The killer app of UMTS (Mobile devices)
The killer app of UMTS (Mobile devices)
A Mobile phone application that stimulates the use of UMTS

By
M.B.Sobat (1054538)
1. Summary

Mobile communication brings deep changes to our everyday lives. The launch of 3rd generation mobile systems in 2001 in Japan and from 2002 throughout the rest of the world enables us to communicate at anytime and anywhere using a variety of services which have until now, only been available to fixed network users. In 2003, T-Mobile and TU Delft held a contest to design a UMTS based service that has to stimulate the use of UTMS. The t-info concept won the price. The essence of this concept is to develop a killer app to make it possible to get information anywhere and anytime in very user-friendly way.

The goal of this project is to implement a prototype of the t-info concept, which runs on a mobile phone device. This application will be able to establish a wireless connection to the server and start communication. This communication will be based on XML, which makes the application dynamic and powerful. The application also uses a media framework, which give the ability to play multi-media files on mobile phone.

This project started with a research, which investigated how mobile devices can be extended in their functionality to support the t-info concept. It covered areas, such mobile phone operating system, programming languages, networking and global positioning systems on mobile phone.
2. Preface

This paper is written by M.B.Sobat in October 2005. The author is currently studying at the Delft University of Technology (TU Delft, stands for Technische Universiteit Delft). Two assignments need to be fulfilled in order to obtain a master degree in Computer Science: A Research paper must be written about a specific technology which will subsequently followed by a Master’s Thesis Project, the former part is done in December 2004 [Sobat, 2004]. This document gives an overview of the implementation of the t-info prototype, which stimulates the use of UMTS. The prototype will be an application that runs on a mobile phone device.

Acknowledgments
I wish to express my thanks to all the people who have helped me during this research, with special regards to my supervisor, ir. H.J.A.M. Geers. I want to thank also Michele Nijhuis and Joost Hietbrink that have thought the t-info idea and subsequently let me work to this project. And last, but not least, I would like to express a special thank to my friends and relatives for supporting me during this period.

-M.B.Sobat, October 2005-
Table of content:

1. SUMMARY .......................................................................................................................... 5
2. PREFACE............................................................................................................................. 6
3. THE PROBLEM DEFINITION............................................................................................... 12
   3.1. INTRODUCTION.............................................................................................................. 12
   3.2. INTRODUCTION TO UMTS ......................................................................................... 13
   3.3. THE PROBLEM .......................................................................................................... 14
   3.4. THE T-INFO CONCEPT .......................................................................................... 16
       3.4.1. The t-info code .................................................................................................. 17
       3.4.2. The t-info application ....................................................................................... 18
       3.4.3. The t-info server ............................................................................................ 18
       3.4.4. The t-info website ......................................................................................... 19
       3.4.5. The t-info admin part ..................................................................................... 19
   3.5. THE PLAN.................................................................................................................. 20
4. REQUIREMENTS .................................................................................................................. 22
   4.1. INTRODUCTION........................................................................................................... 22
   4.2. REQUIREMENTS ELICITATION ................................................................................... 23
       4.2.1. The t-info application ....................................................................................... 23
       4.2.2. The t-info website ........................................................................................... 24
       4.2.3. The t-info Admin part ...................................................................................... 24
5. TECHNOLOGIES AND ENVIRONMENTS ......................................................................... 27
   5.1. INTRODUCTION........................................................................................................... 27
   5.2. THE OPERATING SYSTEM ........................................................................................ 28
       5.2.1. Symbian OS ........................................................................................................ 28
       5.2.2. Windows CE ....................................................................................................... 29
       5.2.3. Symbian vs. WinCE .......................................................................................... 30
   5.3. THE PROGRAMMING LANGUAGES .............................................................................. 32
       5.3.1. Java 2 Micro Edition ........................................................................................ 32
       5.3.2. Java on Symbian .............................................................................................. 38
       5.3.3. Symbian C++ .................................................................................................... 39
       5.3.4. .NET Compact Framework ............................................................................... 39
   5.4. CONCLUSIONS .......................................................................................................... 41
6. THE PURPOSE OF THE SYSTEM ....................................................................................... 44
   6.1. INTRODUCTION........................................................................................................... 44
   6.2. THE CONCEPT OF THE SYSTEM .............................................................................. 45
       6.2.1. The web architecture ........................................................................................ 46
   6.3. ACTORS ..................................................................................................................... 50
       6.3.1. The t-info Application........................................................................................ 50
       6.3.2. The t-info Website ............................................................................................ 50
       6.3.3. The t-info Admin ............................................................................................. 51
   6.4. THE FUNCTIONAL MODEL: USE CASES .................................................................... 52
       6.4.1. The t-info Application ........................................................................................ 52
       6.4.2. The t-info Website ............................................................................................ 53
       6.4.3. The t-info Admin ............................................................................................. 54
   6.5. THE ANALYSIS MODEL: CLASS DIAGRAM ............................................................... 56
       6.5.1. The t-info application ........................................................................................ 56
       6.5.2. The t-info website ............................................................................................ 57
       6.5.3. The t-info admin .............................................................................................. 59
7. SYSTEM DESIGN ............................................................................................................... 63
## Version History

<table>
<thead>
<tr>
<th>Version</th>
<th>Author</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Draft</td>
<td>M.B.Sobat</td>
<td>The problem</td>
<td>June 25, 2005</td>
</tr>
<tr>
<td>Ver 0.1</td>
<td>M.B.Sobat</td>
<td>Requirements</td>
<td>August 15, 2005</td>
</tr>
<tr>
<td>Ver 0.2</td>
<td>M.B.Sobat</td>
<td>Purpose of the system, System design</td>
<td>August 28, 2005</td>
</tr>
<tr>
<td>Ver 0.3</td>
<td>M.B.Sobat</td>
<td>Object design</td>
<td>September 18 2005</td>
</tr>
<tr>
<td>Ver 0.4</td>
<td>M.B.Sobat</td>
<td>Correction, conclusion</td>
<td>September 30 2005</td>
</tr>
<tr>
<td>Ver 0.5</td>
<td>M.B.Sobat</td>
<td>Implementation</td>
<td>October 05 2005</td>
</tr>
<tr>
<td>Ver 0.6</td>
<td>M.B.Sobat</td>
<td>correction</td>
<td>October 08 2005</td>
</tr>
<tr>
<td>Ver 0.7</td>
<td>M.B.Sobat</td>
<td>Implementations</td>
<td>October 09 2005</td>
</tr>
<tr>
<td>Ver 0.8</td>
<td>M.B.Sobat</td>
<td>Implementations</td>
<td>October 15 2005</td>
</tr>
</tbody>
</table>
Problem Definition
3. The Problem definition

3.1. Introduction

Mobile telephony has a great impact on our lifestyle, at levels that are far deeper than might seem at the first sight. Mobile phone technology growth was very fast in the past decades. The first mobile phone, back in 1946, used a single large transmitter on top of tall buildings and had only one channel for both sending and receiving. In the 1960’s, a new technology was installed. It used a high-power (200watt) transmitter. These transmitters were placed on top of hill, the same as previous technology but this one used two frequencies, one for transmitting and one for receiving data. Since different channels were used for sending and receiving. These systems were called: the First Generation Mobile phones (1G).

But everything changed, when the Second Generation mobile phones (2G) were introduced. In contrast with the first generation mobile phone, the 2G systems used digital voice. The 2G mobile phone attracted a lot of mobile phone user, which leads to some problems and new requirements.

First, data traffic already exceeds voice traffic on the fixed network and is growing exponentially. Second, the telephone, entertainment, and computer industries have all been specialized in digital technologies and are rapidly converging. Many people are drooling over lightweight, portable device that acts as a telephone, CD player, DVD player, web interface and more, all with worldwide wireless connectivity to the Internet at high bandwidth. This device and how to connect it is what third generation mobile telephony is all about. Before continuing, a short introduction to UMTS will be given in the next chapter.
3.2. Introduction to UMTS

UMTS (Universal Mobile Telecommunication System) is part of the International Telecommunication Union’s ITM-2000 vision of a global family of 3G communication systems. The idea of UMTS is to combine voice, text and video in the same connection. UMTS is above 2G mobile systems for its potential to support 2Mbit/s data rates from the outset. This capability, together with the inherent IP support, combines powerfully to deliver interactive multimedia services and new wideband applications, such as video telephony and video conferencing.

UMTS builds on today’s significant investments in GSM mobile systems worldwide and has the support of hundreds of network operators, manufacturers and equipment vendors. The launch of UMTS services will start the evolution of a new communication universe, including providers of information and entertainment services, coming together to deliver new communication services.

UMTS represents a new generation of mobile communications systems in a world where personal services will be based on a combination of mobile and fixed services providing seamless end-to-end service to the user. This requires the following:

- An integrated offering of services, irrespective of the serving (wired/wireless) network;
- Mobile technology that supports a very broad mix of communication services and applications;
- Flexible, on-demand bandwidth allocation for a wide variety of applications;
- Standardization that allows full roaming and inter-working capability but responsiveness to different types of markets.
3.3. The Problem

In the past few years a revolution has taken place, and it will continue to do so. The revolution causes the changes in the way we communicate and interact with each other and our environment. The arrival of the World Wide Web and mobile personal communications systems are the most prominent example of this revolution. The Internet made it possible to exchange information instantly on the worldwide scale and has released extraordinary amount of information into the public domain. Mobile phone communications allowed us to communicate everywhere and at any time.

With the technologies such as WAP and UMTS it is even possible to surf on the Internet. The impact of these technologies on all parties in the value chain of services delivery and information exchange is hard to underestimate. They will be an additional channel for some to offer their services, a new enabler for others who can now offer services that could not be offered before.

Nowadays, people want more from their mobile phone device, from taking pictures to reading news with their mobile phone. They also want to be able to get information anywhere and at any time. An example is when a person is visiting a museum and wants to get information, such as a video fragment, about an object in museum. He uses his mobile phone to get that information. There are a lot of companies that provide these kinds of services. They use SMS/MMS\textsuperscript{1} services or UMTS services. In the first case, the user has to send a SMS message to the service provider; the service provider sends a reply with information back to the user. The problem with these kinds of services is that the message contents are limited, for example, it is not possible to send a multimedia stream with a SMS.

In the second case, the information content can be anything, from textual information to multimedia file. This information is provided through a web-server, the user has to use some kind of client application on the mobile phone device to connect to the server and retrieve information from it. The same as the Internet, the UMTS services provide unprecedented amount of information.

SMS services are easy to use, a user send a SMS without knowing anything about the server. On the other hand UMTS provides better and high quality services, but before getting the information, the user has to know the address of the server for connection. In this thesis a research has been done on how to combine both services to get a solution that is easy to use and provides high quality services. The result will be a mobile phone application that connects to a server; this step will be transparent for the user. The user sends a request to the server; which is connected to the Internet and with other servers that provides services. The server retrieves the information and sends a reply with information back to users. The requirement is to take the full advantage of the new technologies, such as UMTS, by providing multimedia services.

\textsuperscript{1} Multimedia Messaging Service (MMS) allows for non-real-time transmission of various kinds of multimedia contents, such as images, audio, and video.
In 2003, T-Mobile and TU Delft held a contest to design a UMTS based service that has to stimulate the use of UMTS. The *t-info* concept won the price. The essence of this concept is to develop a killer application\(^2\) to make it possible to get (multimedia) information anywhere and anytime in very user-friendly way. Next chapter will describe the t-info concept in more detail.

---

\(^2\) An application that dominates competition, or becomes industry standard.
3.4. The t-info concept

The main goal of the t-info concept is that it has to be interesting for the mobile phone user and for the companies that provide services. By providing free of charge multimedia services, the user will be attracted to use the application. From the user’s point of view, the t-info concept should be easy to use and provide high quality services. It should also be attractive for the third party companies. They should be able to provide content with advertising themselves with it.

The t-info should consist of several parts:

- The first one is the t-info code. These (unique) codes can be found anywhere, on DVD-box, in the advertisements on the newspaper et cetera.
- The other part is the t-info application that is executable on mobile phones. It is more or less the same as a web-browser; it establishes a wireless connection with the t-info server and sends requests. It well gets the response back from the server and visualize it.
- The t-info server is the other part of the concept. It is connected to the Internet and contains a database. It is responsible for the computationally intensive tasks. It also uses other service for location-based applications or for e-commerce transactions.
- Other part of the t-info is a web application (the term the t-info web application and the t-info website will be used interchangeably). The t-info website supports dynamic configuration to the t-info application. A user can login into the t-info website and customize his configuration. Then the t-info application can download the configuration and use it.
- The last part of the t-info is the administration part. This part is also a web application and it is responsible for administrative tasks, such as managing the database. This part is attractive for the third part companies that provide information and advertise their products.

