Ubimedia Bracelet

Students:
Jelle Bockstal
Martin Keereman

Supervisor:
Jesper Svensson
1. Preface

Being informatics students at polytechnique ‘Hogeschool Gent’, it was a unique experience to be allowed to write our thesis at Halmstad University, Sweden. During our stay here we developed a multimedia-application ‘Bracelet’, part of a research project called Ubimedia.

Ubimedia is a project started by Media-IT, a team consisting of informatics teachers and researchers, who try to implement multimedia in a ubiquitous manner. Our project consisted of developing a prototype of a user interface for a digital bracelet which can be used to make daily life more easy. With this bracelet it will be possible to read e-newspapers, make phone calls, etc…. Media-IT will introduce this prototype afterwards to several newspaper publishers who are partners in research projects.

Making a thesis is the last step before graduating. It is an excellent opportunity to convert theoretical knowledge into practical objectives after three years of study. It is a pleasure to share knowledge and experiences with other informatics. Therefore it seems us suitable to announce a word of thanks to those by which it was possible to make this thesis.

First of all we would like to thank our internship supervisor, mister Jesper Svensson for his excellent and professional guidance. He let us peek into the practical world of development and gave us the opportunity to participate in the multimedia-applications of tomorrow. Mr. Svensson was open for our new ideas, opinions and our share. We also would like to thank the Media-IT program manager Carina Ihlström Eriksson. She gave us the possibility to work at Media-IT and take part in this project.

Finally we want to thank the general staff of Halmstad University for solving all our administrative problems. They made it possible for us to have a pleasant stay in Sweden.

Thanks to these people we can look back to informative, enjoyable and unforgettable student days.

Halmstad, Halland, Sweden, May 2008
## 2. Table of contents

1. Preface ..................................................................................................................... 1  
2. Table of contents ..................................................................................................... 1  
3. Summary ................................................................................................................. 3  
4. Media-IT ................................................................................................................. 6  
5. Ubimedia .................................................................................................................. 7  
   5.1 Initial scenario ..................................................................................................... 7  
6. Software Development Method .............................................................................. 8  
   6.1 Plan-driven methods ......................................................................................... 9  
   6.2 Agile methods .................................................................................................... 9  
      6.2.1 What are agile methods ............................................................................ 9  
      6.2.2 Suitability of agile methods .................................................................... 9  
      6.2.3 Agile manifesto ....................................................................................... 10  
6.3 Extreme Programming ......................................................................................... 10  
   6.3.1 What is Extreme Programming .................................................................. 10  
   6.3.2 XP values .................................................................................................... 10  
   6.3.3 Twelve steps .............................................................................................. 11  
6.4 The approach on the bracelet ............................................................................. 13  
   6.4.1 Plan-driven Methods .................................................................................. 13  
   6.4.2 Agile Methods ............................................................................................ 13  
   6.4.3 Extreme Programming .............................................................................. 14  
7. Design ...................................................................................................................... 15  
   7.1 Requirements Engineering ............................................................................. 15  
      7.1.1 Original requirements ............................................................................ 15  
         7.1.1.1 Functional Requirements .................................................................. 15  
         7.1.1.2 System Requirements ...................................................................... 16  
         7.1.1.3 Data Requirements .......................................................................... 16  
         7.1.1.4 Non-functional Requirements ............................................................ 16  
      7.1.1.5 User Requirements ............................................................................ 17  
6.2 User Interface Design ......................................................................................... 18  
7.3 Prototyping ........................................................................................................... 19  
   7.3.1 Low-fidelity prototype ................................................................................ 19  
7.4 Platforms ............................................................................................................. 20  
   7.4.1 PDA / Pocket PC ....................................................................................... 20  
   7.4.2 Desktop / Laptop ....................................................................................... 20  
   7.4.3 iPhone / iPod Touch .................................................................................. 20  
   7.4.4 Tablet PC ................................................................................................... 21  
7.5 Programming Tools ............................................................................................. 22  
7.6 Final requirements ............................................................................................... 24  
   7.6.1 Functional Requirements ......................................................................... 24  
   7.6.2 System Requirements ............................................................................... 24  
7.7 High-fidelity prototype ....................................................................................... 25  
   7.7.1 Persona’s ................................................................................................... 25
7.7.2 Demonstration........................................................................................................................................... 25
7.7.2.1 Main menu ............................................................................................................................................. 25
7.7.2.2 Submenu................................................................................................................................................... 26
7.7.2.3 Content menu ........................................................................................................................................ 26
7.7.2.4 Communication menu ....................................................................................................................... 26
7.8 User evaluation ............................................................................................................................................... 27
8. Conclusion ..................................................................................................................................................... 28
9. References ...................................................................................................................................................... 29
10. Appendix ..................................................................................................................................................... 30
3. Summary

