it is not authenticated. Thus the server should never receive any other kind of messages from non-authenticated clients than authentication messages, and this is the reason that these messages can simply be disregarded at the server side.

5.4 File transfer

Although it is not the focus of this project to develop the actual services, it is apparent that many services will be in the need of a file transfer function. For this reason, we have developed a file transfer protocol which can be used by

our example services. It may be noted that the actual file transfer is rather naive with little or no error handling and no option to resume transfering of partially transferred files. In a commercial system this part of the file transfer system may be substituted by a more sophisticated solution. We will in this section focus on showing how downloads from the server to the client is handled at the client and at the server. Uploads from the client to the server is done in a similar fashion with reversed roles for the participants.

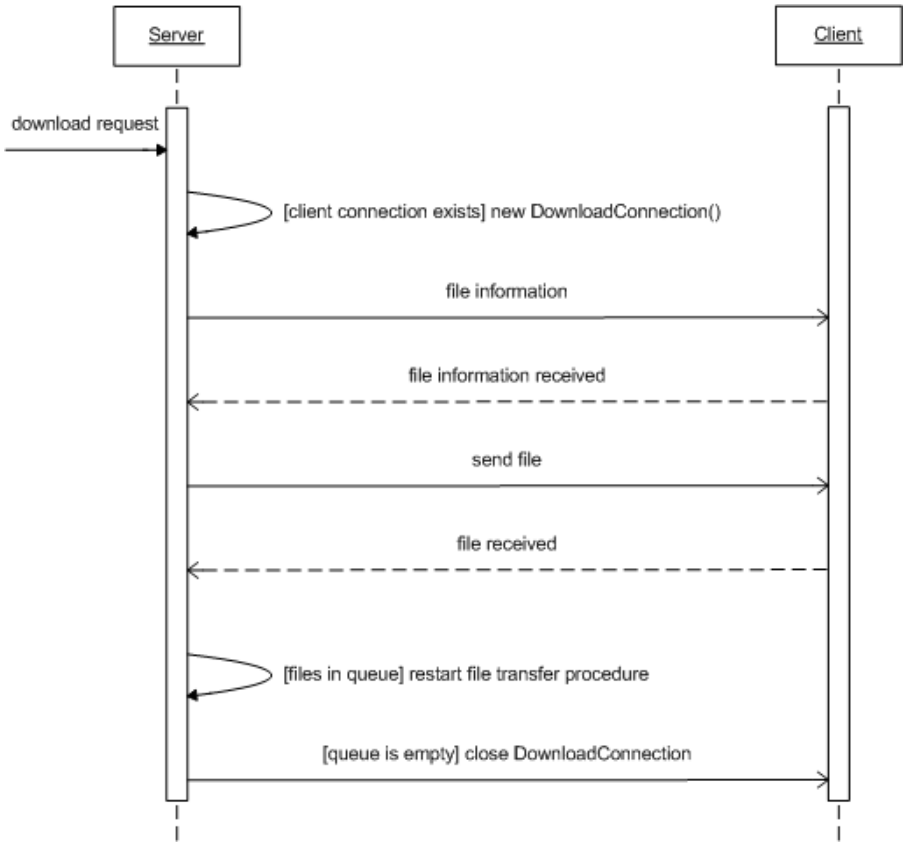

Figure 5.10: File download protocol

Figure 5.10 shows the basics of the download protocol. When a download request is received at the server, a new DownloadConnection is created if there does not already exist a DownloadConnection to that client. Then an XML message with the file information is sent to the client. When the client has received and stored this information, it sends a message back to the server informing the server that the file information is received. The client is now ready to receive the file. The file is transferred from the server to the client. When the client has finished receiving the file, it sends a message informing the server that the file has been received. If the server has more

download requests queued for that client, this file transfer process is restarted with the next file in queue. If the queue is empty, a message is sent to the client instructing it to close the connection used for downloads.

How this file transfer actually is done at both the server and the client can be seen in the sequence diagrams in Figure 5.11 and Figure 5.12.

The whole process starts with the FileTransferParser invoking the method receiveDownloadRequest() at the FileTransferManager. The FileTransfer-Manager checks if the client already has an active DownloadConnection and creates a new one if no such active connection exists. The download request is then queued at the connection and the startTransfer() method is called if this is a new DownloadConnection. The protocol described above is then initialized by the server by sending the file information to the client. What is worth noting is that when the server prepares to start sending the file to the client, the transfer of the file is done in a separate thread at the server, thus not blocking the XMLConnection thread which initialized the file transfer. When the server receives the message from the client that the file has been received, the server removes the current request from the queue and checks if there are more requests queued up. If there are more requests in the queue, the file transfer process is restarted. This is done by recalling the sendFileInfo() method on the DownloadConnection, which prepares the first element of the queue for file transfer by sending the file information to the client. If the queue is empty, the close method is called and the Download-Connection at the server side is closed. When the close message is received at the client, the DownloadConnection is reset and awaits a new connection from the server.

5.5 Updates

Updates are a central aspect of our system. With the chosen architecture, the installation of new services and removal of old services are mostly done only at the server side. Section 5.5.1 presents how services are updated. Some service updates may also require updates to the client software. How client software updates are performed in our system is described in Section 5.5.2.

5.5.1 Service updates

The system lets the users subscribe to different services from a service provider. When the user accesses the "Digital Home" menu of his Media

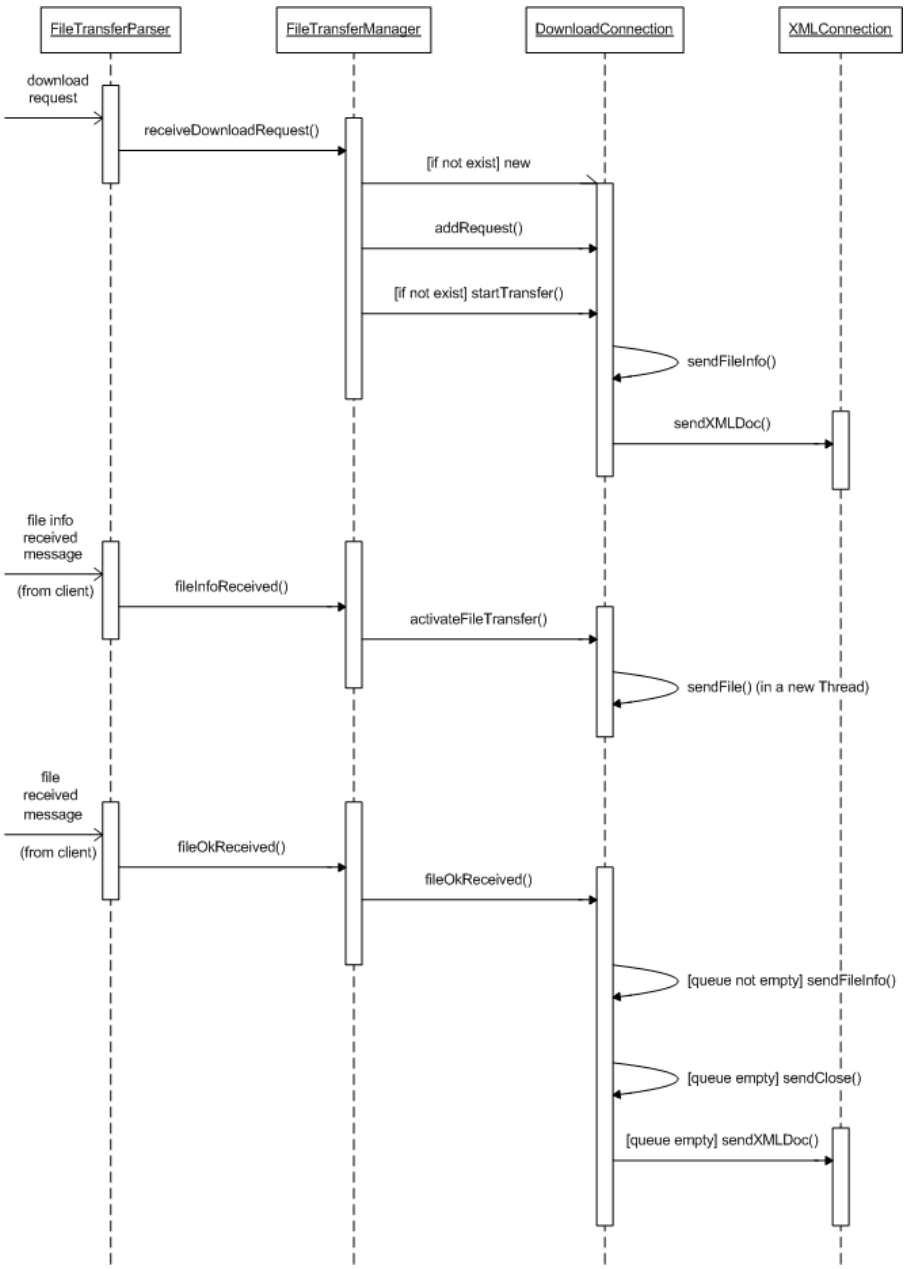

Figure 5.11: Sequence diagram of file download at server

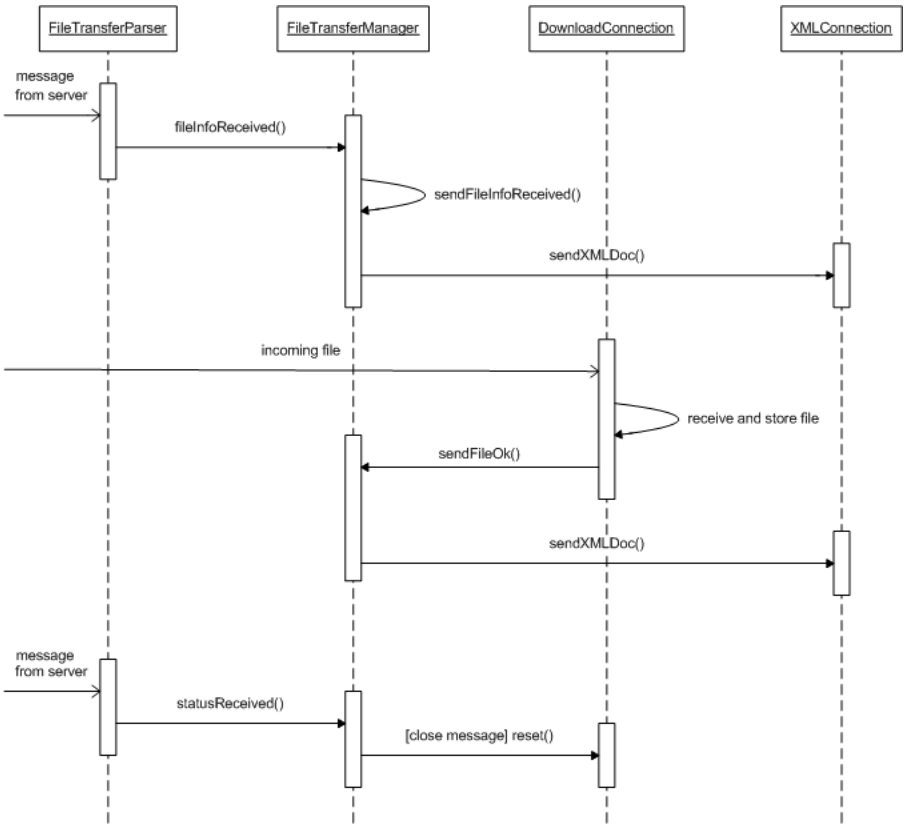

Figure 5.12: Sequence diagram of file download at client

Center or logs into the remote access page, the system will retrieve the list of services the user subscribes to from the database and present them to the user. All of these services are represented as JSP web pages at the server. Thus to add a new service, the service provider can add the service JSP page to the web server and add to the database which users are subscribing to that service. The next time one of those users enters the "Digital Home" menu, he immediately sees the new service listed there. To remove a service, similar actions would be performed. To update the content of a service, the service provider only needs to install an updated version of the JSP page on the web server. The next time any user accesses that service, he will gain access to the new version.