Figure 1 shows all users in one picture.
In the next chapters, each part will be described more precisely.

3.4.1. The t-info code
The t-info code is a unique code to be found everywhere! On DVD’s in the video store, posters in the cinema, in the paper, on TV, at the IKEA, on somebody’s T-shirt, on a church, etc.

The first element makes the t-info code recognizable. The same as the ‘@’ (pronounce: at) sign that expresses the email sign, the t-info sign, shown in Figure 2, expresses the t-info code. One can imagine that, for example, a DVD box contains a lot of texts and pictures. Without a t-info logo it is very difficult to find the t-info code. Figure 2 shows a t-info code with the t-info logo.
Almost all mobile phones use keypads for input. A keypad is used primarily for entering numeric data but they are used to enter alphabetic characters. This is done by assigning multiple characters to a key and letting the user cycle through them by pressing a key several times. This process can be used for limited text entry.

3.4.2. The t-info application

When a user wants to use a t-info service, he enters the t-info code into the t-info application that runs on a mobile phone.

The application must be compatible with as many mobile phones as possible. Moreover, the application must be user-friendly. The user should get his service in just a few steps. Besides these, the application must run on not-UMTS devices (with fewer features). This property should encourage the potential users to use UMTS. And as the finishing touch, the application should support location-based devices. Such as, 9292OV and Reisplanner services.

The idea is to build a killer application that covers all areas, from getting information about a movie to getting information about a historical object somewhere in the town hall. To make the application easy to use, the t-info concept is extended to support dynamic configuration.

3.4.3. The t-info server

The t-info server is connected to the Internet and consists of database. This part is responsible for the main activities, such as, calling other servers and using their services, providing the content to the t-info application and security issues.
3.4.4. The t-info website
The t-info website is used to by the t-info user to customize the favorite list for the t-info application.

3.4.5. The t-info admin part
The t-info admin is responsible for administrative tasks, such as adding a new code, changing the data and keeping the server up to date.
3.5. The Plan

The goal of the project is to design and implement the t-info parts, listed in the previous chapter. The project is divided into several phases:

- **Phase 1, formulating the problem** – this task has been done in the previous chapter by dividing the t-info concept into different parts and describing them.

- **Phase 2, research** – the goal of this phase is to find solutions to some theoretical problems. The first basic problem is to write an application that is suitable for mobile phone? What are the tools and how can we upload the application into a mobile device? This phase is already done in the research phase of this thesis project. [Sobat, 2004]

- **Phase 3, technologies and environments** – this phase will describe different technologies and environments and select the technology that is best suited.

- **Phase 4, requirements** – the goal of this phase is to define the functional and non-functional requirements for each t-info part.

- **Phase 4, a global design** – the goal of this phase is to give a proposal solution to the given problem. It will not go too deep into the technical detail. In the implementation phase different priorities to each part will be given. In the listing below these priority is revealed:
  1. The t-info Application – the main task of this project is to implement an application that is suitable for mobile phone devices.
  2. The t-info Server – the t-info application operates like a client that connects to the t-info server and communicates. Since both parts are dependent on each other, the same priority will be given to both parts.
  3. The t-info Admin – this part will be given a higher priority than the t-info website. This is because this part is much more interesting for the third party companies. This part is meant to show the usability and user-friendliness of managing the content.
  4. The t-info Website – this part will be more interesting for the t-info users and will be give the lowest priority.

- **Phase 5, design and implementation** – the main goal of this phase is to give design for each t-info parts

- **Phase 6, case study** – in this phase a real world case study will be given to show the capabilities of the t-info concept. In this phase the t-info project will be combined with another thesis project (WSDF web-services transaction framework).
Requirements
4. Requirements

4.1. Introduction

After defining the problem the next step is the requirement engineering. This step includes two key activities; the first one is requirements elicitation and the other one is analysis. The task of requirements elicitation is to describe the goal of the system. After identifying the problem area, the client, the developers, and the users define a system that addresses the problem. This definition serves as a contract between the client and the developers, and it is called a system specification.

During analysis the system specification is structured and formalized to produce an analysis model. These two documents, analysis model and system specification, represent the same information but they differ in the language. The system specification is written in natural language while the analysis model is usually expressed in a formal or semiformal notation.
4.2. Requirements Elicitation

Requirements elicitation results in the specification of the system that the client understands. The t-info system consists of several parts, as it is described in the chapter “Requirements of t-info”. This chapter will define the following requirements elicitation concepts for each part:

- Functional requirements – Functional requirements capture the intended behavior of the system. This behavior may be expressed as services, tasks or functions the system is required to perform.
- Nonfunctional and Pseudo Requirements – It describes user-visible aspects of the system that are not directly related with the functional behavior of the system.

4.2.1. The t-info application

This application is executable on mobile phones. It is more or less the same as a web-browser; it establishes a wireless connection with the t-info server and sends requests with some parameters, and gets the response back from the server and visualizes it.

Functional Requirements

- The user has to be able to enter the t-info code into the application and request the service associated to the t-info code.
- The user has to be able to select a service from a pre-defined favorite list and request that service.
- The user has to be able to download a favorite list from the server.
- The user has to be able to save the favorite list into the mobile phone device.
- Depending on the mobile phone device, the application should visualize the information in either textual or multimedia formats (e.g. movie clip, pictures etc.), or both.
- Depending on the mobile phone device and on the provider, the application should be able to get its cell id or geographical coordination.

Nonfunctional and Pseudo Requirements

- After sending a request to the server, the user must get a response within 20 seconds. This latency depends on the amount of information. For example, a multimedia file requires more download time (at most 30 – 40 seconds).
- The application must be user-friendly. The user has to be able to get a service in just few steps.
- The application must be compatible with as many mobile phones as possible.
- The application has to be able to run on not-UMTS devices (with fewer features).
- The application must be maintainable, by providing the new updates and upgrades to the user.
- The application must be extendable, by making use of standards.
- All related software associated with t-info application will be written using Java, to make the application portable.
4.2.2. The t-info website

The t-info system is extended to support dynamic configuration of the favorite list. A user can login into the t-info website and customize his configuration. Then the t-info application can download the configuration and save it.

Functional Requirements
- The website must control the authentication.
- The user\(^3\) has to be able to view the configuration file (e.g. favorite list).
- The user has to be able to change the configuration file.
- The user has to be able to change the setting for his mobile phone. That is, for example, a user wants to get only textual information.

Nonfunctional and Pseudo Requirements
- The website must be protected from unauthorized users.
- The website should be user-friendly, that is, the user should be able to find his way in the website. This can be done by providing a good menu structure and consistent metaphor.

The website should have a standard and consistent layout.

4.2.3. The t-info Admin part

This part is also a web application and it is responsible for administrative tasks, such as managing the database. To make distinction with the t-info website, the term backend will be used for the t-info Admin part

Functional Requirements
- The backend must control the authentication
- The administrator has to be able to view all t-info users.
- The administrator has to be able to add a new t-info user.
- The administrator has to be able to modify an existent t-info user.
- The administrator has to be able to delete an existent t-info user.
- The administrator has to be able to view the statistics of users, such as page hits, number of requests per user etc.
- The administrator has to be able to create a new t-info code and associated information content to it.
- The administrator has to be able to modify the information content, associated with a t-info code.
- The administrator has to be able to delete a t-info code with its related service.

Non-functional Requirements and Pseudo-Requirements
- The backend must be protected from unauthorized administrators.
- The backend should be usable, that is, the administrator should be able to find his way in the website. This can be done by providing a good menu structure and consistent metaphor.

\(^3\) An authorized user.
The backend should have a standard and consistent layout.
Technologies
And
Environments
5. Technologies and environments

5.1. Introduction
Before starting with the design and implementation, a developer has to know the current environments and technologies, and choose the right one. The choice of the right technology and environment will be based on the requirements of the system. In the previous chapter the requirements of the t-info system have been defined. The goal of this chapter is to compare the different environments and technologies, and to select the technology that is best suited. This chapter is divided into two parts: in the first part the operating systems those are used in the mobile phone will be compared. And the second part will compare the programming languages those are used to write a mobile phone application.
5.2. The Operating System

Mobile phones are no longer used simply for voice communication; increasingly, they support SMS, email, web surfing and even video transmissions. All these features, call for more sophisticated mobile operating systems. Most mobile phones in the market today have operating systems designed by their manufacturers, but that will change as more robust capabilities are built into handsets. Complex devices will drive market toward third-party OS makers. In this section we will discuss two major Mobile phone operating systems, Symbian OS and Microsoft Windows CE. In chapter 5.2.1 Symbian OS will be discussed and in chapter 5.2.2 Windows CE will be investigated. And in chapter 5.2.3 a comparison of WinCE and Symbian OS will be given.

5.2.1. Symbian OS

Symbian OS is a rich mobile phone operating system. It has its own libraries, user-interfaces frameworks and reference implementation of common tools, produced by Symbian. Symbian can be used in various interaction style of hand-held devices. This is because it supports different kind of user-interfaces; such as pen-based user interfaces and devices with a keypad or a keyboard.

A mobile phone OS must be compact to fit into the limited amount of memory. But it must still provide a rich set of functionality. Symbian is more than a mini-operating system. It is structured like many desktop operating systems, with pre-emptive multitasking, multithreading and memory protection.

Symbian operating systems are made for handheld devices with a limited amount of memory that may run for years without restarting. There is a strong emphasis on conserving memory, using Symbian-specific programming idioms such as descriptors and a cleanup stack. This and other techniques, keep memory usage low and memory leaks rare. All Symbian OS programming is event-based; the CPU is switched off when applications are not directly dealing with an event. Symbian uses active objects. In this schema, objects request for asynchronous services (e.g. sending an SMS). The OS will do the task and returns the control back to the object. Now, the object will decide to keep the control and use it or give it back to OS. When an object requests for an asynchronous service, the object will be put in some queue. When the task is completed, the OS will identify the thread containing the requesting active object, and wake up that thread and pass the control to the object.

Symbian is committed to open standards. It has a POSIX-compliant interface and a Sun approved Java Virtual Machine (JVM). Symbian is working with emerging standards, such as J2ME, Bluetooth, MMS, SyncML, IPv6 and WCDMA [Symbian White Paper 2003].

---

4 In this form of multitasking each process, in turn, is granted a portion of CPU time (time slice). When a process does not finish its task within the time slice, it will be context-switched with another process.

5 A light weighted process.
Symbian supports a lot of programming languages. In fact, Symbian OS is able to run machine code. All languages that are compiled to machine codes or where its virtual machine can be compiled into machine codes can be uploaded into the device and might run.

Symbian is owned by major mobile phone companies, such as Ericsson, Nokia, Matsushita (Panasonic), Psion, Siemens and Sony Ericsson.

**Symbian OS features**

In this chapter, the main features of Symbian OS will be summed up [www.symbian.com]:

- **Integrated multimode mobile telephony** – Symbian OS integrates the power of computing with mobile telephony, bringing advanced data services to the mass market. Symbian is the operating system of choice for 2.5G and 3G mobile phones.

- **Messaging** – Symbian OS supports SMS, EMS, MMS and e-mail and fax services. Peer-to-peer multimedia messaging is a key revenue generator for 2.5G and 3G networks.

- **Open application environment** – Symbian OS enables mobile phones to be a platform for deployment of applications and services developed in a wide range of languages (java and C++) and content formats.

- **Standards and interoperability** – with a flexible and modular implementation, Symbian OS provides a core set of application programming interfaces (APIs) and technologies that is shared by all Symbian OS phones. Key industry standards such as IP v4 and v6, Bluetooth, Java, WAP, SyncML are supported.

- **Multi-tasking** – fully object-oriented and component-based, Symbian OS includes a multi-tasking kernel, middleware for communications, data management and graphics, the lower levels of the graphical user interface framework, and application engines.

- **Robustness** – Symbian OS maintains instant access to user data. It ensures the integrity of data, even in the presence of unreliable communication, and limited resources such as memory, storage and power.

- **Flexible user interface design** – by enabling flexible graphical user interface design on Symbian OS, Symbian is fostering innovation and is able to offer choice for manufacturers, carriers, enterprises and end-users. Using the same core operating system in different designs also eases application porting for third party developers.

### 5.2.2. Windows CE

Windows CE\(^6\) (WinCE) is operating system for embedded devices. It is a variation of Microsoft’s Windows operating system. Like Symbian, it is not just a mini-operating system or a trimmed down version of desktop computer operating systems. It is scalable,

---

\(^6\) CE stands for Compact Edition, but it is not official.
32-bit operating system that is implemented to meet the needs of broad range of intelligent devices, from enterprise tools such as industrial controllers, communications hubs, and point-of-scale terminals to consumer products such as cameras, telephones and home entertainment devices [Microsoft Corporation 2001]. WinCE is supported on Intel x86 and look-alikes, MIPS, ARM family and Hitachi SuperH processors.