Our task, developing a prototype of a futuristic bracelet, is part of the research project named Ubimedia (Designing Ubiquitous Media Services by Action Research). Ubimedia is being lead by Media-IT, the research group of the Information Science, Computer and Electrical Engineering department of Halmstad University. This project, which started on 1 July 2006, has been accepted 2.490.000 SEK by the KK-Foundation and will end 31 May 2008. The main purpose of this project is designing Ubiquitous media-applications that can be accessed everywhere and at anytime. Ubimedia works together with a newsgroup which involves cooperation with the largest newspapers in the world.

The current state of the project requires the development of two software applications. Our part consists of making a user interface on a small touch screen, which will simulate the prototype of the bracelet. The other part, executed by our fellow Belgian colleagues Kevin Geerinck and Leen Dekeukeleire, comprises the programming of the desktop-application. In the end, these applications will be combined and made public.

The main objective of our project is to write an application which will serve as an operating system for a digital bracelet. With this bracelet the user will be able to control all hardware systems surrounding him using a Bluetooth-, wireless Internet or Mobile phone connection. It’s also possible to view the headers of a certain article online. If the user wants to read one of the related articles, he can connect to an external screen. Besides that, the bracelet has other functions which will assist the user in making choices concerning home & family, news, entertainment, health & food and travel.

Although the application won’t be realized within the next ten years, developing this prototype on a tablet pc will give the user a good idea of how it will look in the future.

This paper contains the research, documentation and development of the project we made during our stay in Sweden.
4. Media-IT

The Media IT Group at Halmstad University was founded in 2004 and is interested in the design of usable IT for different future media genres. Media IT as a research program focuses on three important changes in society:

- The development of new information technologies
- Alterations in media consumption behavior
- The ongoing convergence between different types of media

Media-IT is in command of Carina Ihlström Eriksson who is the program manager. Besides her, the staff consists of Phd. students and research assistants who participate in research matters. In addition, Media-IT works together with students who help with projects.
5. Ubimedia

5.1 Initial scenario

The bracelet is developed within the research project called Ubimedia (Designing Ubiquitous Media Services through Action Research).

The application we created needs to be installed on a digital bracelet. This device will become a must-have for people who like to leave the controls of their personal life over to a machine. The bracelet will allow the users to look up recipes, travels, news and other different sorts of information in which they are interested. The user’s interests are being determined by the artificial intelligence of the machine. Besides that the bracelet allows the users to share their interests with friends, relatives and other contacts. Finally the bracelet also works as a personal processing unit which has the possibility to access external screens.

The main idea and the conceptual design were mostly elaborated by a design group, newspaper designers and people from Media-IT. The design group consists of proficient Photoshop designers as well as people who have many years of experience in designing newspaper layouts.

Ubimedia is developed together with the following partners: Tidningsutgivarna, Aftonbladet, Expressen, Dagens Nyheter, Göteborgs-Posten, Nerikes Allehanda, Norrköpings Tidningar, Sundsvalls Tidning, Stampen AB, Sydsvenskan and Östgöta Correspondenten.
Software engineering is more than the creation of software that consists out of thousands of lines of code. According to Sommerville (2001), software engineering is a discipline which is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use.

Software development is the process of developing software through successive phases in an orderly way. Rehman and Paul (2002) say these are the typical stages of software development:

- Identification of required software
- Analysis of the software requirements
- Detailed specification of the software requirements
- Software design
- Programming
- Testing
- Maintenance

A software project mustn’t be completed without a proper study as it will most likely result in lack of quality of the final product. Software development methods are viable and therefore widely used as a guide throughout a software development project. The number of different models and methods is fairly large and each of them is used under different circumstances.

In the next chapter we will specify some of the candidate methods for our project.
6.1 Plan-driven methods

The waterfall model is probably the best known plan-driven method. It is a sequential software development model in which development is seen as flowing steadily downwards through the phases of requirements analysis, design, implementation, testing, integration and maintenance (Royce, 1970). The term ‘waterfall’ is derived from the way the phases follow each other in a chronological way, without the opportunity of turning back.

Thus the waterfall model maintains that one should move to a phase only when its preceding phase is completed and perfected. This model is inflexible and should therefore only be used when the requirements are set in stone at the beginning of the process (Sommerville, 2001).