All updates discussed so far do not require any changes of either the server or the client software. Many services will need to use the server and client software to function properly. Thus, new parsers for these services and perhaps also specific service classes must be implemented and added to the system. The system is designed to make it easy for the service provider to add and remove services. Thus to add a new service to the existing system, all the service provider has to do is to create an instance of the new parser in the ParseManager class and add the parser to the list of parsers. When this is done both at the server and the client, the new service will be able to perform client-server communication. However, when adding new services to the system in this manner, the customers will need to install new client side software. How this is done is described in the next section.

5.5.2 Client software updates

When new client software is available, the new client software library file is placed in the in the update directory of the server and the version number is updated in the version.txt file. When the administrator of the server selects to update the client software, the server reads the new version number from the version.txt file and pushes the update to all online clients that need it. To make sure all clients get the new update as fast as possible, when a new client authenticates itself at the server the version number of that client is compared to the latest client version number read by the server. If the client is in a need of an update, the update is sent. Before explaining more about the update protocol, a description of how updates of client software is performed.

Media Center hosted add-ins must be installed in the Global Assembly Cache (GAC) [5] to run. Microsoft support two ways of adding libraries to the GAC, through an installer or by the use of gacutil.exe. It is a third alternative too. Gacutil.exe is actually a .NET program, so it is possible to install

libraries to the GAC using self-created code. However, there is no publicly available API for this. Hence, Microsoft never intended for anyone outside Microsoft do this. Since the classes and methods needed are not publicly available, Microsoft can alter them without ever releasing any information of it. Because of this, it would not be wise to use the third alternative.

Installers require user involvement while gacutil.exe can be run automatically. When a customer installs the client software for the first time, the installer puts the needed library into the GAC. When updates are performed, we do not want to involve the customer. Because of this we use gacutil.exe to replace the library in GAC with the new updated library.

Since gacutil.exe is not a standard windows component, the update protocol first need to make sure that the client is in possession of this program. Thus, the server starts the update procedure by sending file information about gacutil.exe to the client, starting a file transfer process of that file. The client checks if gacutil.exe already exists in the destination folder. If there is no need to transfer this file, the client sends a message informing the server that the gacutil.exe has been received. If the gacutil.exe file does not exist at the client, the file transfer is continued. When the server receives a message that the gacutil.exe is received at the client, the server starts a new file transfer process with the client software library file. When the new client software library file is received, gacutil.exe is used to replace the old client library in the GAC with the new file. A pop-up box informs the customer that new client software is installed and that the customer needs to restart his Media Center the changes to take affect.

The server keeps track of which online clients that have installed the updates and which that have not. This makes it possible for services at the server to check the client version number before sending any messages to that client. Thus, this feature makes it possible to make certain services unavailable until the client software has been updated to the proper version and the Media Center restarted.

5.6 Service JSP web pages

Both the user-invoked services accessible through the customer's Media Center and the services accessible through remote access are JSP web pages residing on a Tomcat web server. This section describes the top-level design of these service pages.

The user-invoked services are pretty much standard JSP web pages which include some special MCE JavaScript, CSS and picture files. In this project,

we have chosen to use a third party version of the special MCE files to make the service pages look similar to the rest of the Media Center. This third party package can be retrieved from The Green Button [33].

Both to make the communication with the server secure as well as identifying a customer's MCE, the web pages uses X.509 certificates. Each customer will have a certificate with a unique common name. More about the certificates used in this project is found in Section 5.7. Each service JSP page must have a service name identifying the service. The service name is set at the top of the code by the variable serviceName. The common name of the certificate is retrieved and a check is performed to see if the customer has access to that specific service. Services which have the serviceName set to "main" are not checked against the user database. These services are typically main menus etc which are common for all customers. The code which performs the security check also retrieves a list of all available services to that customer. This list is used when building the main menu of the user-invoked services.

Most services will be in the need to communicate with the server software. This is done by the use of Ajax and Java RMI. Ajax is used to send asynchronous message calls to another JSP page. That means once such a call is sent, the web page does not wait for a response. However, once a response is received, some part of the web page is updated according to the contents response. Ajax makes the services look more like applications in that they will not wait for a response and that the whole web page does not need to be refreshed once the response is received. The service page sends the request to another JSP page which handles the RMI communication with the server and sends a response back to the service page. Some RMI calls to server must wait for the server to communicate with the client software to return an answer to the request. To achieve this, the thread which made the RMI call is put to sleep and awoken once the response is ready using the Java thread programming methods wait() and nofify().

The JSP pages used for remote access services are quite similar to the ones used for user-invoked services. The main difference is that these pages do not use the special MCE CSS and JavaScript files. The remote access pages also contain the same kind of security check as the user-invoked services and the menu of available remote services to the customer is built in the same manner. In addition to this security check, all remote service pages request the online/offline status of the customer's Media Center when they are loaded. This status is shown in the page and is also used to make certain service unavailable if the customer's Media Center is not connected to the server.

5.7 Security

Securing the communication between the clients and the server is important. Both to make sure that sensitive information like username and passwords does not fall into the wrong hands, but also to make sure that it is impossible for malicious programs to impersonate neither the client nor the server. Because of the nature of the file transfer protocol used in the prototype, it is possible for malicious programs knowing the file transfer protocol and the structure of the XML messages to both upload files to a client and download files from a client. Thus our system can actually work as a Trojan horse. Both the issue of data confidentiality and the issue of authentication can be solved by the use of public key cryptography and secure sockets. However, to save development time our prototype does not make use of this. If a commercial system of the prototype is going to be developed in the future, the sockets used for client-server communication must be made secure.

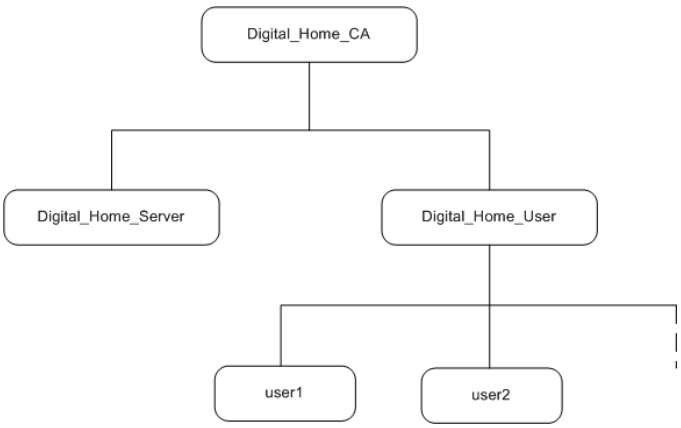

Figure 5.13: Certificate hierarchy

In addition to making the client-server communication secure, the web pages used both for user-invoked and remote services are also in the need of both data confidentiality and authentication. This is solved by the use of X.509 certificates. X.509 certificates make us of public key cryptography and can be used to ensure both data confidentiality and authentication. Both X.509 certificates and Public Key Infrastructure (PKI) is described in more detail in Section 6.3.3. Figure 5.13 shows the certificate hierarchy used for our prototype. The different certificates used in the prototype are as follows:

Digital_Home_CA: This is the top-level certificate used to sign the Digital_Home_Server and the Digital_Home_User certificates.

Digital_Home_Server: This is the certificate used by the Tomcat web server.

Digital_Home_User: This is the certificate used to sign the user certificates.

Different user certificates: These are the certificates used by the different customers.

The Digital_Home_CA certificate is self-signed, meaning this certificate must be added as a trusted root certificate in Internet Explorer to get the certificates to work properly. In a commercial version of our system, the Digital_Home_CA would not be self-signed, but rather signed by a professional certificate issuer.

In the last few years defending the rights of the owners of digital media has become an important issue. With the development of different peer-to-peer file sharing technologies like Napster, Gnutella and DirectConnect, the illegal sharing and downloading of digital media has become widespread. As a result to this trend, Digital Rights Management (DRM) has become an important aspect when developing applications that handles digital media. Since services which include some handling of digital audio and/or video are typical services which could be implemented for our system in the future, DRM is naturally also an important aspect in this project. DRM is further discussed in Appendix B.

Chapter 6

Developing the prototype

Chapter 5 describes the design of our prototype. In this chapter we will look at the technologies and Software Development Kits (SDKs) used when developing it. We will also describe the main code standards used.

We start by justifying the choice of technologies. With some parts of the prototype we do not have a choice of which technology to use. Other parts are totally up to us. Either way, we will describe which technologies are used and why. In addition to the standard SDKs for the programming languages chosen, we use two extra SDKs. These will be described. We then focus on other important technologies. The chapter finishes by looking at the main code standards used.

6.1 Choice of technologies

To develop the prototype, choices of technologies had to be made. This section describes these choices.

The client software must consist of at least one Media Center add-in. Windows Media Center is a Microsoft product. Not too surprisingly, one has to use Microsoft's programming environment, .NET, to develop Media Center Add-ins. Within the .NET environment there is a wide selection of languages which can be used. We have selected to use C#. C# is the premier language of .NET, and was written explicitly to support features found in .NET. It has also become the de facto language used in .NET-oriented training materials. Since we are new to .NET development, C# was the natural choice.

In contrast to the client software, the choice of technologies for server side is not restricted. The server side consist of server software and multiple

web pages, as described in Chapter 5. Many of the web pages will need to communicate with the customers' Media Center. This communication goes through the server software. Thus, both the choice of technology for the server software and for the web pages is connected. We looked into two possible combinations, Java for the server software and JSP for the web pages or .NET for the server software and ASP.NET for the web pages. For two reasons we chose Java/JSP. First, we are most familiar with Java/JSP, so this would make for a more rapid development. Second, this combination makes the server platform independent. For easy communication between the web pages and the server software, we use Java Remote Method Invocation (RMI).

The web pages accessed both from Media Center and remotely should feel as much as an application as possible. To the user, Media Center is an application. Thus the user-invoked services should not feel as web pages, but as an application. Ajax gives us this application feel, and the web pages use Ajax in combination with JSP.

6.2 Software Development Kits

The main Software Development Kits (SDKs) used are the Java and .Net SDKs. These SDKs are well known, and we will not discuss them here. In addition we use two extra SDKs, the Windows Media Center SDK and JDOM.

JDOM is used to work with XML documents in Java. We will take a brief look at JDOM. Windows Media Center SDK is an SDK used when developing for Windows Media Center. Since one should be familiar with this SDK when developing for Windows Media Center, it will be described in some depth. We start by describing Windows Media Center SDK and move on to look at JDOM.

6.2.1 Windows Media Center SDK

Microsoft has provided the MCE SDK for external developers to be able to create applications and software components that take advantage of features provided by MCE. The main areas of the SDK are:

- Media Center HTML applications
- Media Center hosted add-ins

- CD/DVD recording add-ins
- Media Center Extenders
- Status information about the state of Media Center
- Locale-specific input method editors (IMEs) for Media Center

The areas most interesting for the prototype are hosted add-ins and HTML applications. To create a more complete solution, with more services than just the few example services we have chosen, state information and Media Center Extenders are also worth noting. We will take a closer look into these four areas. For more information on the SDK, see [40]. The SDK can be downloaded from [42]

Media Center hosted add-ins

Media Center hosted add-ins, or add-ins for short, are managed .NET Framework assemblies. Add-ins use the API elements exposed by the Microsoft.MediaCenter.AddIn and Microsoft.MediaCenter.Extensibility namespaces to control aspects of the Media Center experience, and to extend the capabilities of Media Center in some way. Add-ins also has access to other .NET Framework namespaces and external assemblies.