WinCE is optimized for devices with limited amount of memory; its kernel may run in under a megabyte of memory. It may be configured as a ‘closed’ system that does not allow for user extension.

WinCE supports hard real-time processes with deterministic interrupt latency, nested interrupts. It also supports 256 priority levels and provides for priority inversion. The most important aspect of WinCE is that the fundamental unit of execution is the thread, providing for simpler, faster concurrent programming [Microsoft Corporation 2004].

Windows CE 3.0 is build up by discrete components. Each of these components provides full or partial support for major features of the OS. A device manufacturer can select a set of these components, put them together and build up an operating system that meets the requirements.

WinCE supports Microsoft ActiveX controls, message queuing (MSMQ), component Object Model (COM) interfaces, Active Template Library (ATL) and the Microsoft Foundation Class (MFC) Library. There is also a built-in support for multimedia, including Microsoft DirectX, communications (TCP/IP) and security [Microsoft Corporation 2001].

WinCE also supports internationalization. That is, it supports different languages and bidirectional texts such as Arabic and Hebrew. And it is based on Unicode.

5.2.3. Symbian vs. WinCE

The two major operating systems, without any doubt, are Symbian and WinCE. The ideal situation is that the developers write an application that would run on variety of mobile phones. But this is possible only if one operating system ran on a variety of devices. Currently, it is not the case. So the developers have to spend a lot of money and time porting applications to different systems.

Symbian tried to make a standard mobile phone operating system. They have garnered significant support. More than 18 mobile phones from big companies- such as Nokia, Siemens and Sony Eriksson- now rely on the OS.

After being successful in desktop computer market, Microsoft has tried to push its Windows operating system down into cell phone with WinCE. The main advantage of Microsoft is their product’s compatibility with their large base of desktop application. Microsoft made an OS for mobile phone which is compatible with Microsoft’s Outlook, Internet Explorer, MSN Messenger and Windows Media Player.
Symbian, currently the leader in the market, is an open source OS; the developers can customize an operating system for their need. Symbian produces operating systems and not produce applications. Microsoft can take this advantage by charging almost zero licensing fees per handset and making money by selling applications. This is possible because, the amount of money that’s involved in the handset operating business is so small that it’s really nothing to Microsoft.

The conclusion is that, Symbian is a better operating system. It is open source and supports a lot of programming languages. But Microsoft is a company with an imago and money. This can be a problem for Symbian.
5.3. The programming languages

This chapter compares three major programming languages for mobile phone applications. Note that any programming language can be compiled into target machine code and run on the mobile phone, but this chapter will put focus on programming languages aimed for embedded devices hence for mobile phone. In the next chapters the programming languages; Java (J2ME), Symbian C++ and .NET Compact Framework will be discussed.

5.3.1. Java 2 Micro Edition

The Micro Edition was aimed at a range of consumer and embedded electronic devices with constrained resources. “Java 2 Platform, Micro Edition is highly optimized Java runtime environment targeting a wide range of consumer products, including pagers, cellular phones, screen-phones, digital set-top boxes and car navigation systems” [http://java.sun.com/j2me/]. From “highly optimized Java runtime environment“, it is clear that J2ME doesn’t define a new Java with new runtime environment. But J2ME is an adapted version of J2SE. This is means that an application written in J2ME can be run in J2SE and even in J2EE environment.

Devices with limited amount of memory require a small runtime environment. J2ME reduces the size of the runtime classes installed with the runtime environment by removing unnecessary classes to form a new set of core classes. This reduction also occurs within the classes themselves, here, unnecessary or duplicate methods are removed. The result of these reductions is a true subset of the J2SE runtime classes.

Since its release in June 1999, over 600 companies have joined the development effort, including large corporations such as Palm, Nokia, Symbian and RIM [Mahmoud, 2002].

Standard Edition used the one-size-fits-all principle. This is not suitable for small devices. The J2ME runtime environment consists of Configurations and Profiles and it is shown in Figure 4.
The Java virtual machine (JVM) runs on top of a device’s host operating system. Configurations consist of programming libraries that provide basic functionality based on the resource requirements of the device. They are placed above the JVM. Profiles are placed on top of the Configurations. Profiles are extra programming libraries that take advantage of relatives’ functionalities on similar devices. In the next chapters, all these components will be described.

The KVM

The KVM (stands for Kuaui VM) is a completely new implementation of a JVM; it is developed for small devices. The KVM implementation is based on the Java Virtual Machine Specification, except for some specific deviations that are necessary for proper functioning on small devices.

The KVM is available on Sun Microsystems’ Web site. Developers can download it and compile it for the Palm, Windows, or Solaris platforms (or port it to any other platform).

As it was mentioned, the KVM implementation was based on the JVMS. The following features are known [Giguere, 2000]:

- Long integer and floating point data-types are optional – these types can be simulated in software on small devices. Floating point operations are expensive without a dedicated coprocessor.
- No object finalization – this makes the job the garbage collector much simpler.
- No JNI support – native methods in KVM are compiled into the virtual machine and are not user installable.
Off-device class verification – Normally, after a class is loaded (but before it is used), the class is processed by the class verifier. This process is an expensive and time-consuming operation. The verification of a J2ME program is partly done on desktop computer. This step is referred to as pre-verification.

- Multidimensional arrays are optional.
- No user-defined class loaders.

The optional features can be included and compiled together with the KVM, if it is needed.

**Configurations**
The main goal of configurations is to define minimum requirements for memory, language support, virtual machine support, and runtime libraries. Configurations define a horizontal grouping of products based on the available memory and processing power of each device. Currently, the expert groups split the devices into two families, each targeted by a different configuration. These are:

- The Connected Device Configuration (CDC)
- The Connected Limited Device Configuration (CLDC)

![J2ME architecture](image)

**The CDC**
The CDC aims at all devices that are less restricted in terms of power, memory and network bandwidth. Devices in this family include television set-top boxes and car entertainment or navigation systems. The CDC contains the same JVM as J2SE. The difference lies in the respective devices’ memory and display capabilities. The following resource requirements for CDC devices are given by the J2ME CDC specifications [www.jcp.org]:

---

**Technologies and environments**

- Off-device class verification – Normally, after a class is loaded (but before it is used), the class is processed by the class verifier. This process is an expensive and time-consuming operation. The verification of a J2ME program is partly done on desktop computer. This step is referred to as pre-verification.
- Multidimensional arrays are optional.
- No user-defined class loaders.

The optional features can be included and compiled together with the KVM, if it is needed.

**Configurations**
The main goal of configurations is to define minimum requirements for memory, language support, virtual machine support, and runtime libraries. Configurations define a horizontal grouping of products based on the available memory and processing power of each device. Currently, the expert groups split the devices into two families, each targeted by a different configuration. These are:

- The Connected Device Configuration (CDC)
- The Connected Limited Device Configuration (CLDC)

![J2ME architecture](image)

**The CDC**
The CDC aims at all devices that are less restricted in terms of power, memory and network bandwidth. Devices in this family include television set-top boxes and car entertainment or navigation systems. The CDC contains the same JVM as J2SE. The difference lies in the respective devices’ memory and display capabilities. The following resource requirements for CDC devices are given by the J2ME CDC specifications [www.jcp.org]:

---
Technologies and environments

- The device is powered by a 32-bit processor.
- The device has 2 megabyte or more of total memory available for Java. This includes both RAM and flash memory or ROM.
- The device requires the full functionality of the Java 2 “Blue Book” virtual machine.
- The device has connectivity to some kind of network, often with a wireless, intermittent connection and with limited bandwidth.
- The device may have a user interface with some degree of sophistication, but a user interface is not mandatory.

The CDC is shown schematically in Figure 6.

![Figure 6: The CDC schema](image)

**The CLDC**

This configuration is more common in the J2ME world. Today, some of the devices that you might find powered by the CLDC include mobile phones, two way pagers, PDAs, and personal organizers. In contrast with CDC, which requires a full and complete JVM, the CLDC deviates a bit from JVMS. The CLDC defines the following requirements [www.jcp.org]:

- The device can have between 160 and 512 kilobytes of total memory available for Java platform, including both RAM and flash memory or ROM.
- The device can have limited power, such as battery-powered operation.
- The device has connectivity to some kind of network, often with a wireless, intermittent connection and with limited (9600 bps or less) bandwidth.
• In addition, the device may have a user interface with some degree of sophistication, but a user interface is not mandatory.

Figure 7 shows the CLDC schema.

Profiles
While configurations provide the foundation for Java programming on small computing devices, they don’t by themselves provide enough functionality for most application development. The developers need to combine the configurations with profiles and perhaps even with device-specific class. For example User Interface classes are common example. A configuration doesn’t define them. This is because user interfaces are not supported by all devices. A profile is a set of APIs that reside on top of a configuration that offers the program access to device-specific capabilities. The goal of a profile is to define APIs for devices that have similar uses.
Figure 8 shows the J2ME environment. It shows some examples of profiles that are currently offered through J2ME. The Mobile Information Device Profile (MIDP) is designed to be used with CLDC, and defines a set of APIs for use by mobile devices, such as mobile phones and two-way pagers. In this paper only MIDP will be discussed.

**The Mobile Information Device Profile (MIDP)**

The MIDP is built on top of CLDC, and defines an open application development environment for what Sun calls Mobile Information Devices (MIDs). A MID is a small computing device that has the following characteristics [Mahmoud, 2003]:

- A monochrome or color display that is at least 96 pixels wide and 54 pixels high.
- A touch screen, keypad, or a conventional keyboard.
- A wireless network connection with limited bandwidth. The connection does not have to be continuously available.
- 128K of non-volatile memory in order to store the MIDP components, 8K of non-volatile memory in order to store persistent data, and 32K of volatile memory for the Java runtime.

All in all, MIDP is a J2ME profile that is used for wireless devices, such as mobile phones.

The MIDP is supported by major companies, including Nokia, NEC, Siemens, Motorola, Palm and more. MIDP 1.0 was introduced in 1999. It attracted a lot of attentions and generated substantial enthusiasm amongst the Wireless Java Community. But it was soon
realized that MIDP 1.0 on its own provided limited access, and so limited capabilities, to the functionality to the typical smartphone from within a MIDlet. Consequently, J2ME came up with MIDP 2.0 and its final form was released in 2002. MIDP 2.0 came with a range of extension API Java Specification Requests (JSRs). These optional packages increase the functionality available to MIDlets. It delivers an enhanced user interface, multimedia and game functionality, greater connectivity, over-the-air (OTA) provisioning, and end-to-end security to mobile information devices such as mobile phones and entry level PDAs.

5.3.2. Java on Symbian

In this chapter Symbian’s Java Technology will be described. Symbian recognized the strength of the MIDP movement by including J2ME MIDP 1.0 as its standard Java offering in Version 7.0 of Symbian OS as well as back-porting it to earlier versions. Symbian released version 7.0s of Symbian OS in 2003 and introduced support for J2ME MIDP 2.0, which brings a new, finer-grained security model, enhanced user interface API, Game and audio APIs and Push Registry to the Java platform. Beside this, Symbian OS also provides support for Java API for Bluetooth Wireless Technology (JABWT, JSR 782). Through this, a MIDlet can gain access to the Bluetooth stack and the Wireless Messaging API (WMA, JSR 120), which allows the MIDlet to send and receive SMS messages. Version 7.0s makes use of Sun’s high performance CLDC HI VM [www.symbian.com].

In the first quarter of 2004, Symbian OS Version 8.0 was announced. This enhances the J2ME CLDC/MIDP implementation adding the following optional packages to Symbian OS [www.symbian.com]:

- Mobile Media API (JSR 125)
- Mobile 3D Graphics (JSR 184)
- File GCF (part of JSR 75)

And Sun’s CLDC HI 1.1 VM was used. All in all, the benefits of Java on Symbian OS are [www.symbian.com]:

- Symbian’s Java implementation is robust. It is running on the Symbian OS kernel, which is itself designed to operate reliably. User data loss in Symbian OS phones is very rare, and system resets are rarely required.
- Symbian OS Java is fast. Symbian’s MIDP 1.0 provides support for ARM’s VTK software accelerated interpreter, whilst MIDP 2.0 uses Sun’s CLDC HI VM. Symbian’s Java UI components map directly to native UI components. Symbian OS in turn is extremely efficient.

---

Java Specification Request: the process by which new Java specifications are defined.
Technologies and environments

- The Java implementation has a small footprint; taking advantage of Symbian OS’s lean and means philosophy.
- Symbian’s implementation of new APIs derived from the Java Community Process is extending the functionality of MIDlets.