6.2 Agile methods

6.2.1 What are agile methods

Agile methods are sometimes characterized as being the total opposite of “plan-driven” methodologies. A more accurate description is to say that these methods are “adaptive”. They focus on adapting quickly to changes. When the needs of a project change, the planning changes as well. An adaptive team has difficulty describing exactly what will happen in the future. The further away a date is, the more vague the definition will be (Boehm & Turner, 2004).

6.2.2 Suitability of agile methods

The suitability of agile methods differ from each project. From a product perspective, agile methods are more suitable when requirements are flexible and rapidly changing. However, they are less suitable for systems that have high reliability and safety requirements.

Boehm and Turner (2004) suggest that risk analysis should be used to choose between adaptive ("agile") and predictive ("plan-driven") methods. Each side of the continuum has its own home ground.

<table>
<thead>
<tr>
<th>Agile home ground</th>
<th>Plan-driven home ground</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low criticality</td>
<td>High criticality</td>
</tr>
<tr>
<td>Senior developers</td>
<td>Junior developers</td>
</tr>
<tr>
<td>Requirements change very often</td>
<td>Requirements don't change too often</td>
</tr>
<tr>
<td>Small number of developers</td>
<td>Large number of developers</td>
</tr>
<tr>
<td>Culture that thrives on chaos</td>
<td>Culture that demands order</td>
</tr>
</tbody>
</table>
6.2.3 Agile manifesto

All agile methods, also known as principles are articulated in what is called the agile Manifesto. This manifesto contains four core values:

1. Individuals and interactions over process and tools.
2. Working software over comprehensive documentation.
3. Customer collaboration over contract negotiation.
4. Responding to change over following a plan.

The first recognizes that individual creativity and group collaboration are more effective than following a prescriptive methodology. The second value recognizes that software is code, not the accompanying documentation. The third recognizes that a good relationship between the clients and the developers is more important than arguing about contracts. The fourth value prioritizes users changing needs rather than adhering to some meaningless inflexible plan (Bell, 2005).

6.3 Extreme Programming

6.3.1 What is Extreme Programming

Perhaps the best-known of the agile methods, Extreme Programming is a software development discipline which main aim is to reduce the cost of change. In traditional system development methods the requirements for the system are determined at the beginning of the development project and often fixed from that point. This means the cost of changing the requirements at a later stage will be high (Bell, 2005).

XP sets out to reduce the cost of change by being more flexible with respect to changes. In XP, the client is indispensable during the development of the project. There has to be a constant communication between the developers and the customer. That way errors are discovered faster and more easy and so the final program will suit the demands of the client.

6.3.2 XP values

Extreme programming is based on four principles:
1. Communication: According to Beck (1999), Extreme Programming favours simple design and frequent verbal communication and feedback. Maximizing communication between developers and between the clients and the team is vital in any project.

2. Simplicity: XP encourages software that has a simple structure, but realizes that achieving simplicity is not easy (Bell, 2005).

3. Feedback: The next principle is closely related to communication and simplicity. Feedback is about obtaining reliable information about the state of the software as it’s being developed. That way, problems can be avoided. To quote Kent Beck (1999), “Optimism is an occupational hazard of programming, feedback is the treatment.”

4. Courage: Courage is probably the most surprising value. It means that the developers must have the courage to throw away code, or even re-design large parts of the architecture, if problems arise. This is dramatically different from the common approach, which attempts to patch up software when it demonstrates faults, minor or serious (Bell, 2005).

6.3.3 Twelve steps

Extreme Programming encourages starting with the simplest solution. Extra functionality can be added later. The focus lies in designing and coding for the needs of today instead of those of the future. Next to the four core values XP uses a combination of 12 techniques:

1. Planning
2. Small releases
3. System Metaphor
4. Simple Design
5. Continuous Testing
6. Refactoring
7. Pair programming
8. Collective ownership
9. Continuous integration
10. 40-hour work week
11. On-site customer
12. Coding standards
XP states that it is undesirable to use one of the techniques on its own, because they complement each other. A weakness in one practice is compensated by strength in another (Bell, 2005).
6.4 The approach on the bracelet

6.4.1 Plan-driven Methods

The Waterfall Model as well as other plan-driven methods are not that suitable for this project. Because of the inflexibility of the model, it can’t be used with changing customer requirements (Sommerville, 2001). The Waterfall Model needs well-established requirements at the beginning of a project. Due to the fact that the bracelet is likely to be undergoing some customer requirement changes this model is not a viable option.

6.4.2 Agile Methods

As mentioned before the prototype of the bracelet is likely to have constantly changing requirements. Since agile methods adapt quickly to changes (Boehm and Turner, 2004), they are the cut-out approach for this project. Because it’s a prototype, parts of the project may change during its development. As stated before, the four core principles of agile methods are:

1. Individuals and interactions over process and tools.
2. Working software over comprehensive documentation.
3. Customer collaboration over contract negotiation.
4. Responding to change over following a plan.