An add-in runs inside Media Center, either as a background process or on-demand. A background add-in is started when Media Center starts. If not closed explicitly, a background add-in runs as long as Media Center is running. An on-demand add-in needs at least one entry point in the Media Center GUI and is started manually by the user. MCE has 16 entry points. These can be summarized into four categories:

More Programs The add-in will appear under "More Programs" in the main menu.

More With This The add-in can be found by clicking the "More.." button in the context menu of a media file.

Services The add-in will appear in the "More" section of each experience (e.g. "More Music").

New For Me A "New For Me" menu will appear in the main menu if an add-in registered with this entry point has produced new information.

Update Rollup 2 for MCE 2005 introduced a separate process (ehExtHost.exe) for hosting add-ins. In prior versions add-ins run inside the Media Center

process (ehShell.exe) itself. It also enabled MCE to run assemblies compiled using .NET Framework 1.1, instead of just 1.0.

An add-in can implement all of the functionality it needs, or it can communicate with separate executables outside of Media Center to implement certain tasks. For example, an add-in could be used to manage a process for downloading or retrieving content while Media Center is running, or it could rely on a separate executable running in Windows to retrieve content even when Media Center is not running.

The only way an add-in can communicate with the user is through Media Center dialog boxes. For a more sophisticated communication, Media Center HTML application must be used to handle the user interaction.

Appendix C shows how to create a simple add-in and registering it with Media Center.

Media Center HTML applications

When creating extensions to Media Center, the GUI of such must be created in HTML. JavaScript and style sheets can be used, and Media Center parses the HTML pages in the same way as Internet Explorer. The SDK includes style sheets to create a GUI similar to that of Media Center, but of different colour. It does not include a style sheet in the Media Center blue style. However many versions of the blue style has been created and published by others than Microsoft. A good version can be found at [33].

MCE exposes a variety of objects that support HTML applications. We will not go into these. Detailed information can be found at [39].

Media state aggregation service (MSAS)

MSAS is a local Component Object Model (COM) server that receives media status information from Media Center and distributes it to sinks that are registered with the MSAS service. This service can used to retrieve information on the state of Media Center. For example it can be used to retrieve information on which channel a TV tuner is decoding.

Media Center Extenders

The Media Center Extenders connect to the Media Center using RDP. The RDP session runs in the context of a limited user account. This puts re-

strictions on what can be done on an extender, and thus puts restrictions on add-ins and HTML applications created to be used from an extender. The limited user account has the following user rights:

- Access this computer from the network
- Bypass traverse checking
- Log on locally
- Create global objects

An extender only supports input from the remote control, not a keyboard or mouse. To let the user enter text, you have to use either the triple-tap method or an on-screen keyboard. Triple-tap is when you click the same button one, two or three times to get the different letters associated with that button, just as typing a sms on a cell phone without dictionary.

An extender can not display user interface elements of programs not running within the Media Center processes (ehExtHost.exe and ehShell.exe). If one tries to display user interface element of programs not running within these two processes, the extender will display an error message. To prevent this, it is important to set such user interface elements not to be visible.

An extender session can write data to the file system of the MCE as long as the access control list (ACL) of the folder allows it. One should be careful not to set a too liberal ACL. Normally extenders can write to the shared folder ..\Documents and Settings\All Users\Documents. An extender session can also write to the registry of the MCE. Since the session runs in the context of a limited user account, it can not write settings to HKEY_LOCAL_MACHINE\SOFTWARE. It can, however, write to HKEY_CURRENT_USER\SOFTWARE.

6.2.2 JDOM

The JDOM library [10] is a library that supports easy XML manipulation in java. The library contains classes which can represent an XML document as a tree structure of objects. Classes for reading and writing such structures to and from XML files are also present in the library. Following is a brief description of all classes in the JDOM library used in this project.

Document The Document object represents an XML document. It contains methods for accessing the root element as well as document-level information.

Element The Element object represents an XML element, i.e. a tag in an XML document. It is possible to add and manipulate the content of this object. The content of this object can be either another Element or a pure textual content represented as a String.

SAXBuilder The SAXBuilder object builds a tree structure consisting of JDOM objects representing an XML document from for example a file, a stream or a URL.

XMLOutputter The XMLOutputter object outputs the JDOM XML document as a stream of bytes which can be for example written to a file using a FileWriter.

6.3 Other important technologies

The prototype uses many different technologies. In this section we look into three of these. First Ajax is explored, then we describe Java Remote Method Invocation (RMI) and finally Public Key Infrastructure and the X.509 standard.

Why we use Ajax and Java RMI is described in Section 6.1. The reason for looking into Public Key Infrastructure and the X.509 standard can be found in Section 5.7.

6.3.1 Ajax

This section will present the basics of Asynchronous JavaScript and XML (Ajax). First we will discuss what Ajax is and how it works. Next the request and response behaviour of Ajax is explored. In the following subsections first some important HTTP ready states and status codes are presented, and then the use of XML and DOM in Ajax is explained. The last subsection presents some security issues concerning Ajax. More information about Ajax can be found in the articles written by Brett McLaughlin at IBM [15] and the article written by Greg Murray at Sun Microsystems [19].

Ajax explained

Ajax makes web pages feel more like applications than normal web pages. The usual request, wait for answer, refresh web page cycle of normal web applications are replaced with a web site that sends a request and just repaints a part of the page when it gets the response. The web page itself is

still available to the user while the page waits for a response from the server. The server can be coded in any language, and the Ajax pages themselves may be created from server script languages such as JSP, PHP or ASP. All this is made possible through a collection of old technologies used together:

HTML Most of the Ajax web page is pure HTML along with CSS. This includes tables, forms etc.

JavaScript JavaScript is the core of Ajax. Both the request/response handling, as well as the code that makes the web page dynamic is pure JavaScript.

Dynamic HTML (DHTML) To make it possible to make the web page dynamic through JavaScript, DHTML is used. This is for example done through the use of div and span tags.

XML XML can be used to create more complex requests and responses to and from the server.

Document Object Model (DOM) The Document Object Model is used both to access XML that is returned from the server, as well as building and accessing HTML structures.

Request and response in Ajax

Request and response in Ajax are done through the `XMLHttpRequest` object. This object is created in different ways depending on the kind of web browser the user is in possession of. Obviously one can not know what browser the user is using, so the code has to support all three kinds of object creation. In Figure 6.1 object creation number 1 is used for non-Microsoft browsers, while object creation number 2 and 3 are used for creating the `XMLHttpRequest` object for different versions of Internet Explorer.

To send a request using the `XMLHttpRequest` object, one first has to open a connection to the server. This is done through the use of the `open()` method. This method takes three parameters: the type of request (`POST`, `GET`, `HEAD`), the URL of the server and a boolean parameter stating whether or not this connection is to be asynchronous. This last parameter is of course set to true in Ajax, since Ajax uses asynchronous communication. Once the connection is open, one need to specify which method is to be run when the client receives a response from the server. This is done by setting the `onreadystatechange` parameter of the `XMLHttpRequest` object to the method one wishes to invoke when receiving a reply. Finally to send the reply, one invokes the `send()` method of the `XMLHttpRequest` object.

```
/* #1 */
xmlHttp = new XMLHttpRequest();

/* #2 */
xmlHttp = new ActiveObject("Msxml2.XMLHTTP");

/* #3 */
xmlHttp = new ActiveObject("Microsoft.XMLHTTP");
```

Figure 6.1: Creation of the XMLHttpRequest object

The information to be send to the server is given as a parameter, or `null` if request type `GET` is used. An example of a simple `GET` request is shown in Figure 6.2.

```
/* Open a connection to the server */
xmlHttp.open("GET", "www.server.com/index.jsp?param=test", true);

/* Setup a function for the server to run when it's done */
xmlHttp.onreadystatechange = updatePage;

/* Send the request */S
xmlHttp.send(null);
```

Figure 6.2: Sending a simple request to the server

HTTP ready states and status codes in Ajax

Although all that is really needed to receive the reply from the server using Ajax is to specify the `onreadystatechange` parameter, the server will actually give up to five different responses to the client depending on the browser used to send the request. Each of these responses corresponds to HTTP ready states from 0 to 4. These ready states have the following meanings [15]:

- **0**: The request is uninitialized, that is before `open()` method has been invoked.

- **1**: The request is set up, but not send through the use of the `send()` method yet.
- **2**: The request is sent and is being processed.
- **3**: The request is in process, some information may now be available from the response.
- **4**: The request is complete.

In most Ajax applications one is normally most interested in the response with ready state 4. One does not usually wish for the method specified in the `onreadystatechange` parameter to be called on every response, so one implement checks to see what the ready state is in that method. The HTTP ready states can be accessed through the `readyState` parameter of the `XMLHttpRequest` object.

Even though the response returns with a HTTP ready state of 4, this does not imply that the request actually succeeded. For this reason, it is also important to check the HTTP status code of the response, which is accessible through the `status` parameter of the `XMLHttpRequest` object. A status code of 200 means that everything is OK. It could also be wise to check for different error codes of the 400-family to give the user more informative error messages if anything goes wrong.

An example of a function called `updatePage()` which is called when the client receives server response is shown in Figure 6.3. In that example, first there is a check to make sure the ready state is 4, then a check to see if the status code is 200. This code example also alerts the user if the URL is not found.

```
function updatePage() {
   if (xmlHttp.readyState == 4) {
      if (xmlHttp.status == 200) {
         /* Handle response */
      } else if (reques.status == 404) {
         alert("Requested URL is not found");
      }
   }
}
```

Figure 6.3: Receiving a response and checking ready states and status codes

XML and DOM in Ajax

The simplest form of communication between the server and the client in Ajax uses only a response in a pure text format. However, it is possible to structure the replies from the server in a better manner. The server can send the response as an XML document, and the client can specify that the reply is an XML document by accessing the `responseXML` parameter of the `XMLHttpRequest` object.

From the `responseXML` parameter, the user will get a Document Object Model (DOM) representation of the XML response. DOM XML is an object representation of an XML object, which is accessible through different methods and parameters. For example the content of the XML document shown in Figure 6.4 can be accessed as shown in Figure 6.5.