5.3.3. Symbian C++

The ANSI C++ does not contain GUI libraries or even thread handling. For this purpose, Symbian C++ is slightly different from ANSI C++.

First, a comparison of Symbian C++ and ANSI C++ will be given and then Symbian C++ will be discussed further.

ANSI C++ vs. Symbian C++

All applications\(^8\) written and compiled in Symbian C++ are actually DLLs, this leads to the fact that applications can’t have static writeable data. Sometimes it is useful to use global variables, this is why, Symbian OS provides a way to implement them on a per thread basis. This is called Thread Local Storage (TLS), which can be accessed through the class Dll. This TLS is simply a TAny * pointer\(^9\). All Symbian C++ application are platform depended, beside this, it is very difficult and risky to write an application with it. There are some programming style, name conventions, inheritance rules and exceptions handling that needs to followed by the programmer. This constrains can not be checked by compilers, which makes it very risky and difficult to use Symbian C++.

5.3.4. .NET Compact Framework

Like Java, .NET provides a compact version of .NET framework. It’s called .NET Compact Framework. In this chapter .NET Compact Framework will be discussed briefly.

The .NET Compact Framework is a hardware-independent environment for devices with limited resources, such as the Pocket PC, mobile phones, set-top boxes, automotive computing devices, and custom-designed embedded devices built with the Windows CE .NET operating system.

It is a subset of the .NET Framework class library and also contains classes exclusively designed for it. It inherits the full .NET Framework architecture of the common language runtime and managed code execution.

The key functionalities of .NET Compact Framework are [http://msdn.microsoft.com/]:

---

\(^8\) The file type of a standard Symbian C++ application is .app

\(^9\) In ANSI C++ it is void *
Technologies and environments

- Runs programs that are independent of hardware and operating systems.
- Supports common network protocols, and connects seamlessly with XML Web services.
- Provides developers with a model for targeting their applications and components to either a wide range or specific category of devices.
- Provides benefits of design and optimization of limited system resources.
- Obtains optimal performance in generating native code using just-in-time (JIT) compilation.

Figure 9 shows the .NET Compact Framework architecture.

![Figure 9: the .NET Compact Framework Architecture](image)

The .NET Compact Framework is designed to operate optimally on systems with the following characteristics [http://msdn.microsoft.com]:

- Battery powered
- Approximately 5 to 10 times the flash memory or RAM file system capacity to store the .NET Compact Framework and its applications in dynamic RAM (DRAM)
- A working space from as small as 128 KB to 1 MB in DRAM
- Optional hard drive
The memory management is very efficient and careful. Only application can access the RAM and it frees the RAM when the application quit. The .NET Compact Framework aggressively releases internal data structures that are not needed by the currently executing code whenever memory is scare.

The same as full .NET Framework, the .NET Compact Framework supports, just-in-time code compilation and garbage collection. It supports the Common Language Specification (CLS).

Applications for the .NET Compact Framework use assemblies. The .NET Compact Framework accesses portable executable (PE) files, which contain the Microsoft intermediate language (MSIL) and metadata that define a .NET Framework application. A PE file can refer to a programming namespace defined and shared by other assembly files. In contrast with J2ME, an application being compiled in the .NET Compact Framework, can not be run in the .NET Framework and visa versa.

Cross-language interoperability is built into the .NET Compact Framework. Visual C# and Visual Basic .NET are the first supported languages. And it provides a multithreaded programming model that uses the scheduling mechanism of the host system.

### 5.4. Conclusions

This chapter has shown that just as a desktop computer, a mobile phone has an operating system. Operating systems are developed with special purpose. Some of the operating systems are to provide convenience while some other operating systems are made to provide efficiency.

From the user point of view, mobile phone operating systems are slightly different from desktop computer operating system. But internally they are largely different. A mobile phone operating systems needs to be smaller than PC operating systems. Beside this, they have a lot of work. From the programmers’ point of view it provides API and system calls. This gives the programmers the opportunity to use high-level programming languages, such Java, to write a mobile phone application.

After exploring the mobile phone operating systems, this chapter continued by comparing three programming languages. The first one was Java, which easy to use. It provides garbage collector, which make it easier and safer to write programs. On the other hand, C++ is more powerful and fast but very difficult to manage and often very risky to use.

C++ object programs are not portable because they are always translated into the machine language for a specific machine. Portability is extremely important to many companies. If a company can "write once and run everywhere" they can save an enormous amount of money. If a developer can do this, they will have a much larger market for their software. Java is portable but some operating systems provide their own package. If a program uses these packages, the portability is not valid anymore.
Technologies and environments

The .NET Compact Framework provides portability and cross-language interoperability. The most common used language for this framework is C#. The same as J2ME, it is meant for devices with constrained-resources. But it is not widely used because the major mobile phone companies don’t support it. The other reason is that .NET Compact Framework requires relatively bigger memory size than J2ME.

From the t-info requirements becomes clear that the application must be compatible with as many mobile phone types as possible. For this reason, the chosen programming language is Java.
The Purpose Of the System
6. The purpose of the system

6.1. Introduction
In the chapter about requirements (chapter 4), the requirements of the system has been defined. This chapter starts by mapping the problem statements into the system specification that is represented as a set of actors, scenarios, and use cases. Then the analysis model will be applied to gain the class diagrams. Before doing this, it is important to describe the general concept of the system.
6.2. The concept of the system

The concept of the t-info system is quite new where a mobile phone user uses his mobile phone to connect to a server and use different kind of services. The communication is based on client/server architecture. This section will give a global description of the system.

Figure 10 shows the t-info system with the t-info server as its main part. The t-info server is connected to the Internet. It also contains an internal database. When it receives a request from a t-info application, it will start a process. During this process, the t-info server may call some external web-services or manipulate the internal database. Before going into the detail, the web architecture will be described in short. Next paragraph will explain the web architecture and the t-info architecture.
6.2.1. The web architecture

The t-info concept is based on the client/server architecture. A t-info user enters a t-info code into the t-info application. The t-info application establishes a wireless connection with the t-info server. After this process, the t-info application sends a request (together with the t-info code and additional information) to the server. The server may invoke some external web-services or retrieve the data from the database and send a response back to the t-info application that shows the result to the user.

The communication between the t-info application and the t-info server is based on the Hypertext Transfer Protocol (HTTP). In the next section an introduction to HTTP is given and the wireless application protocol (WAP) will be described. At the end a description on how to use both technologies to meet the requirements of the t-info will be discussed.

The Hypertext Transfer Protocol (HTTP)

This is a request/response protocol between clients and servers. A client (the t-info application) initiates a request by establishing a TCP connection to a particular port on a remote host (the default port for http is 80). A HTTP server listening on that port waits for a client to send a request string. Each interaction consists of one ASCII request, followed by on RFC 822 MIME-like response. RFC 2616 defines that all clients and all servers must obey this protocol. Some of its important properties will be discussed below.

The first important property of HTTP is the connection. The client establishes a TCP connection to port 80 on the server’s machine. Neither client nor servers have to worry about lost messages, duplicated messages, long messages, or acknowledgements. All these matters are handled by the TCP implementation. There are two different versions of HTTP. In HTTP 1.0, a client sends a request to the server; the server sends a response back to the client. After this, the connection will be released. On the other hand, HTTP 1.1 supports persistent connections. This enables a client to send request and get a response, and then send additional requests and get additional responses. The TCP connection is released after multiple requests. The relative overhead in this case, due to TCP is much less per request. It is also possible to send more than one (usually 2) request before getting responses from previous requests. This technique is called pipeline requests.

The other important property is the methods that are available. HTTP provides some operations, called methods. The built-in HTTP request methods are:

1. GET: Request to read a Web page
2. HEAD: Request to read a Web page’s header

---

10 This stands for Request For Comment. A RFC document is one of a series of numbered Internet informational documents and standards widely followed by commercial software and freeware in the Internet and Unix communities.

11 The names are case sensitive
The purpose of the system

3. PUT: Request to store a Web page
4. POST: Append to a named resource
5. DELETE: Remove a Web page
6. TRACE: Echo the incoming request
7. CONNECT: Reserved for future use
8. OPTIONS: Queries an option

Each request consists of one or more lines of ASCII text with the first line being the method requested, as above.

Figure 11 shows the communication between a client and a server, the figure shows the client as a desktop computer but it can also be a mobile phone or other device.

![Figure 11: HTTP communication](image)

In the figure above, the following steps occur:
- The client requests for a TCP connection to the server
- The server agrees to the connection and sends a response to the client
- The client starts the communication with the server by sending an HTTP request on the open channel
- The server receives the request and responds to it

These steps show that two client/server TCP roundtrips are required: one full trip for the initial connection establishment, a half trip for the client HTTP request, and a half trip for the beginning part of the returned data. The whole transfer process can be summarized by the formula

\[2\rho + \tau\]
The purpose of the system

\( \rho \) stands for the roundtrip time, the time between sending and receiving a request, between client and server. And \( \tau \) stands for the time to transfer the actual data (bytes) of content [G, Barish, 2002].

**Request Processing**
The request processing plays a key role in all server/client communication. The relationships between the server and client aren’t limited to the end user and the Web server but in fact exist throughout many parts of any \( n \)-tier application architecture. As an instance, the Web server itself may also be a client to various other server-side application components and these components, in turn, may be clients to each other. All these communication requires request processing.

![Figure 12: the detail of request processing](image)

Figure 12 illustrates the main requirements needs to be met for the communication to be successful [G. Barish, 2002].

- **Connection management:** clients need to be able to establish connections to servers, and servers need to be able to accept and manage client connections.
- **Data marshalling:** request and replies need to be converted to a suitable form for transmission over a network. The challenge is in reducing the amount of data being marshalled and the complexity of marshalling.
- **Request servicing:** this part involves:
  - Performing some computation
  - Querying some local or external resource

The request processing knows two modes: synchronous and asynchronous. In synchronous communication the client contacts a server and waits for a reply. The main advantage of this kind of communication is that, when things are working well, a request is answered with a timely response. Many times synchronous communication is obligatory or at least desirable. The application requires an immediate reply.

In asynchronous communication the client sends a request to the server and gets a response after some time, or receives no reply at all. In contrast with the synchronous communication mode, the client does not receive a reply as part of the process.

**The Wireless Application Protocol**
Wireless Application Protocol (WAP) is an open international standard for applications that use wireless communication, for example Internet access from a mobile phone. It uses existing digital wireless infrastructures (2G and 3G). A mobile phone device (or
The purpose of the system

other devices) calls up a WAP Gateway over a wireless link and send requests to it. The WAP Gateway is connected to the wired Internet.

Wireless Markup Language WML is the primary content format based on XML for devices that implement the WAP specification. The new version of WAP, WAP 2.0, is a re-engineering of WAP using XML. WAP 2.0 also has some new features. The most significant ones are [Sobat, 2004]:

1. It has a push\textsuperscript{12} model as well as a pull model.
2. It supports integration of telephony into applications.
3. It allows multimedia messaging.

The architecture of a WAP system is illustrated in Figure 13.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{wap_architecture.png}
\caption{The WAP - architecture}
\end{figure}

\textsuperscript{12} In the pull model, the client asks for a page and gets it. The push model supports data arriving without being asked for, such as a continuous feed of stock prices or traffic alerts.
6.3. Actors

Actors represent external entities that interact with the system. An actor can be human or an external system. In the next section for the t-info application, the t-info website and the t-info admin, all actors will be identified.

6.3.1. The t-info Application

In the t-info application there are two kinds of actors, the first one is a t-info user, the owner of the mobile phone. He uses the application to submit the t-info code, choose an item from the favorite list or download the new configuration file (favorite list) from the t-info server. The second actor is the t-info server. The t-info application sends requests to it and gets responses back. Figure 14 shows the t-info actors.

![Figure 14: the t-info application actors](image)

6.3.2. The t-info Website

The t-info website is another part of the t-info system. This part contains two actors: the first one is an unauthorized user. This user can only login into the website or visit public web pages. This kind of user will be called a *visitor*.

After logging into the website, the visitor will be changed into an authorized user. This kind of user will be called a *user*. Figure 15 shows the actors of the t-info website.

![Figure 15: t-info website actors](image)
6.3.3. The t-info Admin

The administrative tasks of the t-info system are done by the t-info Admin part, or backend. The same as the t-info website, this part has two actors: the first one is an unauthorized administrator. This actor can only login into the backend. This kind of user will be called a visitor.

If a visitor logs into the backend, he will be changed into an authorized user. This kind of user will be called an admin. The backend sends requests to it and gets responses back from the t-info server. Figure 16 shows the t-info backend actors.
6.4. The functional model: Use cases

A use case is a description of how end-users will use a system. It describes a task or a series of tasks that users will accomplish using the system, and includes the responses of the system to user actions. In the next sections, the use cases for the t-info application, the t-info website and the t-info admin will be identified.