The first core principle might be the most crucial one in our project. We are constantly motivated to use our own creativity and ideas. Thus become the methodologies less important since we can follow our own path. Nevertheless is a little guidance necessary in succeeding a project of this matter.

The second core value states that code is more important than its documentation. This project demands a working prototype, so working software is the primary objective. However, the code has to be written in a way that it might be reused at a later time and added documentation might simplify this process.

The third principle is also a crucial one. The future public has to be constantly involved in order for us to develop a working prototype with a customer-friendly look and feel. Frequent user test are the perfect way to obtain constant feedback of the user’s needs.

The fourth and last core principle is as important as the first one. Since requirements and parts of the project are likely to change, so must the developers. The demands of the public are more vital than following a plan.
6.4.3 Extreme Programming

Extreme Programming, the best known agile method, is very powerful but must be used correctly. It is more a series of principles for developing software rapidly than a step-by-step methodology (Avison and Fitzgerald, 2003). Jeffries (2001) points out that Extreme Programming is a discipline of software development with values of simplicity, communication, feedback and courage, four core values on which XP is based.

Throughout the project, these four principles will be used. On communication, the approach stresses the role of teamwork. There has to be an open and honest communication between the developers and the customers. It is very important to gain concrete and rapid feedback from continuous testing. These tests will give us the feedback needed to make the proper modifications to the project. Customer requirement changes are likely to be found during these tests. Whereas simplicity is hard to achieve, we will try to keep the structure as easy as possible. As for courage, it will be necessary when drastic changes have to be made due to changing requirements.

XP also contains 12 steps which must be followed in order for the project to succeed. Whereas the main idea of XP fits the project, some of these techniques not always do. There will be frequent but not continuous testing due to the lack of an on-site-customer. As for the 40-hours-week, we will work longer if this is required.

XP is very suitable for this project. Unlike many other methodologies, documentation is not a prime concern, delivering software is (Avison and Fitzgerald, 2003).
7. Design

7.1 Requirements Engineering

In software-engineering, a requirement is a singular documented need of what a particular product or service should do. It is used in systems engineering or software engineering. It’s a statement that identifies a necessary attribute, capability, characteristic, or quality of a system in order for it to have value and utility to a user (Young, 2001).

The requirement phase may be broken down into requirements elicitation (gathering requirements), analysis (checking requirements), specification (documenting requirements) and verification (making sure the requirements are correct) (Wiegers, 2003).

According to Sommerville (2001) there is a clear separation between the different levels of requirements. **Functional requirements** are statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations. **Non-functional requirements** are constraints on the services or functions offered by the system. They include timing constraints, constraints on the development process, standards, etc. **System requirements** set out the system services and constraints in detail. **User requirements** are statements, in a natural language plus diagrams, of what services the system is expected to provide and the constraints under which it must operate.

Requirements generally change with time. During many projects, requirements have altered before the system is complete. This is partly due to the complexity of the software and the fact that users don’t know what they want before they see it (Pressman, 2005).

7.1.1 Original requirements

This section contains the list of requirements gathered from the drafts with which we were provided. In chapter 7.6 we will specify the final requirements that where gathered during the project.

7.1.1.1 Functional Requirements

The bracelet should have 3 panels:

- Main menu
- Sub menu
- Content panel

When the application is run for the first time, the user has to make several choices:

- Selecting a profile
- Selecting scroll-behavior
- Selecting drag-behavior
The top of the screen always shows the ‘com’-button which contains several communication options:

- Phonebook
- SMS
- MMS
- Email

The user should be able to scroll smoothly through the menu’s by dragging his finger either up or down. There are always 3 buttons visible on the screen.

The menu’s contain buttons which highlight when selected and dim when unselected. Only one item can be selected at a time. When the menu-item is selected the com, next, previous, scroll-up, scroll-down buttons highlight depending on the possibilities available.

When the user wants to open a menu-item, he should select a button and then drag it horizontally to the side (either left or right depending on the choice made by the user) of the panel. If not released on a valid place, the button will return to its original position.

When the user wants to go back to a previous screen, he should drag the button horizontally to the other side of the panel (either left or right depending on the choice made by the user).

When the user selects an item on the Content-panel he should be able to drag it to the com-button as well. When released over the com-button, the user has to make a choice to whom he wants to send the content.

The bracelet gives the user feedback concerning WIFI, Bluetooth, battery and Mobile Phone – signal status.