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
   <msg>
      This is an important message.
   </msg>
</root>
```

Figure 6.4: An example XML document

```
var doc = xmlHttp.responseXML;
var msg = doc.getElementsByTagName('msg').item(0).firstChild.data;
```

Figure 6.5: Receiving an XML file and retrieving a message from it

The code shown in Figure 6.5 first get hold of the DOM representation of the XML reply, then extracts the content "This is an important message." from the XML document and stores this content in a variable.

There also exist DOM representations of HTML documents. This makes it possible to create and modify HTML documents through JavaScript. It is by the use of this technology that content and appearances of web pages written in Ajax can be made dynamic.

Security in Ajax

Like all applications using JavaScript, Ajax is subject to the sandbox security model. This means that Ajax code can not access files locally stored on the user's computer, nor can it access files on other domains than the one the Ajax code is located on. This also means that the server side code, which the Ajax application makes use of, must be located within the same domain as the Ajax code.

However, Ajax applications are really just a fancy way of filling out forms, so in theory any user may view the source code and fabricate similar requests to the server. This means that in some way one need to authenticate and validate requests. We solve this problem in our project by using PKI technology. More information about this can be found in Section 6.3.3.

6.3.2 Java Remote Method Invocation

Java Remote Method Invocation (RMI) is a Java based middleware used to allow objects to invoke methods on other remote objects. These objects are typically used by other programs on other computers, and Java RMI is a way to make Java programs communicate through method invocation instead of socket programming.

As can be seen from Figure 6.6, Java RMI has three different layers. The first layer consists of stubs and skeletons. A stub of a remote object is a representation of the remote object at the client side. When a client object makes a remote method invocation, this invocation is received at the stub of the remote object. The stub marshals the parameters of the method call and sends the request through the network using the second layer, the Remote Reference Layer (RRL). The RRL manages the location of the different remote objects and also handles multi-threading and garbage collection used by the remote objects. The third layer of the Java RMI system is the network layer where the transmission of the remote method invocation occurs. At the server side, the skeleton of the remote object receives the request from the network. The skeleton then unmarshals the parameters and performs the actual method invocation on the remote object. When the method invocation returns, the answer is sent back to the stub through the network. When sending a request, the stub waits for a reply and finally returns this to the invoking object at the client side [21].

To create a remote object using Java RMI, one must first create an interface which extends Remote. All methods that are going to be accessible remotely must be defined in this interface. When implementing the remote object, this

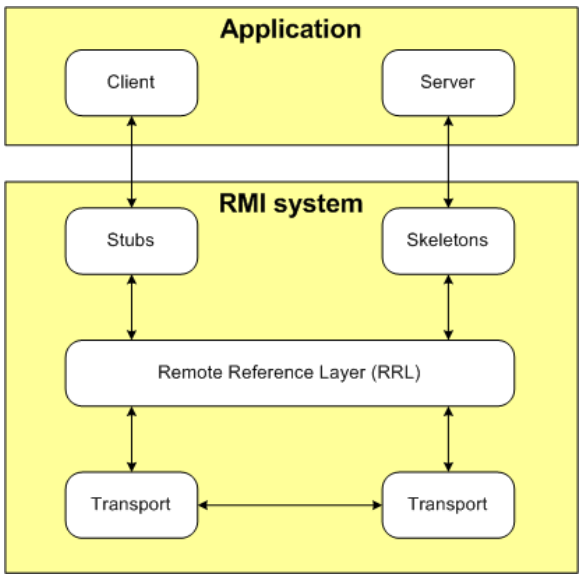

Figure 6.6: RMI system model

object must inherit the earlier created remote interface as well as extend UnicastRemoteObject. It is also important to know that all objects which are used as input to or return from the remote methods must be serializable. Once the remote objects are created and compiled, one need to use the "rmic" application to create the stubs and the skeletons of the remote objects. When starting the server program, the remote objects should be registered with a Java RMI registry using one of the Naming.bind() or Naming.rebind() methods. On the client side, a reference to the remote object is obtained from the same Java RMI registry using the Naming.lookup() method.

For more in-depth information about Java RMI, see [9].

6.3.3 Public Key Infrastructure and the X.509 standard

The purpose of this section is to give a brief overview of public key cryptography, Public Key Infrastructure (PKI) and the X.509 certificate standard.

Public key cryptography makes use of a pair of keys, one private key and one public key, for encrypting messages. To send a secure message from A to B, A encrypts the message with the public key of B (which is publicly available). The encrypted message is sent to B, where it is decrypted with the public key of B (which only is available to B). It is also possible to use public key cryptography to authenticate messages. This is also known as

digital signatures. If A wants to digitally sign a message, he encrypts the message with his or her private key before sending it. At the receiving end, B can decrypt the message using A's public key. Anyone can decrypt the message using A's public key, but only messages encrypted with A's private key can be decrypted using A's public key. Thus B know that the originator of the message was A.

Although public key cryptography can be used to both ensure data confidentiality and authenticity of messages, it leaves the problem of key distribution quite open. For example, anyone can get access to A's public key, but there is no way of knowing if that key really belongs to A. This is where PKI comes in. The essence of PKI is the use of public key certificates. In general, a public key certificate consists of a user ID along with the user's public key. The certificate including the private key is then hashed using some hashing algorithm and then encrypted with the private key of a Certificate Authority (CA). The CA is an institution (special security companies, government agencies etc) which is trusted by all the users. The users must obtain the public key of the CA in a secure fashion, for instance by getting a physical copy on a disk directly from the CA. Any user can then use the public key of the CA to verify that the public key and the user ID in a certificate has not been altered after being signed by the CA. Figure 6.7 shows how PKI in general works.

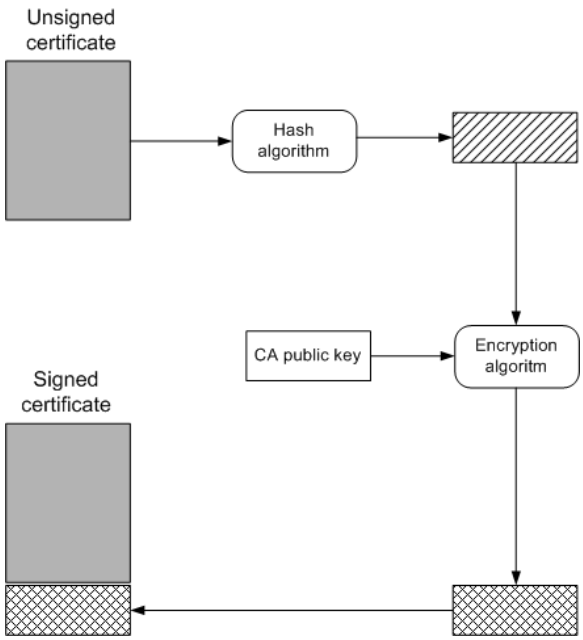

Figure 6.7: Public Key Infrastructure

The X.509 standard is a universally accepted standard for formatting public key certificates. The fields in a X.509 v3 certificate are shown below:

Version: The version of the used X.509 format.

Serial number: A unique serial number within the CA identifying the certificate.

Algorithm ID: The algorithm used to sign the certificate.

Issuer: Information identifying the issuer, i.e. the CA, of the certificate.

Validity: The date and time this certificate starts to be valid and end to be valid.

Subject: Information identifying the subject of this certificate.

Subject public key info: The public key of the user as well as the name of the algorithm with which this key is to be used.

Issuer unique identifier (optional): An optional bit string field uniquely identifying the CA.

Subject unique identifier (optional): An optional bit string field uniquely identifying the subject.

Extensions (optional): To stop making new versions of the X.509 format every time a new field is needed, this extensions field was created. Different extension fields may be optionally added here.

Signature: The hashed version of the certificate, encrypted with the CA's private key, along with the encryption algorithm used to create the signature.

In addition to defining the format of the certificates, the X.590 standard also suggests which encryption and hash algorithms to use, and describes how to obtain user certificates using certificate chains, authentication procedures and procedures for revocation of certificates among other things. These subjects, however, will not be further explored here.

For more information about public key cryptography, PKI and X.509, see [31].

6.4 Code standards

Having standards for writing source code and commenting source code is important. The main purpose of having such standard is to make the source code more readable. This section describes the standards used for writing and commenting the source code.

6.4.1 Writing source code

The main languages used in our system are Java and C#. The client software is written using C# and the server uses Java.

For all code written in Java, we follow the "Sun's Code Convention for the Java Programming Language" [32]. To make the source code of the server and the client software as similar as possible, we also follow this standard as tight as possible for the C# code.

6.4.2 Commenting source code

Java code documentation is performed follow Sun's "How to Write Doc Comments for the Javadoc Tool" [6]. The code is further commented to make it easier to understand the different algorithms using the standard defined in [1].

The C# code is documented using the standard automatically applied by Visual Studio. In addition, we comment the algorithms in the same manner as with source code written in Java.

Chapter 7

Results of the prototype

This chapter presents the results of the development of the prototype. First the prototype is illustrated through screenshots, and explanations of the different parts of the prototype are given. Hopefully this will give a better overview and understanding of the system. The prototype is based on Windows Media Center, which is a fairly new platform. Thus developing for Windows Media Center is a relatively new skill. Because of this we will share some of the experiences we have had during the development of the prototype.