6.4.1. The t-info Application

A t-info user uses his mobile phone to submit the t-info code to the t-info server and get information from the server. Beside this, the t-info user can also put a favorite list of his favorite t-info codes together and download it to his mobile phone. After this process he can select a t-info code from his favorite list, the selected item will be submitted to the server, and the returned information will be displayed on the mobile phone. These use cases are shown in Figure 17.

![Figure 17: t-info application use case](image)

An example of the use SubmitTinfoCode case description is given in Table 2.

<table>
<thead>
<tr>
<th>Use case name</th>
<th>SubmitTinfoCode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating actor</td>
<td>Initiated by t-info user (user)</td>
</tr>
<tr>
<td></td>
<td>Communicates with the server</td>
</tr>
<tr>
<td>Entry condition</td>
<td>The user starts the t-info application.</td>
</tr>
</tbody>
</table>
| Flow of events       | 1. The application responds by presenting a text field and an option menu (for Favorites, Download Config., etc).
|                      | 2. The user fills the blank text field with a t-info code and submits the form by pressing the “Send” button. |
|                      | 3. The server receives the request and retrieves the parameters from it. These parameters can be t-info code, the mobile phone identification number, the cellid, the request type etc. |
The purpose of the system

<table>
<thead>
<tr>
<th>Exit condition</th>
<th>Special requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>4. The server uses this information to retrieve the required information from a database or to engage other web-services. It sends the information as response back to the user.</td>
<td></td>
</tr>
<tr>
<td>5. The user receives the response from the server.</td>
<td></td>
</tr>
<tr>
<td>If the t-info code is not valid, the server sends an error message to user.</td>
<td></td>
</tr>
<tr>
<td>The text field accepts only digits.</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: an example of Use case description: SubmitTInfoCode

6.4.2. The t-info Website

The t-info website is mainly used for managing the favorite list and user’s setting. Before being able to do this process, a visitor has to be registered with the t-info website and logged into it.

Figure 18 shows the t-info website UML diagram. Note that user in not a subclass of visitor. This is because once visitor is logged in he becomes user and user cannot login for the second time.

Figure 18: t-info website UML use case diagram

And an example of the ChangeConfig use case description is listed in Table 3.

<table>
<thead>
<tr>
<th>Use case name</th>
<th>ChangeConfig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating actor</td>
<td>Initiated by t-info website user</td>
</tr>
<tr>
<td></td>
<td>Communicates with the website</td>
</tr>
</tbody>
</table>
The purpose of the system

**Entry condition**  
User selects “change configuration” item from the navigation menu.

**Flow of events**
1. Website receives the request and retrieves the parameters (user id and request type).
2. Website finds the configuration and sends it as response back to user.
3. The t-info website represents a form that contains options for adding new items to the favorite list and deleting items from the favorite list.
4. User adds new items from a default list to the configuration or deletes items from the configuration, and submits the form by pressing the “Submit” button.
5. Website gets the requests, retrieves the parameters (user id, list of items) and saves the changes. After this process it sends an acknowledgement to the user.

**Exit condition**
6. The t-info website shows the acknowledgement.

**Special requirements**

Table 3: an example of Use case description: ChangeConfig

6.4.3. The t-info Admin

An admin is responsible for managing the database. For example an admin can add a new t-info code into the database, or can change a t-info user. The same as the t-info website, the t-info Admin is protected from the unauthorized users.

Figure 19 shows t-info Admin UML use case diagram.

![Figure 19: t-info Admin UML use case diagram](image-url)
The purpose of the system

An example of the t-info admin use case is shown in Table 4.

<table>
<thead>
<tr>
<th>Use case name</th>
<th>AddCode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participating actor</td>
<td>Initiated by t-info admin</td>
</tr>
<tr>
<td></td>
<td>Communicates with the server</td>
</tr>
<tr>
<td>Entry condition</td>
<td>The admin selects the “add Code” item from the navigation menu.</td>
</tr>
<tr>
<td>Flow of events</td>
<td>1. The server gets the request and retrieves the parameters (request type).</td>
</tr>
<tr>
<td></td>
<td>2. The backend responses by presenting a form with text fields.</td>
</tr>
<tr>
<td></td>
<td>3. Admin fills the blank fields, such as t-info code, a title, multimedia file and information content (textual) of a new code and submits the form by clicking the “Save” button.</td>
</tr>
<tr>
<td></td>
<td>4. The server receives the request and retrieves the data from the request. The server saves the new code into the database and saves the multimedia file into file-system.</td>
</tr>
<tr>
<td>Exit condition</td>
<td>5. The t-info backend gets an acknowledgment.</td>
</tr>
<tr>
<td>Special requirements</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 4: an example of Use case description: AddCode
6.5. The analysis model: class diagram

6.5.1. The t-info application

Figure 20 shows the t-info application UML class diagram.
The purpose of the system

The t-info application is a standalone application that runs on a mobile phone. It has graphical user-interface for interaction with the user and it is also able to use the devices storage medium to store the favorite list. Below follows the description of classes.

**Identifying Entity Objects**
- **TInfoUser**: The t-info user that uses the mobile phone device. Users are identified by their mobile phone identification number.
- **ContentInformation**: This is a response from the t-info server. It can be either textual information or multimedia data, or both.
- **TInfoCode**: The object sent from the mobile phone to the server.
- **Favorite**: This is a favorite list with predefined t-info code.

**Identifying Boundary Objects**
- **AppForm**: The application form. This is presented to the user on his mobile phone device. The user uses this form to submit the t-info code, download a new favorite list or choose an item from its favorite list.
- **OptionMenu**: The interaction style that is used in the mobile phone applications is based on menus.

**Identifying Control Objects**
- **SubmitTInfoControl**: This object is responsible for establishing a communication with the server. It submits the t-info code to server and gets a response back. This object is created when the user wants to submit a t-info code.
- **SelectFavoriteControl**: This object is responsible for creating a connection with the server. Once the connection is established, it sends the t-info code to the server and gets a response back. This object is created when the user selects an item from the favorite list.
- **DownloadConfigControl**: This object is responsible for establishing a connection with the server. It downloads the favorite list from the server, parses it and saves it.
- **MainApp**: This object controls the whole process.

**6.5.2. The t-info website**

Figure 21 shows the t-info website UML class diagram.
The purpose of the system

The t-info website is a standalone web-application. It contains a graphical user-interface for user interaction and it also uses a database management system to store the data. The description of the classes is given below.

**Identifying Entity Objects**
- **TinfoUser**: this is an authorized t-info user. The user has to subscribe for a unique username and password.
- **Favorite**: it is a favorite list that can be defined by the user.
- **Setting**: this object contains all information about user’s mobile phone and user settings.
- **RegistrationData**: this object contains all information about the potential TinfoUser. The registration process is done by sending this object to the administration. Then the administrator adds the user into the database. After this process, the user will be notified.

**Identifying Boundary Objects**
- **LoginDialog**: this dialog contains text inputs for username and password of the user, and two buttons for submission and annulment.
- **ConfigView**: for viewing the favorite list to the users.
The purpose of the system

- **ConfigChange**: this form contains two lists. One list is used for default favorite items and the other list is used for user favorite list. The TinfoUser can add new items from the default list to his list, or delete items from his list.
- **SettingForm**: it has text inputs for changing the setting.
- **RegistrationForm**: this form contains text inputs for username, password and address information of a potential TinfoUser

**Identifying Control Objects**

- **LoginControl**: for authentication process; makes a connection with the database and checks the username and password
- **ViewConfigControl**: makes a connection with the database and retrieves the Favorite from the database.
- **ChangeConfigControl**: it is responsible for saving the changes into the database.
- **ChangesSettingControl**: this object is responsible for viewing the settings and saving the changes into the database.
- **RegistrationControl**: this object is responsible for sending the RegistrationData object to the admin by an email.

**6.5.3. The t-info admin**

The t-info admin UML class diagram is shown in Figure 22.
The purpose of the system

As the t-info website, this part is also a standalone web-application that contains a graphical user-interface and user a database management system. Below follows the description of classes.

**Identifying Entity Objects**
- **Admin**: this is an authorized t-info administrator. It is identified by its username that is unique in the whole system.
- **TinfoUser**: this object contains all information about the t-info user, such as, username, password and address information.
- **TinfoCode**: This object contains information about the t-info code and its associated information contents, such as, textual information, title and media file etc.
- **Stat**: this object contains the timestamp and IP-address of the visitors and users.

**Identifying Boundary Objects**
The purpose of the system

- **LoginDialog**: this dialog contains text inputs for username and password of the admin, and two buttons for submission and annulment.

- **WebPage**: the website form, also called the backend. This is presented to admin on the client terminal. The admin always interacts with this from. It also contains a menu for navigation.

- **NavigationMenu**: the admin uses this menu for navigation. It is always visible on the webpage.

- **ViewStat**: for viewing the statically information of the user to the admin.

- **UserView**: for showing the user list to the admin.

- **UserChange**: it is responsible for getting the user data by providing text fields to the admin. These fields will be filled with user values. The admin can change these values, and save the changes.

- **UserAdd**: it has text inputs for user attributes. It contains blank entries. The admin fills the blanks and saves the user into the database.

- **CodeView**: for viewing the list of t-info codes.

- **CodeChange**: it has text inputs for code attributes. These text inputs are filled with the old entries. The admin can change the values and saves the changes into the database.

- **CodeAdd**: the admin uses this object to fill the required data that is needed for adding a new TinfoCode.

**Identifying Control Objects**

- **LoginControl**: for authentication process, makes a connection with the database and check the username and password.

- **ViewUserControl**: this object is responsible for retrieving the data for the database and providing the data to the UserView boundary object.

- **ViewStatControl**: for ViewStat business logic, makes a connection with the database and retrieves the data.

- **ViewCodeControl**: for codeView business logic, makes a connection with the database and retrieves the data.

- **ChangeUserController**: it is responsible for saving the changes into the database.

- **ChangeCodeControl**: it is responsible for saving the changes into the database.

- **AddCodeControl**: it is responsible for add a new Code to the database.

- **AddUserControl**: it is responsible for adding a new user to the database.
System Design
7. System Design

7.1. Introduction
This chapter documents the system design. It describes design goals set by the project, the subsystem decomposition, the hardware/software mapping, the persistent data management, the access control, the global software control, and boundary conditions.

7.1.1. Design goals

The t-info Application
The t-info application runs on a mobile phone. The research [Sobat, 2004] has shown that a mobile phone device has a lot of limitations in term memory, connectivity, display and portability. During design, all these limitations need to be taken in to account. For example it is not permitted to ask users to upgrade the application after a small extension.

- **Response time** – After sending a request to the server, the user must get a textual response within 20 seconds. The latency of the response depends on the amount of information. A multimedia file requires more download time (at most 30 – 40 seconds).
- **Usability** – The application must be user-friendly. The user has to be able to get information in just few steps. The application should start by pressing a few buttons.
- **Portability** – The application must be compatible with as many mobile phones systems as possible. The application has also to be able to run on not-UMTS devices (with fewer features).
- **Robustness** – The system should be robust; it must be able to handle user input errors and corrupted data from the server.
- **Extendibility** – The application must be extendable. It should be possible to extend the application with new features dynamically, which mean that the extensions should be done during the run time.

The t-info website

- **Security** – The website must be protected from unauthorized users.
- **Usability** – The website should be user-friendly, that is, the user should be able to find his way in the website. This can be done by providing a good menu structure, consistent metaphors and standard and intuitive layout.

The t-info admin part

- **Security** – The backend must be protected from unauthorized administrators.
- **Usability** – The backend should be user-friendly, that is, the administrator should be able to find his way in the website. This can be done by providing a good menu structure, consistent metaphors and standard and intuitive layout.
7.2. Proposed software architecture

7.2.1. The t-info architecture
As it was mentioned in the previous chapters, the t-info application uses the HTTP protocol to send requests to a t-info server. On the server side, the t-info server will parse the request parameters, generate a response and send it back to the client. One of the main requirements of the t-info concept is the extendibility. For this reason it is chosen that the response from the server to the client will be an XML document.

The following scenario shows user who observes a t-info code, runs the t-info application on his mobile phone, enter the t-info code and submits. This scenario is shown Figure 23.

![Diagram of the t-info code submission process]

When the t-info server receives the request, it will parse the parameters, after some computational tasks it will retrieve the data from the database or invoke other web-services. Sometimes, the server may need additional information. For example, a t-info user starts a process that requires additional authentication. The server then sends a message to the t-info application requesting the user password. If the t-info application receives that message, it will show a password dialog to the user and requests the user to fill in his password. After filling the password, the user submits the additional parameters
to the server. At the end the server send the final message and the dialog communication will be finished.