7.1.1.2 System Requirements

The resolution of the screen is: 240 pixels by 320 pixels.

The bracelet should contain a WIFI - / Bluetooth - / Mobile Phone – connection.

7.1.1.3 Data Requirements
/

7.1.1.4 Non-functional Requirements

The menu should be scalable, if there are more than 6 buttons, you should be able to scroll through the menu as fluently as you would with 6 items.

The application should run in a way that it would respond fast to the inputs of the user.
7.1.1.5 User Requirements

The user should be able to use the application without problems after +/- 1 minute.

The application should run smooth as well as the dragging and scrolling.

There should only be a fail-rate in the using of the application of about 10%.

The time to find specific information on the device should be as low as possible.
7.2 User interface design

The interface that the user sees during the use of the device is the single paramount aspect of the system (Bell, 2005). The user doesn't know and probably doesn't care how the system works. If the interface is easy to use, the user will be encouraged to use the device frequently, if not, the device will be neglected and the producer of the device will lose its value on the market.

Therefore it's important that we design an interface that is easy to learn, that has a variety of interaction styles and that gives feedback to its user.

In the bracelet application, we try to keep the user interface as simple as possible, although we will provide it with a futuristic touch- and feel-element. Because it's important that the user understands the true meaning of every element in the menu, every button is provided with a large icon that explains a lot of the button’s functionality. For example: the main menu contains among some other items, a button with a screwdriver-icon on it, referring to the settings of the device.

Next to the easy learnability we tried to implement a lot of flexibility in the application. On the systems first run, it will ask the user for some preferences. The user can select if the ‘next’ menu opens up by dragging to the left or to the right side of the screen. He also can adjust the scroll-settings. Once these settings are stored, the user still has the possibility to adjust them in the options menu later. Besides that, the user can choose a profile which allows him to change the background into his favorite picture, change the buttons into another color, etc…. Because we only made prototypes for three personas, this functionality isn’t implemented in our application.

A final important design principle is feedback. Every action the user performs or every interaction that is made with another device should provide the user with clear, obvious feedback. When the user taps a menu-item with his finger, various elements will lit up on the side of the screen, showing the different options available. When the user has lost his WiFi, Bluetooth or Mobile phone connection, the background color of the application will slightly turn red, signaling there are some issues. If the user receives a message, the communication icon will lit up slightly.

There are many ways users can interact with their device and ask for feedback, but unfortunately we can’t implement all of them. The user receives feedback for every sense in his body if everything still works fine. We designed this application for different persona’s, assuming that they aren’t deaf, colorblind and still have most of their fingers to interact with the device. Also, feedback needs to remain limited. People won’t be pleased with a strange artificial smell informing them about the latest news.
7.3 Prototyping

“The complexity (and the growing customer expectations) of modern GUIs makes prototyping an indispensible element in software development. The feasibility and usefulness of the system can be estimated through prototypes well before real implementation is undertaken” (Maciaszek, 2005).

Prototyping is a process model which gives the end-user, in an early state of the project, a preview of what he might be expecting in the end. A prototype can be a quick rough sketch, provided with some comments about the basic functionality or looks. Although, during the evolution of the project, the prototype sketches evolve and will become more detailed. In the project we use two sorts of prototypes: the low-fidelity and the high-fidelity prototype.

7.3.1 Low-fidelity prototype

A low-fidelity prototype is a quickly created sketch, that gives an overall view of what the application is consisted of. The sketches are called low-fi, because their functionality and design is far apart from the final product. The low-fi sketch you see in appendix 1 is the drawing we received from our initiators. It’s one of the most repeating sketches we used during the whole project. Later on, when we started implementing the design, we changed the default sketch to what you see in appendix 2.

The sketch in appendix 2 represents the first menu, containing three buttons who have the possibility to be scrolled either up or down, who are draggable to the next menu and who lit up when tapped. Of course, we could split up this low-fi sketch into three other sketches according to their different functionality, but that would lead us too far and wouldn’t be an asset to this paper. The sketch in appendix 2 is the most overall low-fi sketch we used during our project.
7.4 Platforms

During the evolution of the project we experienced many problems when choosing a suitable platform to run the application on. The following enumeration will explain the (dis-) advantages of the several platforms.

7.4.1 PDA / Pocket PC

At the start of the project, the main purpose was to make the application in Java ME (Java Mobile Edition, previously called J2ME). This language is particularly used to create menus in older mobile phones. Because we have to create an modern, good-looking interface, we think that Java ME is too limited in its possibilities. If we had to make the program in this language, we would have had to add functionalities of our own and this would have asked too much memory of the mobile phone.