7.1 The prototype in pictures

This section presents a picture of the prototype overview and screenshots of the different parts of the prototype.

Figure 7.1: Prototype overview

Figure 7.1 shows the prototype overview. At the center of the picture we have the MCE PC with Media Center running. The Media Center holds a connection to the server, shown at the right of the picture. At the left of the picture we have a laptop. The laptop uses remote access to the MCE PC. This is done by connecting to a web site at the server.

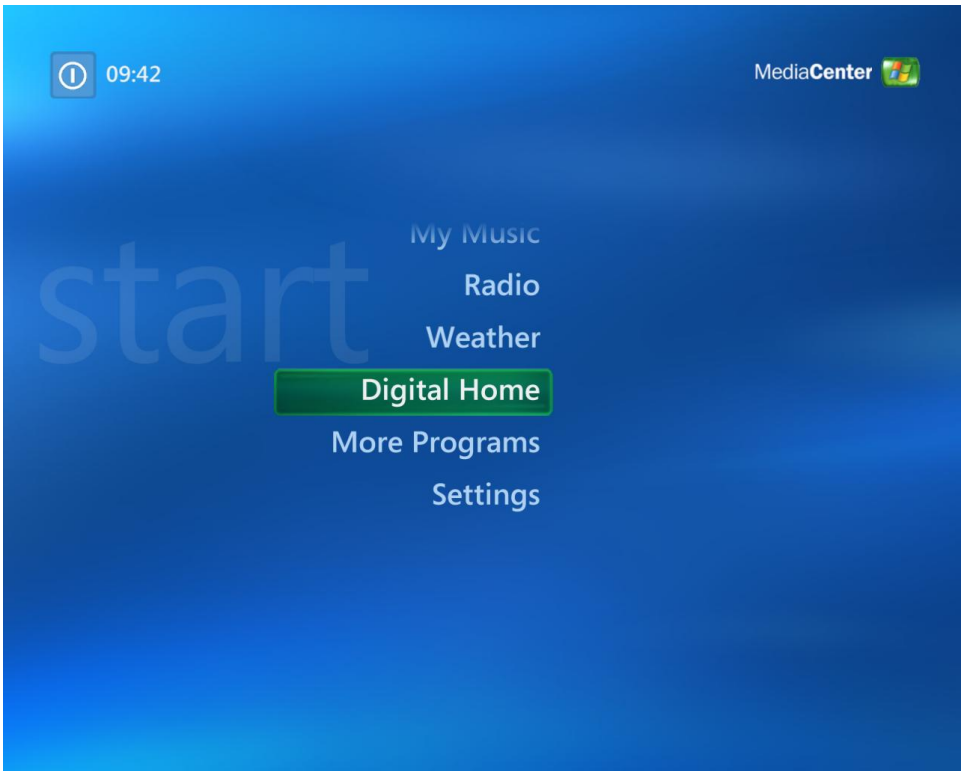

Figure 7.2: MCE: Media Center main menu

The Media Center main menu is shown in Figure 7.2. It contains a menu item "Digital Home". This is the menu item we have added, and it appears when the client software is installed. The "Digital Home" menu item gives access to the service menu, shown in Figure 7.3. The service menu is created based on the services the customer subscribes to. In the figure you can see that this customer does not subscribe to "Service 2".

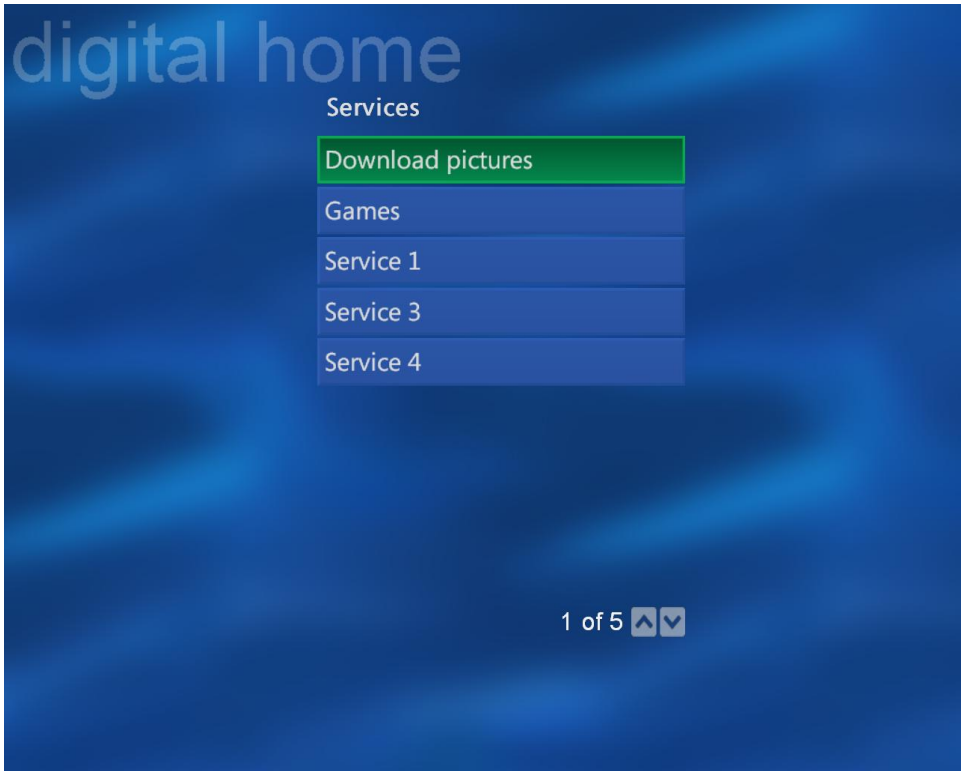

Figure 7.3: MCE: Digital Home service menu

Figure 7.4 shows one of our example services, picture download from the Media Center. At the left there is a list of pictures available for download. A thumbnail of the picture selected is shown to the right. By clicking on picture name in the menu, it is downloaded and stored in the "My Pictures" folder. Beneath the thumbnail, you find a button which brings you to the Media Center's "My Pictures" service.

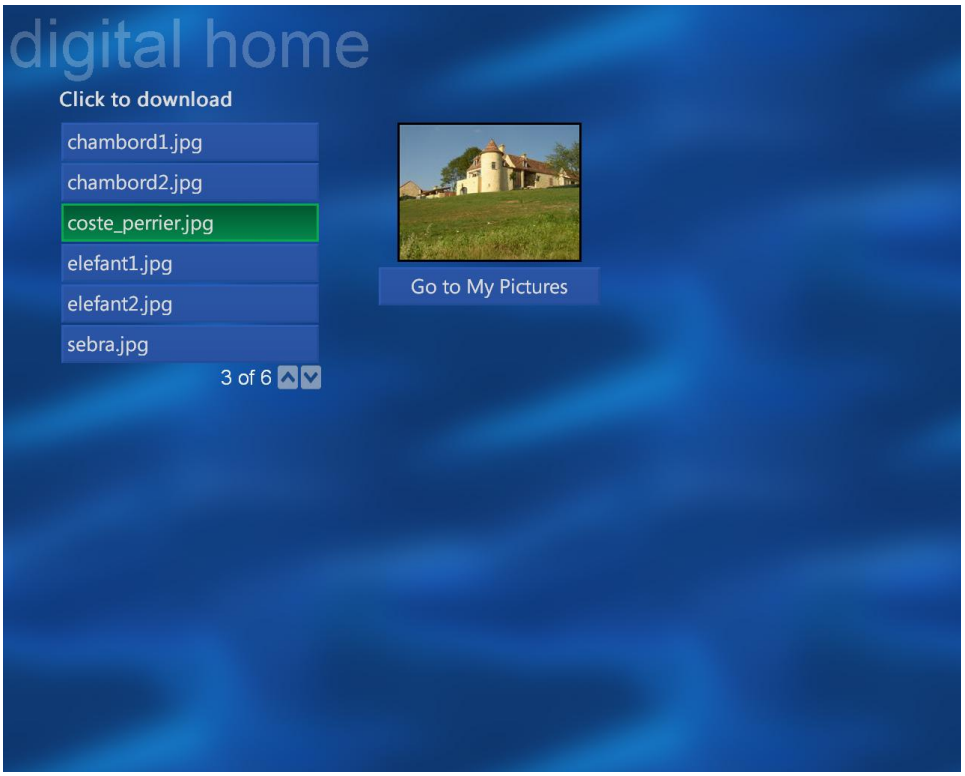

Figure 7.4: MCE: Picture download service

The remote access web page, showing the picture download service, is shown in Figure 7.5. To the left, there is a menu consisting of the different services the customer subscribes to. At the top right corner the status of the customer's Media Center is shown. The center of the picture displays the selected service. This picture download service is the same service as found from the Media Center. Only the GUI is modified to fit the remote access web page. By selecting a picture and clicking "Download picture", the picture is sent to the customer's MCE PC.

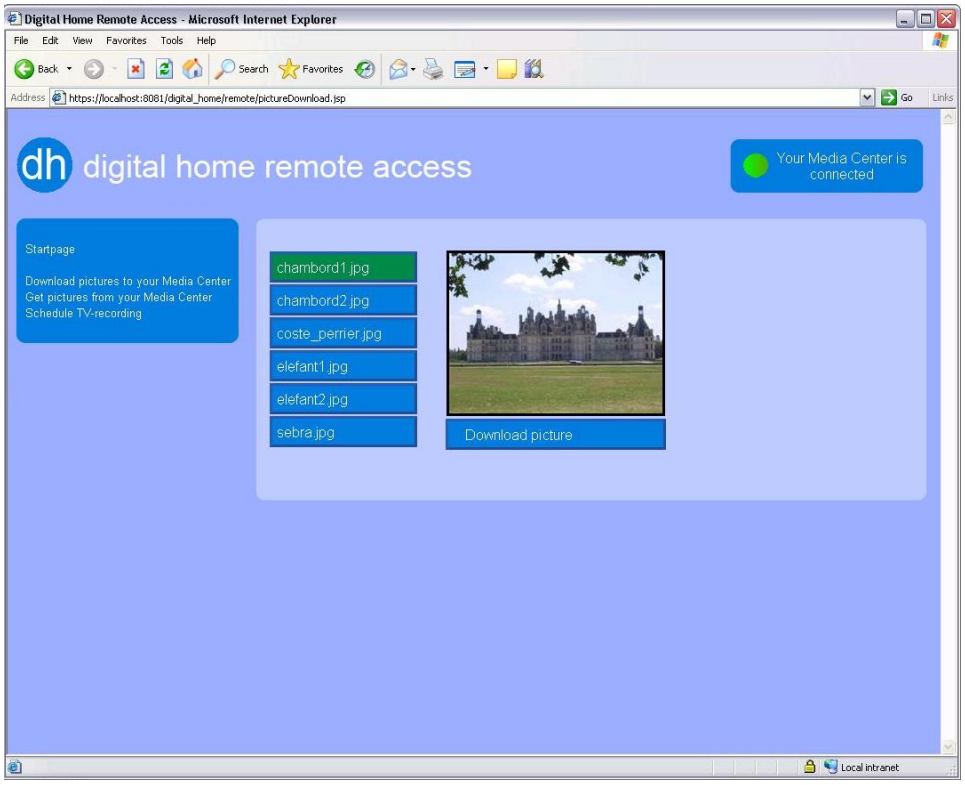

Figure 7.5: Remote: Picture download service

Figure 7.6 shows the picture upload service. A list of the pictures in the "My Pictures" folder of the customer's MCE PC is retrieved and displayed at the left of the service window. The customer selects which picture he wishes to retrieve and clicks the "Get <picture name>" button. The selected picture is then uploaded to the server and displayed in the web page. The figure shows the web page after "coste_perrier.jpg" has been retrieved.

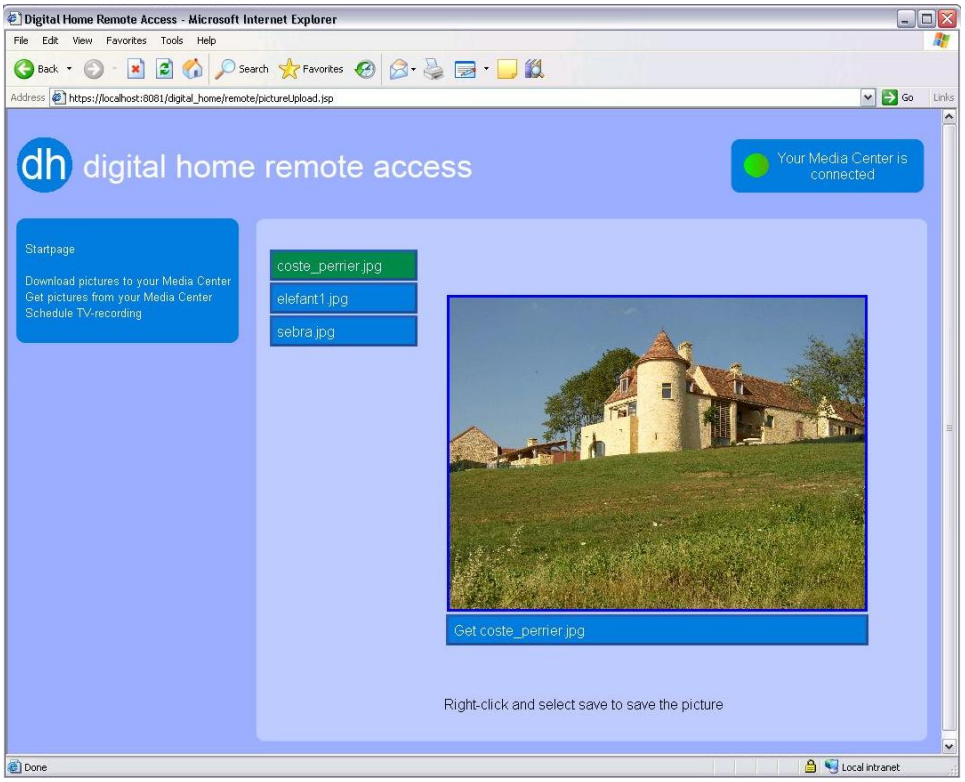

Figure 7.6: Remote: Picture upload service

Figure 7.7 shows the last of the remote access example services. This service lets the customer remotely schedule TV-recordings on his Media Center. Such a service should provide an EPG of the channels the customer subscribes to. However, for a quick demonstration we have provided two programs we know run regularly, the news and MacGyver. The news can either be selected to be recorded once or as a series every day. The text on the buttons of the available programs is in Norwegian.

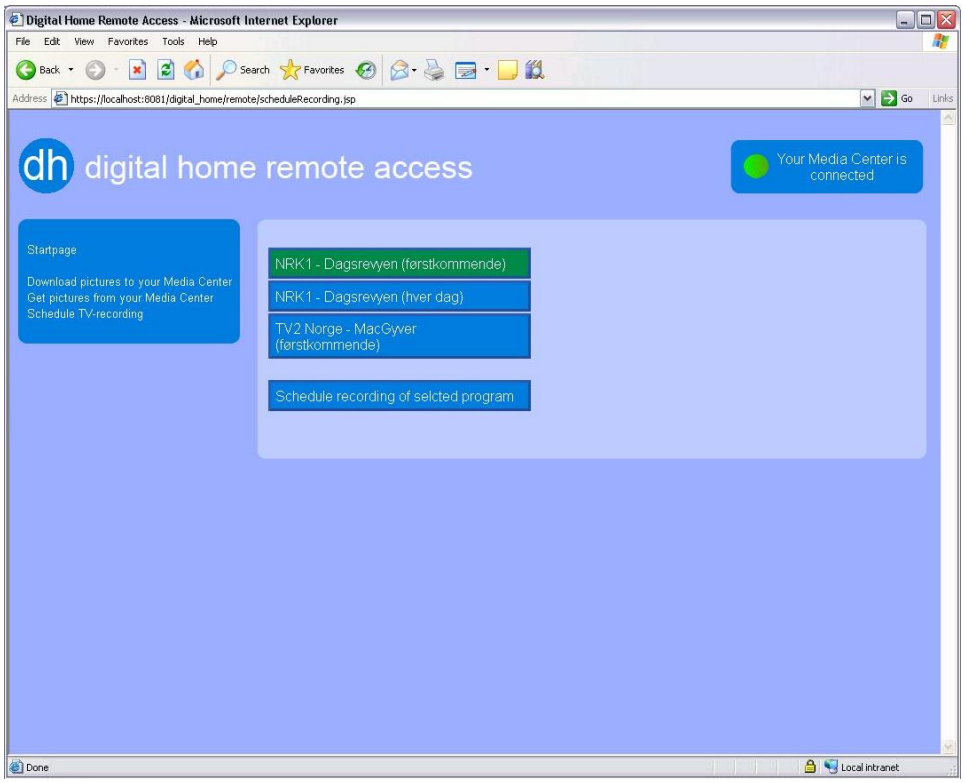

Figure 7.7: Remote: Schedule TV-recording

If the customer's MCE PC is not connected to the server, most services will be unavailable. This is shown in Figure 7.8. At the top right corner the status of the Media Center has now shifted to "disconnected" and the light has shifted to red.

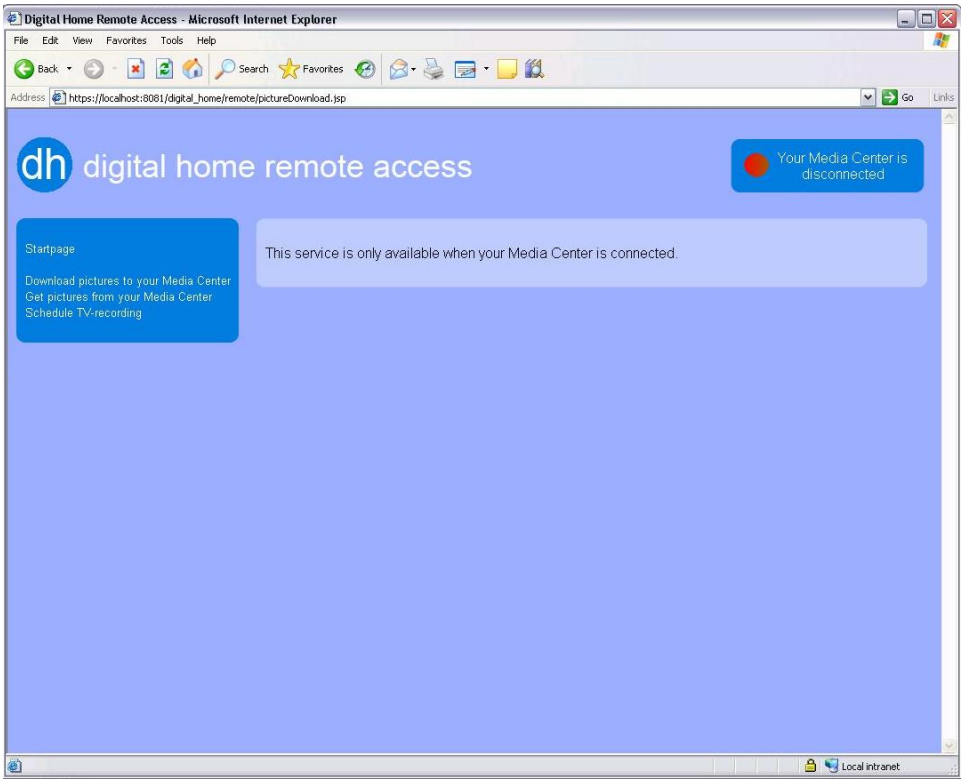

Figure 7.8: Remote: Media Center not connected to server

Figure 7.9 shows the server software. It contains a list of non-authed and authed connections. It also has a list of active download connections. At the bottom half log messages are displayed. From the "Server" menu it is possible to start the distribution of updated client software. This is not shown in the figure.

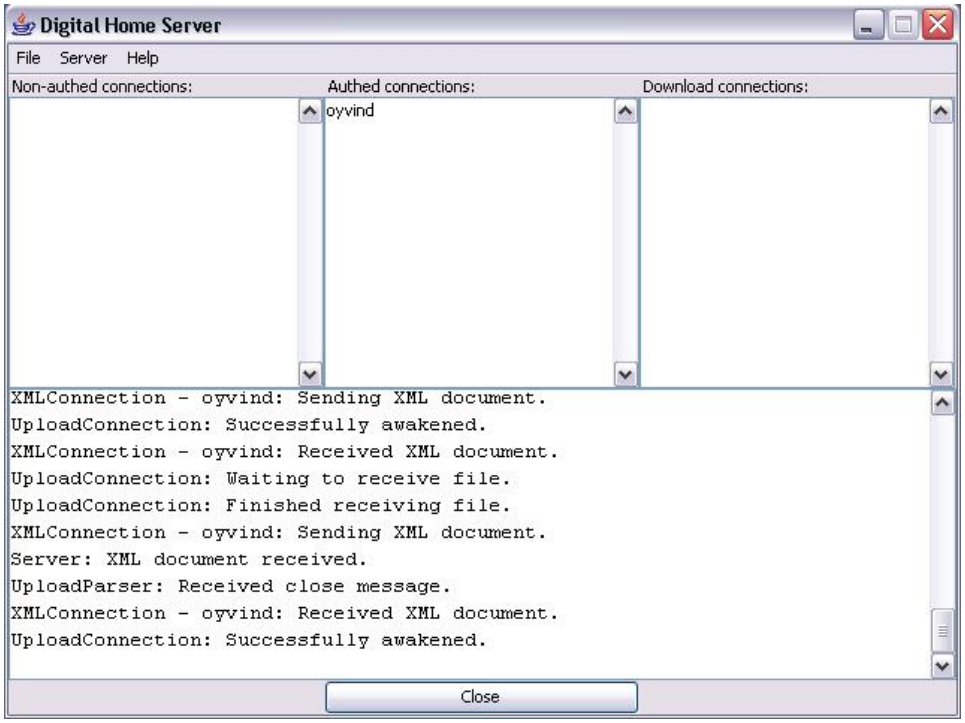

Figure 7.9: Server software

7.2 Experiences

Developing for Windows Media Center is not too different from normal application/web development. However, during the development of our prototype, we came across some issues and made experiences. We share these experiences to help future Media Center developers.

Media Center share Internet Explorer's Internet options. Adding a certificate in Internet Explorer will make it available to Media Center. The same goes for cookies and other Internet options. Media Center also shares favourites with Internet Explorer. To see this in action, check out the MCE Web Browser [14], an add-in created by Anthony Park.

Microsoft uses HTML and DirectX for the GUI of Media Center. The use of DirectX is not available to add-ins. For this, Media Center uses a modified version of Internet Explorer. One can still create add-ins which looks almost identical to the rest of the Media Center GUI, but Microsoft has not included a blue style for this. People in the Media Center community has created blue styles them selves. A good one can be found at The Green Button [33]. Media Center does not support everything Internet Explorer does, so be careful when creating the GUI of your add-in.

When creating add-ins, especially background add-ins, it can be difficult to see what happens. Thus it can be difficult to debug. A great way to work around this is to first create your add-in as a Windows application. By having a simple GUI, it is easy to get good debug information. It is actually very easy to convert a Windows application to a Media Center add-in. In stead of a main-method, you need an Initialize-, Uninitialize- and Launch-method. The Initialize-method can be compared to a constructor. The Uninitialize-method is run when the add-in closes and the Launch-method is the add-ins main-method. In addition, the class containing these methods must inherit MarshalByRefObject, IAddInModule and IAddInEntryPoint. We created two projects in Visual Studio. The only difference between them was the main class. To create an add-in of the Windows application, all files except the main class was copied into the project containing the add-in main class.

To install an add-in the user must have administrator privileges of his computer. The add-in must be put into the GAC [5], and this can only be done by an administrator. Our prototype updates the client software without involving the user. If the user is not logged in as an administrator, the update will be blocked. The correct way of dealing with this is to tell the user to log in with his Windows administrator account and start his Media Center for the latest update to be installed. This makes the update less transparent to the user. Thus with this solution updates should be as infrequent as possible.

For simplicity of the prototype the user on our MCE PC has administrator privileges.

Part IV

Related work and conclusions

Chapter 8

Conclusions

This chapter concludes our work on this project. First related work is described, then our conclusions are made. In the end of the chapter we present the most important future work which we feel is needed to expand the prototype into a commercial system.

8.1 Related work

As far as our research has taken us, it does not seem like there are anyone else working on a similar project. Our project is a highly commercial one. This means that if someone else is developing a similar system, chances are that they will not reveal any information about it until the project is completed. For this reason it is hard to find any information about related work on this subject.

The closest thing to related projects we have come across is the mDisk project at Fast. This project is described in Section 8.1.1. There are people developing other kinds of Media Center add-ins, and we have learned a lot from many different Media Center development community sites. A description of some of these sites are given in Section 8.1.2. There exist broadband TV service systems. One example is briefly presented in Section 8.1.3.

8.1.1 mDisk

Fast [4] is working on a product called mDisk. Among other, it addresses remote access of media files and can be used with Media Center. This makes it partially related to our work. The description of mDisk given here is based

on personal contact with Fast.

mDisk offers each user some amount of storage capacity. A user will choose which files or folders he wants to be able to access remotely. All files selected for remote access are uploaded to the mDisk server. At this time, the user can delete the local file if he pleases. Thus, the mDisk server acts as network storage. Files can be uploaded and accessed from different devices, such as any web browser, Windows Media Center and mobile phones.

mDisk has a lot of features in addition to access to files. It discovers which artists the user likes and offers him to buy more music with these artists. It includes an instant messaging service. The user can share a selection of his files with his friends. It shows the latest news, has an activity/calendar functionality, enables blogging and much more.