All messages from the server to the t-info application are based on XML. The t-info application, on the other hand, parses the XML-document and visualizes the information to the user. These processes are shown in Figure 24.

![Diagram showing the interaction between request and response](image)

Figure 24: between request and response

If the response contains multimedia files, such as pictures or videos, the t-info application will send requests for these files to the server. After receiving a response from the server the t-info application displays the multimedia content to the user.

After receiving the XML-document from the server, the t-info application can either close the connection or keep it open and close it when the content is displayed to user. In the first case, if the XML-documents contains multimedia file, the application would use one HTTP connection for the XML-document and then one for each multimedia file. In the later case, the application can use one connection for XML-document and multimedia files.

In the chapter about the Hyper Text Transfer Protocol, the transfer time is defined by

\[ 2\rho + \tau \]

If the t-info application does not uses persistent connections, the total transfer time leads to

\[ (2\rho(n + 1)) + \tau = 2\rho n + 2\rho + \tau \]
where $n$ stands for the number of multi-media files.

With a persistent connection, this would be reduced to

$$2\rho + n\rho + \tau$$

Beside the information content associated with the t-info code, a user can also download his settings and favorites list from the t-info server to his mobile device. In this scenario, the user sends his IMEI (the mobile phone identification number) together with the request. The server retrieves the user’s settings from the database and generates an XML-document and sends this document back to the application. As in the first case, this scenario will be based on synchronous request processing.

Once the XML-document is received, the t-info application parses it and saves the new settings and favorites into the local database. The transfer time for this operation will be

$$2\rho + \tau$$

where the $\tau$ depends on the number of items in the favorite list.

### 7.2.2. The t-info Server architecture

The t-info server has a very important task. It is connected to the Internet, uses a database and provides services to the client (the t-info application). The server will accept requests from the client and will start a process. Some processes needs external (web) service, while other processes manipulate the internal database. The following steps will be taken:

1. Receive a request
2. Parse the request and retrieve the parameters
3. Validate the parameters
4. Retrieve the session from the database, if available.
5. Check if additional parameter is needed
   - If the process needs additional information from the user then it sends a message to the t-info application requesting for those information. Before sending the request, the server saves the session into the database and start from step 1.
6. Check whether an external web-service is needed
   - If an external web-service is needed, prepare the request and invoke the external service
7. Retrieve the data from the database if needed
8. Create a response and send it back to the client
The t-info Application
The t-info application can be decomposed into a Communication subsystem, implementing the communication between the mobile phone device and the server, a Storage subsystem, responsible for storage of data, UserInterface subsystem,
implementing the user interface for the t-info user, and Control subsystem, responsible for controlling the system.

![Subsystem Decomposition for the t-info Application](image)

The t-info application is based on closed layered architecture. In this architecture each layer can only depend on the layers immediately below it. Figure 27 shows the t-info subsystems with classes.

![t-info Application Subsystem and Classes](image)

**The t-info website and t-info Admin**
This section will discuss the other parts of the t-info system. It shows the details of the t-info website application and the t-info admin application. Each application is separated into 3 tiers. Each tier has its functionality and responsibility. These tiers are shown in Figure 28.

![Figure 28: Functional application tiers](image)

The client tier provides a way for users to interact with the application. For this interaction a web browser is used. Regardless of the type of client, the interaction includes submitting a request and receiving some type of response from the server.

All of the application logic is resided in the web tier. This tier is responsible for managing screen flow based on the application and user state, such sessions. The web tier is the glue that binds the client application to the database tier. Static web-pages, such as HTML, and dynamic pages, such as PHP, are resided in this tier.

The database tier contains data. It provides access to resources such as databases.

The decomposition for the t-info website and the t-info admin are shown in Figure 29 and in Figure 30, respectively.
7.2.4. Hardware/software mapping

The t-info application
All subsystems are purely logical and there are not physically separated. The platform to be used is a Java Virtual Machine. The t-info application uses packages that require floating point number and it should be able to use streams for showing multi-media contents. For this reason the JVM should be use with CLDC 1.1\(^{13}\) and MIDP 2.0\(^{14}\)[Sobat, 2004].

---

\(^{13}\) J2ME Connected Limited Device Configuration: The main goal of configurations is to define minimum requirements for memory, language support, virtual machine support, and runtime libraries.

\(^{14}\) Mobile Information Device Profile: The goal of a profile is to define APIs for devices that have similar uses.
The t-info website and the t-info admin
All subsystems will run on an Apache web-server with MySQL and PHP support. All subsystems are written in OO-PHP. Figure 31 shows this situation.

A client requests a page resource from a Web server, identified by its IP-address. The application server processes the page, which runs in a separate thread. During this process it may manipulate on the database and invoke some external web-services. Figure 32 illustrates the deployment of the t-info application, the t-info website and the t-info admin in one figure. As it illustrated, all parts communicate with the t-info server. This figure also shows the t-info website and the t-info admin as (web) applications that run on user’s web-browser. This is because the user always interacts with these parts through his web-browser.
Figure 33 shows the sequence diagram for handling the dynamic resource request.

7.2.5. Persistent data management

The t-info Application
J2ME and MIDP provide the Record Management System (RMS) for the persistent storage. An RMS database consists of a collection of records that remain persistent after the MIDlet (an application that runs on a mobile phone) closes. When the MIDlet is again invoked, it can retrieve data from the persistent record store.

There is one object that will be stored into the mobile phone device, the Favorite.

**Favorite attributes:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>This is the t-info code.</td>
<td>Integer</td>
</tr>
<tr>
<td>Title</td>
<td>This is a text that will appear on the mobile phone display.</td>
<td>String</td>
</tr>
</tbody>
</table>

**The t-info Website**

The t-info website application and the t-info admin application are connected to the database. The t-info website application is responsible for managing the user’s settings and favorite list. This section describes the t-info website entity objects.

The following entity objects will be stored into the database:

- TinfoUser
- Favorite
- Setting

**TinfoUser attributes:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>A unique name that identifies the t-info user.</td>
<td>String</td>
</tr>
<tr>
<td>Password</td>
<td>The password is used to log into the website.</td>
<td>String</td>
</tr>
<tr>
<td>AddressInfo</td>
<td>Address information of the t-info user. It is not required to store the specific detail of the user’s address.</td>
<td>String</td>
</tr>
</tbody>
</table>

The username is the primary key.

**Favorite attributes:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>The t-info code.</td>
<td>Integer</td>
</tr>
<tr>
<td>Title</td>
<td>Appears on the mobile phone display.</td>
<td>String</td>
</tr>
<tr>
<td>Userid</td>
<td>This is the foreign key that refers to the username of the tinfoUser.</td>
<td>String</td>
</tr>
</tbody>
</table>

The combination of the code and userid is the primary key.

**Setting attributes:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>settingId</td>
<td>A unique ID to identify the setting.</td>
<td>Integer</td>
</tr>
</tbody>
</table>
System Design

<table>
<thead>
<tr>
<th>InfoType</th>
<th>And identifier to identify the information content type. Such as, only textual information or multimedia information.</th>
<th>Integer</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMEI</td>
<td>Identifies the mobile phone of the used by the user</td>
<td>String</td>
</tr>
<tr>
<td>Userid</td>
<td>The foreign key that refers to the username of the user</td>
<td>String</td>
</tr>
</tbody>
</table>

The combination of the settingId and userid is the primary key.

The t-info Admin
The t-info admin is responsible for managing the t-info user and the t-info code. The following entity objects will be stored into the database:
- TinfoUser
- Code
- Stat
- Admin

TinfoUser attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>A unique name that identifies the t-info user.</td>
<td>String</td>
</tr>
<tr>
<td>Password</td>
<td>The password is used to log into the website.</td>
<td>String</td>
</tr>
<tr>
<td>AddressInfo</td>
<td>Address information of the t-info user. It is not required to store the specific detail of the user’s address.</td>
<td>String</td>
</tr>
</tbody>
</table>

Code attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>This is the t-info code.</td>
<td>Integer</td>
</tr>
<tr>
<td>Title</td>
<td>The title of the information. This text will be shown in the favorite list on the mobile phone display.</td>
<td>String</td>
</tr>
<tr>
<td>TextInfo</td>
<td>The textual information associated with the Code.</td>
<td>String</td>
</tr>
<tr>
<td>mmURL</td>
<td>This is the address where the multimedia content is located.</td>
<td>String</td>
</tr>
</tbody>
</table>

The code is the primary key.

Stat attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ipadd</td>
<td>The IP address of the user</td>
<td>String</td>
</tr>
<tr>
<td>Timestamp</td>
<td>The timestamp of the page hit</td>
<td>Long</td>
</tr>
</tbody>
</table>

The combination of the ipadd and timestamp is the primary keys.

Admin attributes
### Name | Description | Data type  
--- | --- | ---  
Username | A unique name that identifies the t-info admin. | String  
Password | It is used to log into the admin area. | String 

The username is the primary key.

#### 7.2.6. Access control and security

**The t-info Application and Server**

There is no security restriction for the t-info application. The application will start immediately without any authentication process. The server called by the mobile phone, where the t-info application is running, however needs to identify the mobile phone.

The CommunicationSubsystem is responsible for establishing a connection with the server and sending requests. Together with the request, it sends the Mobile Phone Identification for authentication process. The server uses this number to authenticate the t-info user and retrieves his setting and favorite list.

**The t-info Website**

The t-info website is able to responds to two kinds of actors, visitor and tinfoUser.

- The visitor – this actor can only view the public pages and can log into the website, after being registered by the admin.
- The tinfoUser – when the visitor is logged in, by submitting his username and password, it become a tinfoUser. This actor can manage his setting and favorite list.

**The t-info Admin**

- Admin – this actor must login before using the admin part. After logging in he can manage. He is capable of creating a new user, changing an existing user or deleting an existing user. And he can also manage the code and add a new code with its associated information content.

#### 7.2.7. Global software control

The t-info application, the t-info website and the t-info admin use an event-driven control. The system reacts on event from the t-info user.

#### 7.2.8. Boundary conditions

**The t-info Application**

The t-info application is implemented as a MIDlet. A MIDlet is a Java class that extends the abstract class `javax.microedition.midlet.MIDlet` and implements the `destroyApp`, `pauseApp`, and `startApp` methods. These are the same as start, stop,
and destroy methods of the java.applet.Applet class in J2SE. The MIDlet application manager calls the MIDlet’s methods to change the MIDlet’s state.

Initialization: When the t-info application is started, the program loads the favorite list from its flash memory, see chapter 7.2.5. After this process the user can either choose an item from the favorite list or submit a t-info code.

Shut down: The application can be terminated at any time by selecting the exit item from the option menu. Before termination, the application checks if it is connected to the server. If this is the case it closes the connection before being terminated.

Exceptions: 
An exception will occur, for example, when the amount of free memory is low or the connection with the server can not be established. These exceptions can be caught by the application manager and a proper message will be shown to the user.

The t-info Website and the t-info Admin
These two parts will run on an external web-server. The web-server provider will guarantee the availability. The provider is responsible for fixing the server failures.

Initialization: The t-info website and the t-info admin run on a dedicated or shared server. It is assumed that the server is always online and it is connected to the database.

Shut down: The server provider guarantees the availability. It is assumed that the server is always available and reachable.

Exceptions: 
User error – The system will give an informative feedback to the user.
Object Design
8. Object Design

8.1. Introduction
This section refines the analysis and system design models, identifies new objects, and closes the gap between the application objects and off-the-shelf components. The object design model can be partitioned into sets of classes such that they can be implemented by individual developers.
8.2. **Object design trade-offs**

[Sobat, 2004] The t-info application is an embedded application that runs on a mobile phone. Embedded devices are small and have some limitations. Most of embedded devices have a small amount of memory. Another concern for developers of mobile phone applications is the speed of the processor used in the device. Some consideration needs to be taken before coding an embedded application. In this part some of these considerations will be discussed.

**Move Computation to the Server**

If the application model is based on client/server architecture, it is better to avoid running computationally intensive tasks on the client (device). Instead, let the server computer run them. This process is similar to deploying a thin-client Web application where most the logic is in the web server, leaving the client (web browser) to do the user interface.

**Simplify the Application**

Make the application as simple as possible. The most obvious simplification is to remove unnecessary features. Removing code is the simplest way to reduce the size of an application. Another point is to reuse the user interface whenever possible, beside size reduction; it makes it easier for the user to learn the application. This step is done during the application design.

**Use Less Memory at Run time**

Most of embedded devices have limited memory. There are many ways to reduce the amount of runtime memory that Java applications use.