After we decided not to use Java ME, we did some research to check if it is possible to run standard Java applications on a PDA/Pocket PC. We managed to make a simple test application where the user can drag images on the screen to a certain button. It didn’t take long for us to see that this was hopeless as well. We made only one of the three menu’s and the phone ran out of memory.

JavaScript

We chose JavaScript because it can be implemented on every device that has a browser. Although, Pocket Internet Explorer didn’t support our code, we managed to run our first JS application with Opera Mobile. It didn’t take long for us to notice that this code would also demand too much memory of the device. We checked, and double-checked our code and concluded that we needed a new, better machine.

7.4.2 Desktop / Laptop

It would be very easy to make an application that simply runs on a desktop or laptop. The latest models have plenty of RAM-memory and are really good looking. But, when we use one of these possibilities we would ignore the aspect of being ‘mobile’ and ‘portable’. Presenting a bracelet on an 15.4 inch screen isn’t exactly small. Besides that most of these machinery don’t support touch-screen as well.

7.4.3 iPhone / iPod Touch

Modifying the application so it would run on one of Apple’s devices would have been terrific. However, after some research we found several problems. First of all, the iPhone and the iPod Touch don’t support a drag function. This means that the user can select menu items and drag them to the next level. Instead he would have to tap the next and previous buttons. This option would change the whole concept on the bracelet and that’s not what we, neither our constituents want.
The second problem was that the apple SDK (Software Development Kit) was delayed several times and eventually released too late. Our project started four weeks earlier, and it would be useless waiting on that. However, if we had started using the SDK, we still would have had the scroll problem.

7.4.4 Tablet PC

Eventually we chose to make our final program fittable on a tablet pc. The presence of the touch screen and the portability of the device are the biggest advantages of the tablet pc. We are well aware that this device isn’t made to wear on your wrist, but at this moment it suits most of the requirements. Besides that, our tablet pc (a Samsung Q1) is at least five times as powerful as the PDA we used before.
7.5 Programming Tools

As of the beginning of this project, PDA’s and smartphones were pointed out as the devices that were going to be used for testing the application (see chapter 7.4). It was the best choice since the hardware could simulate the real size, mobility and usability of the futuristic bracelet. However, due to the lack of power of these small devices we decided to use the tablet PC as the new main goal for testing purposes. Because of its powerful hardware, its mobility and its touch screen, the tablet PC was an obvious choice. However, the real size of the bracelet wasn’t to be changed, so we simulated the bracelet on the tablet PC using its original dimensions.

Because a wide audience had to be reached, a web application was chosen. As a choice of programming language, we decided to develop a solution using HTML, CSS and JavaScript.

HTML (Hypertext Markup Language) is the predominant markup language for web pages. It provides a means to describe the structure of text-based information in a document and to supplement that text with interactive forms, embedded images, and other objects. HTML is written in the form of tags, surrounded by angle brackets. HTML can also describe, to some degree, the appearance and semantics of a document, and can include embedded scripting language code (such as JavaScript) which can affect the behavior of web browsers and other HTML processors.

CSS (Cascading Style Sheets) is a stylesheet language used to describe the presentation of a document written in a markup language. Its most common application is to style web pages written in HTML and XHTML. CSS is used to help readers of web pages to define colors, fonts, layout, and other aspects of document presentation. It is designed primarily to enable the separation of document content (HTML) from document presentation (CSS).

JavaScript is a scripting language most often used for client-side web development. It is a dynamic, weakly typed, prototype-based language with first-class functions. JavaScript was influenced by many languages and was designed to look like Java, but be easier for non-programmers to work with. The language is best known for its use in websites (client-side JavaScript).

Mootools (My Object Oriented Tools) is an open source, compact, and modular object-oriented programming JavaScript framework. It makes the process of writing extensible and cross-browser compatible code more efficient and concise.

- Mootools allows the developer to hand-pick the preferred components to use.
- It follows object-oriented practices which makes the framework rich, powerful and efficient.
- Mootools contains advanced Effects components with optimized Transitions used by many Flash developers.
- It is a framework that is mainly developed for developers

Combined with CSS and HTML, Mootools was the perfect candidate to produce the various animations of the bracelet.
7.6 Final requirements

During our project, several changes were made to the requirements. Due to insufficient communication, not every aspect was clearly pointed out at the beginning of this project. This resulted in vague information concerning the requirements of the application. However, while the project was making progress, we were able to get all the information needed to continue the development of the prototype. Because of the constant feedback of the customer we were able to obtain the final requirements (also see chapter 7.8, ‘User evaluation’).

Below is a list of requirements that were either modified or added during the project.