Fast are creating a system rich on functionality. It has great support of remote access to multimedia, but no ability for a user to control his Media Center remotely. A system similar to mDisk, including Media Center control functionality, would be a really interesting remote access system.

8.1.2 Blogs and communities

There are a lot of people working on different Media Center add-ins. This has lead to the forming of some Internet communities. Also a lot of people are writing their own blog on the subject. Most add-ins discussed, and created, are nice little extra features. Some of them do, however, touch areas applicable to our work. Following the threads on the communities' forums and reading blogs concerning Media Center is great for picking up hints and tips.

The main Windows Media Center community is The Green Button [33]. In addition to news and blogs, this site contains forums for Media Center hardware, using Media Center and developing for Media Center. During autumn 2005 a new community called RemotelyCool was created. Because of The Green Button's leading role, RemotelyCool was shut down in spring 2006.

There are a lot of blogs concerning Windows Media Center and Home Theatre Solutions. We would like to mention two of these. The first one is Eirik Solheim's blog [28] on Home Theatre Solutions. The second is Retrosight [24], Charlie Owen's blog on Windows Media Center development.

8.1.3 Broadband TV service

There exist broadband TV service systems. These systems have a mix of multimedia functionality and service provider functionality, covering only parts of both areas. One such system is offered by Nord-Trøndelag Elektrisitetsverk (NTE) [20]. This solution uses a decoder for each TV. The decoder is a thin client connecting to NTE's server when booted. When connected, the user has access to a selection of TV and radio channels and can browse an EPG, and it is possible to buy movies. Other functionality such as weather forecast and the latest news are available.

This kind of solution can not offer home automation or remote access. The entire service is dependent on the user's internet connection, and it has no support for playing local media files. Because of these limitations, it is not an alternative for a digital home by it self. If a service provider is not interested in providing remote access, and is not concerned with a complete digital home, a thin client solution such as NTE's solution could be a good choice.

8.2 Conclusions

The purpose of this project was to find an architecture that lets a customer access services from a service provider both on his Media Center PC as well through remote access. The architecture had to enable the service provider to easily update and change the range of services, involving the customer as little as possible. Earlier we defined the following three goals for the project:

G1 - Find a base for the architecture: The first goal is to find a base for the architecture. This could either be an existing digital home system or it could be developed from scratch.

G2 - Find a service architecture: The second goal is to find the service architecture. This architecture must support both services at the customer's digital home system as well as remote access to services.

G3 - Develop a prototype of the architecture: The third goal is to develop a prototype which demonstrates the system.

Chapter 2 and 3 described the ideal digital home as well as existing digital home solutions. Both centralized and distributed solution architectures were considered for the ideal digital home, and a centralized architecture was chosen. This choice was based on the fact that we consider price and the ease of installation to be the most important aspects for such a system to be

commercially successful. From the myriad of available digital home systems, we chose Microsoft Windows Media Center Edition (MCE) to be the base of our system. MCE was chosen mainly because it is the solution that resembles the ideal digital home most in functionality. Furthermore, MCE is easily extensible through add-ins, both pre-made and self-developed. Since the focus of this project is to find a service provider architecture and not to develop a whole digital home, we found it more feasible to base our system on MCE than to develop everything from scratch.

We looked at two different main service architectures, named add-in solution and web solution. The main difference between these architectures is how the user-invoked services are accessed from the customer's Media Center. In the add-in solution all access to services goes through add-ins, and the services have to be installed on the customer's MCE. In the web solution, the services are web pages located on the service provider's web server. Thus the part of the system installed on the customer's MCE is much smaller in this case. The most important aspect of the final architecture is that it should be easy to add, remove and update services, involving the end user as little as possible. We found that the web solution was superior to the add-in solution when it comes to updates. In the web solution, most updates of services can be done at the server alone. Thus the web solution architecture was chosen.

We developed a prototype of the chosen architecture, demonstrating the main principals of the architecture through some simple example services. The prototype consists of three different parts: the client software, the server software and the service web pages. The client software was developed as add-ins to MCE and was for this reason developed in C#. The server software was developed in Java to make the server platform independent. Because of this technological choice, the service web pages were developed in Java Server Pages. Java RMI was used for communication between the web pages and the server software. Public Key Infrastructure and X.509 certificates were used to make the service pages secure as well as a way to identify connecting customers. The service web pages were made to feel more dynamic and more like applications than normal web pages through the use of Ajax.

In addition to accessing services through the customer's Media Center, our architecture also supports remote access to services. These services includes both special remote services as well as some of the user-invoked services. Control of the customer's Media Center can be implemented as remote services. To demonstrate remote access to the customer's Media Center, we developed a service for upload of pictures from the customer's MCE PC to the remote access web page and a service for scheduling recordings of TV shows at the customer's Media Center.

As described in Section 3.3.1, there only exist hardware extenders to MCE today. If the GUI of the remote access web pages where made to look like the Media Center GUI, and services for streaming audio, video and live TV were to be implemented, our remote access system could be used as an almost fully functional software extender. As a difference from the normal extenders, this software extender would not be limited to be in the same LAN as the MCE PC. This software extender could be used from any computer in the world connected to the Internet.

The ideal digital home can be divided into four main areas; home theatre functionality, home automation, service provider functionality and other functionality. MCE covers home theatre functionality well. Home automation and some of the other functionality can be gained through add-ins. All the wanted functionality does not exist, but it should be possible to create it. With our solution we add service provider functionality. As we see, the ideal digital home might not be too far away.

As of today, if a service provider wants to change the services offered to their customers, the flash memory of the Set-Top Boxes (STB) are updated through satelite. For larger updates, the whole STB could have to be replaced. When providing services through a Media Center instead, it is enough to change some settings at the service provider's server and possibly update the client software at the customer's PCs to make changes to the services offered. This constitutes fewer updates at the client side as well as no major hardware replacements. Since STBs are proprietary products developing updates for them might not be as easy as developing for Media Center.

Our prototype has proven that the architecture we proposed meets our goals. Using this architecture a service provider can easily update and administrate their services. Furthermore our architecture lets a service provider offer their customers remote access to, and control of, their Media Center. Updates require minimal involvement of the customers. Only when the client software is updated will the customers be involved, and then they only have to restart the Media Center software for the changes to take effect.

8.3 Future work

This section summarizes what changes we feel should be done to the prototype to make it into a commercial system. These changes include changes to the existing prototype source code to make it more resilient, the inclusion of some key services and the integration of the STB into the MCE PC.