Objects have variable size. As a result, there is no way to determine the size of an object at compile time. When an object is created, the runtime environment allocates the object from the runtime memory heap. To reduce the number of objects that are allocated, consider using scalar types. For example: There are two ways to invoke the `setBounds` method of `java.awt.Component`.

1. `setBounds(Rectangle);`
2. `setBounds(int, int, int, int);`

The first one requires an object-allocation, while the second one takes scalar types as parameters. Consequently, if the second alternative is used, allocation will be avoided.

Java uses the garbage collector for searching through the runtime heap and reclaim unused memory. By setting the object references to `null`, it becomes easier for the garbage collector to find and reclaim unreferenced objects.

Using lazy instantiation is another way to reduce the memory usage. By initializing the objects as they are needed, the overall and peak memory usage will be reduced.
Java supports exceptions, which are extremely convenient. By avoiding exceptions, the size of the class files will be reduced. And also the number of objects that are allocated will be reduced.

**Performance tuning**

Another popular technique is to tune the code. There are several ways to reach this goal. One of them is to use local variables instead of class members. It is generally slower to access class members than to access local variables.

It is easy to build strings by concatenation. But the performance of this process is extremely poor. Consider the follow case:

```java
for( int i = 0 ; i < 1000 ; i++)
{
    str += "#";
}
```

In this case, the concatenation involves creating a new `StringBuffer` object, calling its `append` method, and then calling its `toString` method in order to get the final string. This process is repeated 1000 times.

The better solution is to handle the whole process manually, as it is shown in the following code fragment.

```java
StringBuffer strB = new StringBuffer();
for( int i = 0 ; i < 1000 ; i++)
{
    strB.append('# ');
}
String str = strB.toString();
```

In this case, the number of object creation is reduced extremely.

**8.2.1. Interface documentation guidelines**

The coding convention is based on Hungarian notation. Hungarian notation is a naming convention in computer programming, in which the name of an object indicates its type and intended use. For example in variable `tfTest` the prefix “tf” is used for `textfield`.

The following section, `packages`, describes the decomposition of subsystems into packages and the file organization of the code. Then in the chapter about class interfaces the classes and their public interfaces will be described.

---

15 During this process, the garbage might have trouble keeping up and collecting unreferenced objects.
8.3. Packages
This chapter describes the decomposition of subsystems into packages and the file organization of the code for the t-info application, the t-info website, and the t-info admin. It includes an overview of each package, its dependencies with other packages, and its expected usage.

The t-info Application

- **UserInterface** – This subsystem is responsible for displaying the t-info content to the user. It is also responsible for interaction with the t-info user.
- **Control** – This subsystem is responsible for controlling the whole process. It uses other subsystems to complete the task. Such as, downloading the favorite list from the server and storing it into the mobile phone.
- **Communication** – This subsystem is responsible for establishing a wireless connection (HTTP connection) with the server, sending the request to the server and getting a response from the server.
- **Storage** – This subsystem is responsible for all the persistent data in the t-info application.

![Diagram of t-info application packages and their dependences](image-url)
Figure 35: The t-info website packages and their dependences

- **GUI** – This package contains all dynamic and static web-pages that are used for interaction with the user.
- **ConfigControl** – This package is responsible for managing the configuration. Such as adding, removing and editing the favorite list.
- **LoginControl** – This package is responsible for the authentication process.
- **SettingControl** - This package is responsible for managing the settings. Such as, editing the user settings.
- **DataBase** – This package is responsible for managing the database.

Figure 36: The t-info Admin packages with their dependences

- **GUI** – This package contains all dynamic and static web-pages that are used for interaction with the user.
- **CodeControl** – This package is responsible for managing the configuration. Such as adding, removing and editing t-info codes.
- Object Design -

- **UserControl** – This package is responsible for managing the configuration. Such as adding, removing and editing t-info users.
- **LoginControl** – This package is responsible for the authentication process.
- **DataBase** – This package is responsible for managing the data base.
8.4. Class interfaces

This section describes the classes and their public interfaces. This includes an overview of each class, its dependencies with other classes and packages, its public attributes, operations, and the exceptions they can raise.

8.4.1. The t-info Application

MainApp:

<table>
<thead>
<tr>
<th>Methods / attributes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void start()</td>
<td>* Post: starts the application. Initialize the forms and screens.</td>
</tr>
<tr>
<td>void downloadConfig()</td>
<td>* Post: calls the downloadConfig method of Control class, which downloads the configuration from the server and saves it into the mobile phone.</td>
</tr>
<tr>
<td>void submit()</td>
<td>* Post: reads the content of the textfield. If the content of the textfield is not null, it call the submit function of the control class and passes the content of the textfield.</td>
</tr>
<tr>
<td>void selectItem()</td>
<td>* Post: the MainApp contains a menu with favorite list. This method will be invoked if the user selects an item for the favorite list menu. Then it calls the submit method of the control class and passes the item from the favorite list.</td>
</tr>
</tbody>
</table>

Control:

<table>
<thead>
<tr>
<th>Methods / attributes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void downloadConfig()</td>
<td>* Post: downloads the favorite list from the server and saves it into the storage.</td>
</tr>
<tr>
<td>Favorite[] getFavoriteList()</td>
<td>* Post: retrieves the Favorites from the storage and returns them as an array of Favorite.</td>
</tr>
<tr>
<td></td>
<td>* @return Array of Favorite</td>
</tr>
<tr>
<td>void submitCode(int code)</td>
<td>* Pre: 0 &lt; code</td>
</tr>
<tr>
<td></td>
<td>* Post: it creates parameters for an http connection. Then it calls the submit function of the Network class and passes the parameter.</td>
</tr>
<tr>
<td></td>
<td>@param code the t-info code</td>
</tr>
</tbody>
</table>

Network:
String server_address

* Represents the IP address of the t-info server. It can also be a domain name such as. www.t-info.com

ContentInformation submit(String param)

* Pre: param is not null
* Post: connects to the server, identified by server_address. Sends a request with parameters (param) to the server, receives a response. It parses the response (the xml document).
* @return The content information
* @param param HTTP parameter

Favorite[] downloadConfig()

* Post: connects to the server, identified by server_address,
* Downloads the favorite list.
* @return array of Favorite

<table>
<thead>
<tr>
<th>Desk:</th>
<th>Methods / attributes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String db_name</td>
<td>Represents the name of the database.</td>
<td></td>
</tr>
</tbody>
</table>
| void saveFavorite(Favorites[] list) | * Pre: list is not null
* Post: stores the Favorite list in the storage medium. |
| Favorites[] getFavoriteList() | * Post: retrieves the Favorite list from the storage medium. |

<table>
<thead>
<tr>
<th>ContentInformation:</th>
<th>Methods / attributes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String title</td>
<td>Represents the title of the content information.</td>
<td></td>
</tr>
<tr>
<td>String content</td>
<td>Represents the textual information of the content.</td>
<td></td>
</tr>
<tr>
<td>String mm_url</td>
<td>Represents the multi media URL of the content.</td>
<td></td>
</tr>
<tr>
<td>int tinfo_code</td>
<td>Represents the t-info code.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Favorite:</th>
<th>Methods / attributes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String title</td>
<td>Represents the title of the t-info code.</td>
<td></td>
</tr>
<tr>
<td>int tinfo_code</td>
<td>Represents the code of the t-info code.</td>
<td></td>
</tr>
</tbody>
</table>
Implementation
Implementation

9. Implementation

Previous chapters discussed the design of the t-info system. This chapter will put focus on the implementation of the designed system. First the used programming languages and tools will be discussed. Then, the t-info application implementation will be described.

9.1. Methods, Programming Language and Environment

The development of this project is based on the waterfall model. The first step was to understand the problem by reading the requirements and specifications. Then I started with a research, which helped me to get more insight in the problem and answer the questions. The requirements elicitation was the next step in this model, followed by Analysis, System design, object design, implementation and testing.

The t-info application is developed using the Java programming language (J2ME) under Microsoft Windows XP (SP2). The Java programming language is chosen because it is platform independent. Beside platform independency, Java provides useful packages for different goals. The software environment Eclipse is chosen as the development tool because it is an open source platform-independent software framework.

J2ME requires toolkits for compiling, pre-verifying and deploying a J2ME application. Sun Microsystems provides the “J2ME Wireless Toolkit” for this purpose. The t-info application was implemented using the J2ME Wireless Toolkit 2.2 and EclipseME, an Eclipse plugin for J2ME projects.

kXML was used for implementing the parsers for parsing the XML-documents. It is a small XML pull parser, specially designed for constrained environments such as Applets, Personal Java or MIDP devices.

The t-info application must be able to play the multimedia files, for this purpose the Mobile Media API (MMAPI) is used. It extends the functionality of the J2ME platform by providing audio, video and other time-based multimedia support to resource-constrained devices. The J2MEUnit is used for unit testing. Using J2MEUnit is very similar to the original JUnit framework. The differences between J2MEUnit and JUnit are mainly caused by the fact that on the micro edition platform of Java there is no reflection API available.

For all other parts of the t-info system the OO-PHP programming language was used. PHP is a server-side, cross-platform, HTML embedded scripting language. And it was also chosen for its flexibility, maintainability and readability. And finally, during design, the Poseidon [by Gentleware] has been used to make the UML diagrams.
9.1.1. Tools Evaluation

Most of problem encountered were result of lack of documentations. Most of packages used for the t-info application were not ready to use. This section will describe this problem and other problems encountered during the implementation phase.

When we write a Java application for a desktop computer, we start with writing a Java file. Then, this Java file will be compiled into the bytecodes, which results in class file. A Java Virtual Machine (JVM) will upload the class files and it starts with verifying, this task is done but the class-verifier. It checks if the loaded classes have a proper internal structure. If the class-verifier discovers a problem with a class file, it throws an exception.

Class-verification is an expensive and time-consuming operation. Limited devices, like mobile phones, lack the necessary amount of memory to verify the type-safety of Java bytecode on their own. The J2ME Wireless Toolkit provides a pre-verifier to partly verify the class-file offline, outside the devices. This means that a J2ME application requires an additional step. Beside compiling a Java file into class file, the class file must be pre-verified into a pre-verified class file. These steps are shown in Figure 37.

The packages used for the t-info application were only compiled but not pre-verified, which was also not described in their documentations. This caused a lot of time to find out the problem.

Another problem was related with the MMAPI (Mobile Media API). In its specification it was mentioned that it supports video and audio streaming. But this feature was not implemented. Current implementation uses a HTTP connection to connect to the server and download the media file. This method saves the file into device storage before being able to show it.
9.2. The implementation of the t-info application

Paragraph 7.2.1 described the t-info architecture. It showed an overview of message exchanges between the t-info application and the t-info system. This paragraph describes the t-info application and the t-info server implementation and communication protocol between both t-info parts.

The t-info application starts the communication by sending an HTTP GET request to the t-info server. Then the t-info server checks if it needs additional information from the t-info application. If it needs additional information then it will send a request\(^{16}\) for that information. This request is based on an XML document. The t-info application then parses the request and shows the next screen to the user. The code fragment below shows an example of a request from the t-info server to the t-info application.

```
1<?xml version="1.0"?>
2. <ContentInformation>
3.  <type>textfieldpassword</type>
4.  <label>Your password please:</label>
5.  <sessionid>222</sessionid>
6.  <values>
```

\(^{16}\) Actually a response
This will result in a password dialog.

Your password please:

********

Submit
Conclusion
And
Evaluation
10. Conclusion and Evaluation

10.1. Introduction

The goal of this thesis was to design and implement a working prototype of the t-info system. The t-info system uses a mobile application, which establishes a wireless connection with the t-info server and uses its services. Before starting with design and implementation, this thesis was started with a research. The goal of this research was to give an overview of possible problems and to answer the questions.

After the research, the second phase of the project, the design and implementation phase, has been started. The goal of this phase was to implement a working prototype of the system. Paragraph 10.2 gives an overview of questions and answers, then the evaluation will discussed in paragraph 10.3.

10.2. Conclustions

How to write a program for a mobile phone?
The implementation of an application depends on the platform. If a mobile phone doesn’t have an operating system then application should be written in machine codes, or there must be compiler that compiles a high-level language to target machine code. Writing an application for a mobile phone without an operating system requires a lot of work. The programmer should take care of all resources allocation, memory usage, concurrency issues and more. On the other hand, if a mobile phone contains an operating system then a programmer can use a high-level programming language to implement an application and don’t have to take care about major issues, such as concurrency issues, allocation issues, et cetera.

During my research, I have investigated on what kind of operating systems are used in a mobile phone and what are their supports. In general, operating systems are developed with special purpose. Some operating systems are to provide convenience while some other operation systems are made for efficiency. From programmers’ point of view, operating systems provide API and system calls, which can be used to for their targets.