7.6.1 Functional Requirements

The top of the screen always shows the ‘com’- button which contains a phonebook divided into categories. Each of these categories has a list of people which the user can call.

When the application is run for the first time, the user has to select a profile.

The user can change scroll and slide behavior in the settings menu.

When the content panel does not contain any draggable content, the user should be able to return to a previous menu by dragging his finger horizontally to the side.

7.6.2 System Requirements

The resolution of the screen is: 180 pixels by 320 pixels.
7.7 High-fidelity prototype

High fidelity prototyping uses materials that you would like to expect to be in the final product and produces a prototype that looks much more like the final thing. (Preece, Rogers & Sharp, 2002)

7.7.1 Persona’s

In total, three persona’s where developed by Media-IT and the design group. The main purpose of having several persona’s is to show that the concept of the bracelet is made for a wide target group.

Albin is a 19 year old student who likes football. Besides cheering for his favorite soccer team Hammerby, he likes to go out and have a drink. His bracelet has a simple futuristic design in grayscale, decorated with a Jimmy Hendrix picture.

Diana is a 33 year old divorced women who has an adorable baby of three. She’s mainly interested in healthy cooking, fitness and spending time with her daughter. The interface of her bracelet is developed with soft relaxing colors.

The last persona is Maria, a 51 year old art director who works at an advertising agency. She likes cooking, painting and enjoys art. Her bracelet was designed with an artistic touch and uses soft pastel colors.

7.7.2 Demonstration

The high-fidelity prototype we use as an example was the one built for Albin. The interfaces of the other persona’s work similar with the example given. These can be viewed in appendix 3 and 4. The application consists of three menus. The image below shows you the main menu. Like in every other menu, the communication button is placed at the top of the screen and is always visible.

7.7.2.1 Main menu

Next to the communication button, the main menu consists of other standard buttons. After consideration with the design group we decided to incorporate “Home and Family”, “News”, “Entertainment”, “Health and Food”, “Travels and Vacation” and “Settings”. Because the application had to be intrusive, we provided the buttons with icons instead of text. Visual icons can be remembered easy and will be experienced better. If the user wants to scroll through the menu, he simply needs to touch the screen and move his finger slightly down. The menu items will start moving slowly. When the user taps a button from the menu, the button will lit up and show the different possibilities of the button (moving left, right).
7.7.2.2 Submenu

When the user drags a button from the first menu to the left (this might as well be right, depending on the users’ settings, but in the whole example we will assume the user prefers the default settings), he will go to one of the submenus. Every item from the main menu contains at least three submenus. Because the application is a prototype, some of them will partially work, others will be disabled. When the user wants to open an item in the submenu, he needs to drag the actual button the left. When he wants to go back to the main menu, he can choose any button in the submenu and drag it to the right.

7.7.2.3 Content menu

When the user enters the content menu, he’s able to scroll through it. If he thinks one of his contacts is interested in a certain part of the content, the user can drag this item to the communication button, select a destination and send the information. After the message is sent, the user will return to the content menu.

7.7.2.4 Communication menu

When the user wants to open the communication menu, he has to slide the communication button to the left. Once that’s done, the communication menu will appear. Here the user needs to choose who he wants to communicate with by selecting a certain category of people. After he picked one, the selected group will open. There he has to select a specific contact. Because our application is a prototype, we didn’t build in more functionality after this menu.

When the user wants to leave the communication menu, he needs to go back to the first level in the com-menu and drag one of the communication groups to the previous menu. Once that is completed he will go back to the screen where he opened the com-menu in the first place.
7.8 User evaluation

“Evaluation has three main goals: to assess the extend and accessibility of the system’s functionality, to assess users’ experience of the interaction, and to identify specific problems with the system” (Dix, Finlay, Abowd & Beale, 2004).

Test driven development is one of the most important aspects of Extreme Programming. By receiving constant feedback of the user, us programmers can get an excellent view of the way the user sees and ‘feels’ the application. In that way we can continuously determine what needs to be adjusted or corrected.

In appendix 5 you can find an example of our first survey. This questionnaire was given to people who didn’t know anything about concept of the bracelet. They were allowed to use the application for about seven minutes before they had to fill in the survey. Because the largest part of the design in our application was determined by the design group, we changed the main purpose of the user testing into debugging the program. Besides tracking errors, we wanted to know how users felt about the usability of the application. If the results had turned out dramatically, we could have contacted the design group about this. Luckily, the survey didn’t bring up any major problems.

Although, before we started our survey, some people from Media-IT where given the occasion to test our application. Particularly our mentor, Jesper Svensson, provided us with the necessary feedback. He is involved in the whole project and knows the requirements like no other. When the program was completely tested by him, we made the step to the public.