The prototype demonstrates the key features of our architecture. Some aspects, not vital to the key functionality of the prototype, have been implemented in a rather naive fashion. A commercial version of the prototype should be able to handle disconnects in a better way than the current prototype, especially for the upload and download connections. Furthermore, the customer should get more information through pop-up boxes when connection errors occur. The file transfer algorithm used in the prototype does not support the resumption of partial file transfers and does not have a check to see if the complete file was received. This must be improved. Lastly, some security issues must be improved. First of all, the server-client communication needs to use a secure socket. The authentication of connecting clients at the server side should be done by verifying that the username and password sent by the client matches the ones found in the user database. This also makes it necessary to store the username and password of the client in a secure manner on the client's computer. As of now, the login to the remote access page only identifies the customer through the common name in the certificate. Thus anyone using a computer containing that certificate could access the customer's Media Center through the remote access web pages. To make the remote access web pages more secure, there should be necessary for the customer to enter a username and password to be able to enter the site.

Some essential services should be implemented for the system to be commercially interesting. First most customers would like to be able to change his or her subscription data, both through the customer's Media Center as well as through the remote access web pages. Secondly, to make the remote services more complete and closer to a software extender, the streaming of audio, video and live TV from the customer's Media Center should be accessible at the remote access web pages. To make the remote access system into a software extender, there should also exist a version of the remote access web pages using the GUI of the Media Center.

The last area of work we find interesting for the expansion of our system is to try to include the STB into the MCE PC. As of today, MCE does not support satellite TV tuner cards. However, if this were to change, it would be possible to include the STB functionality into the MCE PC and to make the STB obsolete. In addition to a satellite TV tuner card, a smart card reader and software for decrypting the encrypted TV signals would also be needed.

Part V

Appendix

Appendix A

Problem description

This master thesis is based on the report "Digital Home - Home Theatre Solution" written during the autumn of 2005. The report suggests a centralized architecture for a digital home, which uses a "nerve center" as the control unit, focusing on the client side of a digital home. This master thesis will focus on a service provider's role.

The goal of this report is to find an architecture of how a service provider can offer services through the customers' nerve centers. The architecture must be of such a manner as to enable the service provider to easily administrate and update the services offered. These updates should be as transparent as possible to the customers. Ideally no involvement of the customer should be necessary.

The architecture shall also provide the customers with remote access to their nerve center, giving them access to a selection of remote services. These remote services could be such as remote control of the customer's nerve center or streaming of media from the nerve center.

The architecture must take security into consideration. This applies both to the communication between the service provider and the customers' nerve centers and to protection of copyrighted material.

It might be feasible to extend an already existing digital home system. In this case, it is important that the system is as forward-looking and complete as possible.

It is desirable that the students create a prototype of the chosen architecture.

Appendix B

Digital Rights Management

In the last few years defending the rights of the owners of digital media has become an important issue. With the development of different peer-to-peer file sharing technologies like Napster, Gnutella and DirectConnect, the illegal sharing and downloading of digital media has become widespread. As a result to this trend, Digital Rights Management (DRM) has become an important aspect when developing applications that handles digital media. Since services which include some handling of digital audio and/or video are typical services which could be implemented for our system in the future, DRM is naturally also an important aspect in this project.

MCE uses Windows Media Player to play media files. This media player has built-in support for Windows Media DRM (WMDRM), so it is natural that this technology is used to ensure DRM in our system. WMDRM needs to be used on all media files on the system, but is considered most important when creating different types of multimedia-on-demand services for a service provider. The rest of this section will be used to explain how WMDRM works and display different kinds of scenarios on how licences in WMDRM may be used.

WMDRM works by encrypting media files by a key, which the user will receive when acquiring a license to play the file. For this to work, only media players which incorporates Windows Media Rights Manager are able to play files protected by WMDRM. Figure B.1 shows how this is done [36]. First the digital media goes through a packaging process. Here the media files are encrypted with a key, which is again stored with an encrypted license. Other information can also be added to the media file, like for instance a url with the location of the license. The protected media file can then be distributed to the user, either as a whole file by letting the user download it, get it from a CD etc, or by streaming the media file. When the user tries

to playback the media file or stream, he or she will either be routed to a web page to acquire the license or the media player may acquire the license automatically. When the license is acquired, the media file or stream may be played using the media player which features a Windows Media Rights Manager. The media file must be played according to the information in the license, and licenses may not be transferred to another system.

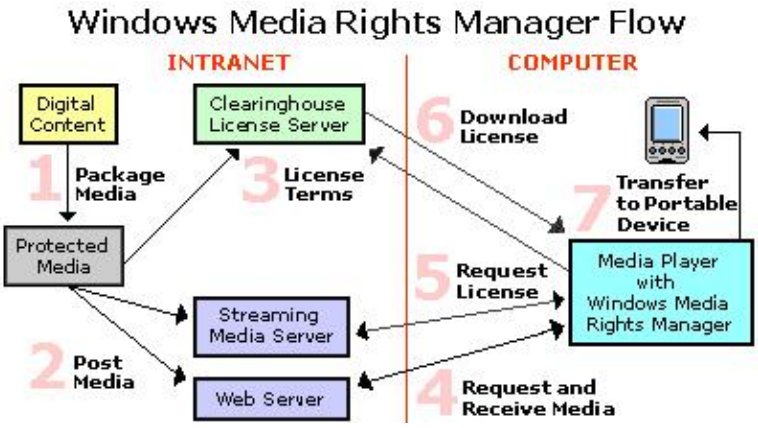

Figure B.1: Windows Media Digital Rights Management

When creating a DRM solution for media content from a service provider, different licensing schemes might be of interest. WMDRM features different licensing schemes suitable for a service provider [37]. For rental purposes and different kinds of multimedia-on-demand services, the license can include information on when the user can start playing the file and the expiration date of the file. It could also be the case that a service provider might want to broadcast a file on their television network or a user wants to transfer a file to a portable device. This is made possible by stating which devices should be able to playback the file in the license. Before purchasing digital media, some users might prefer to preview the chosen media first. The license of media files may include the number of times the file may be played. To create a preview version of a media file, the file can for instance be licensed so that it only can be played 1-3 times. It is also possible to offer different licensing schemes to the same media file. For example, one user might want to buy the right to watch a movie for a week. Another user might want to buy the right to watch the same movie forever and perhaps also have the possibility to transfer the file to a portable device. Both of these preferences can be met by offering different licensing schemes for the same file. Subscription services are another type of possible licensing schemes. Here the user might for example subscribe to different media files by paying a monthly fee. In this way, the user will get a license to play the media files covered by the

subscription for that month.

It is possible to acquire license in two different ways: indirect and direct license a acquisition. With direct license acquisition the user or the device trying to play the media files connects to the Internet and receives the licences directly. This makes it possible for mobile devices such as PDAs which have an Internet connection to acquire licenses on their own. Indirect license acquisition is first and foremost used with portable devices. This works by letting the user transfer media content and licenses from their computer to the portable device. For this to work it must be specified in the licence that the licence can be transferred to the portable device.

Appendix C

Creating and registering a Media Center add-in

An add-in can be created in any .NET language. This example shows a HelloWorld add-in created using C#. This add-in does nothing more than displaying a dialog box in Media Center.

This example is a summary of Microsoft's information on how to create an add-in. For the full version, please see [41].

C.1 Writing an add-in

To be able to write an add-in, you must have the MCE SDK installed. You should also install the Media Center Rollup 2 update. In addition the .NET Framework must be installed. To get hold of the MediaCenter name space, %windir%\eHome\Microsoft.MediaCenter.dll must be added to the references of your project.

The code for a simple HelloWorld add-in is as follows:

```
using System;
using System.Collections;
using Microsoft.MediaCenter.AddIn;

namespace MediaCenter.Sample.AddIn
{
   public class HelloWorldAddIn : MarshalByRefObject,
```

```
    IAddInModule, IAddInEntryPoint
   {
       void  IAddInModule.Initialize(IDictionary dictAppInfo,
        IDictionary dictEntryPoint)
       {
           // Write any initialization code here.
       }
       void IAddInModule.Uninitialize()
       {
           // Write any clean up code here.
       }

       void IAddInEntryPoint.Launch(AddInHost host)
       {
           host.HostControl.Dialog("Hello World", "My First Add-In",
                                   1, 0, false);
       }
   }
}
```

All Media Center add-ins must implement two interfaces: IAddInModule and IAddInEntryPoint. These interfaces expose methods that are called by Media Center. The add-in must also inherit from MarshalByRefObject since Media Center uses .NET Remoting to communicate with the add-in.

The interfaces the add-in must implement, includes three methods:

IAddInModule.Initialize is the first method that Media Center calls when it loads the add-in, giving the add-in an opportunity to initialize internal variables and obtain any system resources that it needs.

IAddInModule.Uninitialize is called before terminating an add-in. Media Center then briefly waits for the add-in to save its state, terminate any threads it created, and free any other system resources that it may have allocated. If the add-in's Uninitialize method does not return quickly enough, Media Center unloads the add-in's application domain, terminating the add-in.

IAddInEntryPoint.Launch The AddInHost object is guaranteed to be valid only until the Launch method returns. Therefore an add-in must make all calls to the Media Center API within the context of the Launch method. Attempting to call the AddInHost object after the Launch method returns can result in a fatal error. Multiple threads

spawned by the Launch method must be terminated before the Launch method returns.

C.2 Creating a strong-name

Media Center loads only locally stored add-ins that reside in the %windir%\ehome folder, and those that are installed in the global assembly cache (GAC). For security purposes, the GAC requires that all .NET assemblies, including add-ins, have strong names. This means that Media Center cannot load and run any add-in that is not a strongly named assembly.

A strong name consists of the following:

- The add-in assembly identity
  - Simple text name
  - Version number
  - Optional cultural information
- Pair of cryptographic keys used to digitally sign the assembly.

To create the cryptographic keys for your add-in, use the Strong Name tool (sn.exe) that is included with the Microsoft .NET Framework SDK. This tool creates the keys and stores them in a file with the specified name.

You add the strong-name information to your add-in's assembly by setting the various assembly attributes in the source code for your add-in. This is done in the AssemblyInfo.cs file that Visual Studio automatically generates for your project. An example of the AssemblyInfo.cs is as follows:

```
[assembly: AssemblyTitle("MyAddIn")]
[assembly: AssemblyVersion("1.0.0")]
[assembly: AssemblyCulture("Neutral")]
[assembly: AssemblyKeyFile(@"..\..\KeyPair.snk")]
```

C.3 Adding an Add-in to the GAC

Media Center add-ins must be installed in the global assembly cache (GAC). Media Center ignores add-ins that are in different locations, even if they are registered with Media Center. The GAC places strict security requirements

on all assemblies (including add-ins), and these requirements are enforced by Media Center.

During the development and testing stages, you can use the Global Assembly Cache tool, Gacutil.exe, to install your add-in into the GAC. This tool is included with all versions of the Visual Studio .NET SDK.

C.4 Registering the add-in

Registering an add-in is handled by an XML file that contains the registration information and by a command-line tool called RegisterMCEApp.exe.

The following XML registers an add-in so it can be launched from More Programs. The application element in this example has two attributes: title, which is the name of the application, and id, which is a GUID that serves as a unique identifier for the application. Remember to change the public key token to your public key token.

```
<application
title="Hello World Add-in"
id="{CCA64374-182E-43d8-9F7A-6DBBC4AEA4F0}">

<entrypoint id="{CCA64374-182E-43d8-9F7A-6DBBC4AEA4F1}"
addin="MediaCenter.Sample.AddIn.HelloWorldAddIn,
HelloWorldAddIn,
Version=1.0.0.0, Culture=neutral,
PublicKeyToken=43525660dcbb828f,
Custom=null"
title="Hello World Add-in"
description="A sample add-in for Media Center" >
<category category="More Programs"/>
</entrypoint>
</application>
```

To register the add-in, run "RegisterMCEApp.exe <path to xml file>"

Abbreviations

ACL Access Control List

Ajax Asynchronous JavaScript and XML

COM Component Object Model

COTS Commercial Off-The-Shelf

DRM Digital Rights Management

EPG Electronic Program Guide

GAC Global Assembly Cache

GUID Global Unique Identifier

HTPC Home Theatre Personal Computer

IME Input Method Editor

MCE Microsoft Windows Media Center Edition

MCE PC Personal Computer running Microsoft Windows Media Center Edition

MSAS Media State Aggregation Service

OEM Original Equipment Manufacturer

PDA Personal Digital Assistant

PKI Public Key Infrastructure

PVR Personal Video Recorder

RDP Remote Desktop Protocol

RMI Java Remote Method Invocation

STB Set-Top Box

WMDRM Windows Media Digital Rights Management

Bibliography

[1] Code Conventions for the Java Programming Language, http://java.sun.com/docs/codeconv/, May 9th 2006.

[2] Eclipse http://www.eclipse.org, May 3rd 2006.

[3] Embedded Automation, http://www.embeddedautomation.com, May 2nd 2006.

[4] Fast, http://www.fastsearch.no, May 3nd 2006.

[5] Global Assembly Cache, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconglobalassemblycache.asp, May 10th 2006.

[6] How to Write Doc Comments for the Javadoc Tool, http://java.sun.com/j2se/javadoc/writingdoccomments/index.html, May 9th 2006.

[7] HP Web Site, http://www.hp.com, May 2nd 2006.

[8] INSTEON Web Site, http://www.insteon.net/, May 2nd 2006.

[9] Java Remote Method Invocation documentation, http://java.sun.com/j2se/1.4.2/docs/guide/rmi/spec/rmiTOC.html, May 3rd 2006.

[10] JDOM, http://www.jdom.org, May 16th 2006.

[11] Saxlund, Kim: Digital Homes - Home Theatre Solution, Accenture Norway 2005.

[12] Leviton Web Site, http://www.leviton.com, May 2nd 2006.

[13] Linksys Web Site, http://www.linksys.com, May 2nd 2006.

[14] MCE Web Browser, http://www.anpark.com/software.aspx, May 19th 2006.

[15] McLaughlin, Brett: Mastering Ajax, http://www-128.ibm.com/developerworks/views/web/libraryview.jsp?search_by=Mastering+Ajax, Marsh 2nd 2006.

[16] Media Center Extender Web Site, http://www.microsoft.com/windowsxp/mediacenter/extender/default.mspx, May 2nd 2006.

[17] Media Portal Web Site, http://mediaportal.sourceforge.net, May 2nd 2006.

[18] Meedio Web Site, http://www.meedio.com, May 2nd 2006.

[19] Murray, Greg: Asynchronous JavaScript Technology and XML (AJAX) With Java 2 Platform, Enterprise Edition, http://java.sun.com/developer/technicalArticles/J2EE/AJAX/, Marsh 6th 2006.

[20] Nord-Trønderlag Elektrisitetsverk, http://www.nte.no, May 29th 2006.

[21] Nygård et al.: Kompendium i TDT4190 Distribuerte Systemer Vår 2004, IDI, NTNU.

[22] ORB Web Site, http://www.orb.com, May 2nd 2006.

[23] Proximis Girder Web Site, http://www.promixis.com/products.php?section=girder, May 2nd 2006.

[24] RetroSight, http://blog.retrosight.com/, May 15th 2006.

[25] SageTV Web Site, http://www.sage.tv, May 2nd 2006.

[26] Showshifter Web Site, http://www.showshifter.com, May 2nd 2006.

[27] Snapstream media Web Site, http://www.snapstream.com, May 2nd 2006.

[28] Eirikso - Eirik Solheim's blog, http://www.eirikso.com, May 15th 2006.

[29] Solheim, Eirik: HTPC Frontend roundup, http://www.eirikso.com/2005/05/30/htpc-frontend-roundup/, May 2nd 2006.

[30] Solheim, Eirik: NRK makes one of the worlds largest media center services, http://www.eirikso.com/2005/10/14/nrk-makes-one-of-the-worlds-largest-media-center-services/, May 2nd 2006.

[31] Stallings, William: Network Security Essentials Second Edition, 2003.

[32] SunŠs Code Convention for the Java Programming Language, http://java.sun.com/docs/codeconv/, May 9th 2006.

[33] The Green Button, http://www.thegreenbutton.com, May 2nd 2006.

[34] TvOnTime Web Site, http://www.tvontime.com, May 2nd 2006.

[35] Wikipedia, http://www.wikipedia.org, May 2nd 2006.

[36] Windows Media - Architecture of Windows Media Rights Manager, Microsoft Corporation 2004, http://www.microsoft.com/windows/windowsmedia/howto/articles/drmarchitecture.aspx, May 2nd 2006.

[37] Windows Media DRM Web Site, http://www.microsoft.com/windows/windowsmedia/drm/, May 2nd 2006.

[38] Windows XP Media Center Edition 2005 Home Page, http://www.microsoft.com/windowsxp/mediacenter/default.mspx, May 2nd 2006.

[39] Windows XP Media Center Hosted HTML Object Model Reference, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/MedctrSDK/htm/mediacenterhostedhtmlobjectmodelreference.asp, May 2nd 2006.

[40] Windows XP Media Center SDK, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/MedctrSDK/htm/windowsxpmediacentereditionsdk.asp, May 2nd 2006.

[41] Windows Media Center SDK - Creating an add-in, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/MedctrSDK/htm/creatinganaddin.asp, May 2nd 2006.

[42] Windows XP Media Center SDK download, http://go.microsoft.com/fwlink/?LinkId=19779&clcid=0x409., May 2nd 2006.

[43] Xbox Web Site, www.xbox.com, May 2nd 2006.