From the user point of view, mobile phone operating systems are slightly different from Desktop computer operating system. But internally they are largely different. A mobile phone operating systems needs to be smaller than PC operating systems. Beside this, they have a lot of work. They differ in:

<table>
<thead>
<tr>
<th></th>
<th>Mobile phone OS</th>
<th>Desktop OS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Size</strong></td>
<td>A few KB’s</td>
<td>MB</td>
</tr>
<tr>
<td><strong>Reboot</strong></td>
<td>Should never reboot</td>
<td>May reboot</td>
</tr>
<tr>
<td><strong>Memory management</strong></td>
<td>Very effective</td>
<td>Effective</td>
</tr>
<tr>
<td><strong>Power consumption</strong></td>
<td>Very low</td>
<td>No constraint</td>
</tr>
</tbody>
</table>
Conclusion and Evaluation

<table>
<thead>
<tr>
<th>Connectivity</th>
<th>Needs to deal with patchy connectivity</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>History</td>
<td>New</td>
<td>Old</td>
</tr>
</tbody>
</table>

Since mobile phones contain an operating system, one can use a high-level programming language to implement an application. Implementing an application for mobile phones is different from the desktop applications. Mobile phones have limited processor speed and available memory when compared to a desktop or server computer. These limitations require us to change the way in which we code our applications.

**Which programming language suits best?**
During my research I have considered that an operating system supports different kind of programming languages. In fact, it is able to run machine code. All languages that are compiled to machine code or where its virtual machine code can be compiled into machine code can be uploaded into the device and might run. But the choice of a programming language is very important. The two powerful programming languages are the Java programming language and the C++ programming language. The Java programming language provides a garbage collector, which make it easier and safer to write programs. On the other hand, C++ is more powerful and fast but very difficult to manage and often very risky to use.

C++ object programs are not portable because they are always translated into the machine language for a specific machine. Java is portable but some operating systems provide their own package. If a program uses these packages, the portability character is not valid anymore. Portability is extremely important and it is one of the requirements.

From the requirements of the t-info it became clear that portability is very important. For this reason the Java programming language suits best for this project.

**How to connect to the Internet with a mobile phone application?**
Connectivity is very important for the t-info system. The client, the t-info application, establishes a wireless connection with the t-info server and uses its web-services. The research has shown that the J2ME CLDC is a configuration for connected devices and it provides the Generic Connection Framework. This framework defines functions without dependency on specific physical features or capabilities of a device.

**Is it possible to use location-based services?**
Location-based services are an extension to the t-info concept. The goal of this extension is to provide the geographic location of the device to the server by sending the coordination as request parameter. Java provides an optional package that enables mobile location-based applications for resource limited devices. It defines a generic API that produces information about the present geographic location of the device to Java application. On way to get the device’s coordination is to use a GPS module. But this contrasts with the portability requirement, because not all devices support a GPS module.
Another way to get the device’s coordination is to request the cell id from the base-station. But it depends on the network provider to provide the cell id to the user. Because of the privacy and security issues, it is not possible to get the cell id of a base-station in the Netherlands. For this reason, the location-based service is not implemented in this project. But the design gives enough room for future extension.

**How to handle an xml document in a mobile phone?**
The t-info application sends a request to the server and gets an xml document as response back. Java provides an optional package for parsing the xml document. This package is called the kXML. It provides an XML pull parser and writer for all Java platforms, including the J2ME.

**How the t-info application can be made dynamic?**
The dynamic aspect of an application is very important. Once the t-info application is installed in a mobile phone, there must be some way to add extra features during runtime.

This problem has been solved by using dialog messaging during the communication between the server and the t-info application. The t-info application initializes the dialog by sending a request to the server. The server then sends a response and if it needs more information from the client, it makes it clear in the response. Then the t-info application sends the additional information by sending a request to the server. This process continues until the server gets all information and sends the final dialog.

For example, if a service requires a password for extra authentication, it sends a message (response) to the t-info application. The t-info application parses the response and shows the password dialog to the use.

**User-friendly**
From the t-info concept it becomes clear that the application must be user-friendly. The ideal situation is that the application starts when the user press on a button. But this requirement is almost impossible to meet. For this goal, another application is needed to hook the key events and run our application. But this technique is not supported by current mobile phone operating systems.

**Which issues are not solved in this thesis?**
The security issues haven’t been solved in this project. Some common security issues are data encryption and strong authentication, which are not implemented in this project.
10.3. Evaluations
At the beginning of the project, there were a lot of questions and uncertainties. The research assignment helped to answer the questions. The objective was to implement the t-info system. During the project, a prototype has been implemented that met the major part of the requirements.

The problems
The major problem during the design and implementation of this project was the novelty of the mobile phone technology. Not implemented specifications and incomplete documentation caused a lot of problems and frustration.

The tools and environments
During the research assignment it became clear that J2ME was the right language to write an application for a mobile phone. Java provides a lot of useful packages, such as kXML to parse the xml document in a mobile phone, to write the t-info application prototype. For this project I used the Eclipse environment with EclipseME and J2MEUnit plugins (see Appendix B) for J2ME projects. I used EclipseME to compile, pre-verify and run a J2ME application and J2MEUnit was used for unit testing.

The other parts of the t-info system were implemented using the OO-PHP programming language.

The future aspects of the t-info concept
The t-info concept will be used a lot in the future. Mobile phone technology growth was very fast in the past decades. The telephone, entertainment, and computer industries have all been specialized in digital technologies and are rapidly converging. Besides calling, nowadays, people can use a mobile phone to surf on the Internet, to play music, and more. The t-info concept can be used in different areas and for different purposes. An article in EMERCE describes that BBC was planned to implement a similar concept. See Appendix A.
11. Appendix A(BBC: programma’s opnemen met mobieltje)

6 mei 2005 - Door Tjeerd Wiersma (source: http://www.emerce.nl)

De mobiele strategie van de BBC begint zich steeds verder te ontvouwen. Zo gaat de BBC kijkers nog dit jaar de mogelijkheid geven om programma's op te nemen met behulp van hun mobiele telefoon. Dat kan eind 2005 met behulp van een persoonlijke videorecorder (pvr) of via de pc, zo schrijft The Guardian.

In de Verenigde Staten is het al langer mogelijk om programma's via internet op te laten nemen op persoonlijke videorecorders zoals TiVo. In het Verenigd Koninkrijk biedt het commerciële televisiestation Sky sinds kort een soortgelijke dienst. Gebruikers van Sky+ kunnen hun persoonlijke videorecorders namelijk ook op afstand in werking zetten. Dat kan via internet, maar ook met behulp een sms'je. Sky zegt tegen The Guardian (registratie verplicht) dat er nog niet is gesproken over een eventuele samenwerking met de BBC op dat vlak.

Angel Gambino, business development manager van de BBC, zegt dat de ambitieuze mobiele plannen van de BBC er voor zullen zorgen dat een mobiele telefoon zowel zal gaan fungeren als een draagbare tv als een afstandsbediening. "We zijn ook van plan een mobiele browser te maken waarmee je toegang krijgt tot alle BBC content van radio tot videobeelden. Het is nu allemaal nog erg gericht op het vertonen van fragmenten, maar je zal zien dat er uiteindelijk hele programmas op het mobieltje zullen worden uitgezonden," zegt Gambino tegen The Guardian.

12. Appendix B (Tests)

12.1. Introduction
Software testing is a process used to identify the correctness, completeness and quality of developed computer software. During this process, the differences between the specified and observed behavior is detected. In this document we follow these following test activities:

- Unit testing – unit test focuses on the building blocks of the software system, that is, objects and subsystems.
- Integration testing – this test finds faults by integrating several components together.
- System testing – system test focuses on the complete system, its functional and nonfunctional requirements, and its target environment.

12.2. Unit Testing
A unit test examines the behavior of a distinct unit of work. Within a Java application, the “distinct unit of work” is often (but not always) a single method. By contrast, integration tests and acceptance tests examine how various components interact. A unit of work is a task that is not directly dependent on the completion of any other task.

For this process the automated tests is used. In this kind of test, a component cannot be proven to work until it passes a comprehensive series of tests. JUnit is used for propose. It is a unit test framework for Java. Since the t-info application is a J2ME application, the J2MEUnit is used instead of JUnit. Using J2MEUnit is very similar to the original JUnit framework. The differences between J2MEUnit and JUnit are mainly caused by the fact that on the micro edition platform of Java there is no reflection API available. That means that all the features of JUnit that use reflection to build and run test suites automatically can't be implemented as easily in the J2ME environment. J2MEUnit tries to alleviate this by providing some new mechanisms or by implementing them differently.

For debugging the following code fragment will be used:

```java
/**
 * This static method will be used to display trace points.
 * At the end it wil be removed.
 * @param o
 */
public static void debug(Object o) {
   System.out.println("DEBUG : " + o);
}
```
### 12.3. Test cases

<table>
<thead>
<tr>
<th><strong>Submit t-info code (correct code)</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>A t-info code</td>
</tr>
<tr>
<td><strong>Expected output</strong></td>
<td>Tries to connect to the server. If there is a connection, it connects to the server and gets the response from the server. It is an XML file. The system should parse the XML file and display it on the display</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Submit t-info code (incorrect code)</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>Some number that is not t-info code</td>
</tr>
<tr>
<td><strong>Expected output</strong></td>
<td>Tries to connect to the server. If there is a connection, it connects to the server and gets the response from the server. It is an XML file. Since the code doesn’t exits in the database, the server returns a default error message. The mobile phone should parse the XML file and display it on the display</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Download Favorites</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>Incorrect input is not possible. The user selects the download config item from the option menu.</td>
</tr>
<tr>
<td><strong>Expected output</strong></td>
<td>Tries to connect to the server. If there is a connection, it connects to the server and gets the response from the server. It is an XML file. The system should parse the XML file and store the data into the mobile phone.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Select an item from favorite list</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>Incorrect input is not possible. The user selects an item from the favorite list.</td>
</tr>
<tr>
<td><strong>Expected output</strong></td>
<td>Tries to connect to the server. If there is a connection, it connects to the server and gets the response from the server. It is an XML file. The system should parse the XML file and display it on the screen.</td>
</tr>
</tbody>
</table>
13. Reference:

13.1. Books

- M.B.Sobat, *The killer app of UMTS (Mobile devices) Research*, 2004
- Symbian white paper, *Why is a different operating system needed*,
  www.symbian.com, 4 Oct 2003
- Richard Harrison, *Symbian OS C++ for Mobile Phones*, John Wiley & Sons Ltd, 2004
- JSR-179 Expert Group, *Location API for J2ME Version 1.0*

13.2. Websites:

- 3GPP Home page, [http://www.3gpp.org/](http://www.3gpp.org/)
References:

- The Java Community Process(SM) Program, [www.jcp.org](http://www.jcp.org)
- Forum Nokia, [www.forum.nokia.com](http://www.forum.nokia.com)
- Symbian, [www.symbian.com](http://www.symbian.com)
- Java Performance Tuning, [www.javaperformancetuning.com](http://www.javaperformancetuning.com)
14. Index

.A
.NET Compact Framework, 39

.active objects, 28

.analysis, 22

.asynchronous, 48

.B
.backend, 24

.C
.Configurations, 32, 34

.Connected Device Configuration

.CDC, 34

.Connected Limited Device Configuration

.CLDC, 34

.E
.event-driven control, 75

.F
.First Generation Mobile phones

.1G, 12

.H
.HTTP 1.0, 46

.HTTP 1.1, 46

.J
.Java 2 Micro Edition

.J2ME, 32

.Java Virtual Machine

.JVM, 28

.K
.KVM

.Kuauï VM, 33

.M
.methods, 46

.Mobile Information Device Profile

.MIDP, 37

.Mobile Information Devices

.MIDP, 37

.multithreading, 28

.O
.on closed layered architecture, 68

.P
.pipeline requests, 46

.pre-emptive multitasking, 28

.Profiles, 32, 36

.R
.Record Management System

.RMS, 73

.requirements elicitation, 22

.S
.Second Generation mobile phones

.2G, 12

.Symbian OS

.Symbian, 28

.synchronous, 48

.system specification, 22

.T
.The Hypertext Transfer Protocol

.HTTP, 46

.the waterfall model, 87

.Thread Local Storage

.TLS, 39

.U
.Universal Mobile Telecommunication System

.UMTS, 13

.Use Case

.AddCode, 55

.ChangeConfig, 53
<table>
<thead>
<tr>
<th>W</th>
<th>Windows CE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WinCE, 29</td>
</tr>
<tr>
<td>Wireless Application Protocol</td>
<td>WAP, 48</td>
</tr>
<tr>
<td>Wireless Markup Language</td>
<td>WML, 49</td>
</tr>
</tbody>
</table>