After analyzing the results of the first survey, we noticed that a lot of people would like to use the feedback icons on the borders as clickable scrollbuttons. Also, some users mentioned that the screen size is too large for a bracelet. After consideration with our mentor, we decided to remove the feedback icons. Making this change, more unused screen space became available. Adjusting the resolution to the new content solved the second problem.

In appendix 6 you see the results of our second questionnaire. This survey was presented to a group of students who were involved with the project and knew what the main purpose of the bracelet was. Before they could test the application, they attended a presentation about the Ubimedia project, including the development of our bracelet.

If we compare the results of the first and second survey it’s noticeable that we got back better results concerning the size of the bracelet and the feedback. This shows that we made the right choice when we changed the size of the bracelet and it’s feedback options.
8. Conclusion

As of the very beginning of this project, it was clear that this task was going something completely different from our previous work. Whereas most other projects are narrowed down, we were given the opportunity to give our personal input. Not only were we given the freedom to choose the selected programming language, we were also motivated to share our ideas and opinion.

But as in each company, the communication between the different levels can be somewhat tedious. During our stay in Halmstad, we experienced similar problems. It was sometimes difficult for us to continue the development because we needed to wait for certain feedback. Nevertheless was our work appreciated and were we involved in the process. Not only were we responsible for the creating of this prototype, we were also encouraged to make design proposals for the different personas.

Besides the programming of this application, we were asked to document our project. In this documentation we had to write down the evolution of the project. As for the selected approach (Software Development Method) we chose Extreme Programming. We would like to mention that it was a novelty for us to use this method in a real life situation. We also learned a lot of prototyping and user evaluation. We were able to put all these aspects in practice which gave us interesting information and feedback. Thanks to the users we were able to make the proper modifications to the application.

All in all, this was an unforgettable experience in which we not only learned a lot concerning programming, we also had an insight of the real business life. That our work and our documentation may be useful to others in the future.
9. References


10. Appendix
Appendix 1
Det ska gå att bygga en personlig profil i användningsutbudet. Det kommer också att finnas en del text som behöver översättas och läsbar text i olika språk.
   Till Nisse.
   Skriv.
   Sitter i parken
   och väntar.
   Kommer du
   snart?
   Åvittm.
   Sönd.

2. Nisse sitter i möte, får
   meddelande på sin dator,
   skriver ett svar.

3. Kalle får besked
   om att han har
   ett text-
   meddelande.

4. Läs.
   Mötet drar
   ut på tiden.
   Kommer om
   en kvart,
   hoppas jag.

Scrollad skärm:
(personligt inställd)

- Videosamtal
- Text/mess
- Handla
- Hälsovård

- Musik
- Familjekoll
- Nyhetskoll
- Nöjekoll
- Resekoll
- Nya Lumariform 7.0 etc

Scroll i sidled

- Startar på dagens schema
- Scroll vänster: värder
- Scroll höger: András schema, med
  klickbara detaljer
  Längst t h: Andra familjekolljänster

Inställningar - förbättring
Just nu: Söver hemma
Senare: Fika med Bellan

Just nu: Söver hemma
Senare: Fika med Bellan
Appendix 4
Appendix 5

Bracelet project
Survey March 2008

This is a survey of our project. Please select the answer that most closely reflects your views. With your help we can improve the quality of our program.

Rate the following statements.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Really bad</th>
<th>Bad</th>
<th>No opinion</th>
<th>Good</th>
<th>Very good</th>
</tr>
</thead>
<tbody>
<tr>
<td>The main concept</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The positioning of the buttons</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The usability of the scroll</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The usability of the drag and drop</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The smoothness between the different panels</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The size of the buttons</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The feedback given by the arrows</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The size of the bracelet (resolution)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The look of the application (color, shapes, ...)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Would you recommend the system to a friend?

Yes ☐ No ☐

Why (not)?

What can we improve?

The following questions will help us to understand how opinions differ between different groups. Your answers will not be combined in such a way as to risk indentifying individual respondents.

About you

Are you?

Male ☐ Female ☐

How old are you?

Under 20 ☐ 20-24 ☐ 25-29 ☐ 30-44 ☐ 45-54 ☐ 55 or over ☐

Thank you for taking part in this survey!
Appendix 6

The main concept

The positioning of the buttons
The usability of the scroll

The usability of the drag and drop
The smoothness between the different panels

The size of the buttons
The feedback given by the arrows

The size of the bracelet (resolution)
The look of the application (color, shapes,...)

- No opinion
- Good
- Very good