Consumer-based Application Customization for Mobile Phones

Bachelor thesis, 15 credits, INFK01, in Informatics.

Presented: June, 2009

Authors: Dersén, Tobias
Mårtensson, Karl

Supervisor: Svensson, Anders
Examiners: Persson, Claus
Wallin, Erik

Abstract

Title	Consumer-based Application Customization for Mobile Phones
Authors	Dersén, Tobias Mårtensson, Karl
Publisher	Department of Informatics
Supervisor	Svensson, Anders
Examiners	Persson, Claus Wallin, Erik
Year of publishing	2009
Essay level	Bachelor
Language	English
Keywords	interaction design, software development, components, IS, customization, mobile, application, EUD

Abstract

There is an interest among users for customization of mobile phones, as is indicated by several companies and websites which offer themes, ringtones, backgrounds and games. However, unless the users possess programming experience, this customization rarely goes beyond simple look-and-feel and into the actual functionality. In the other end of this spectrum, lies End User Development (EUD), where end users with programming expertise make modifications to programs themselves. In their work of designing a system for Sony Ericsson – meant to enable end users without programming knowledge to customize applications and functionality for mobile phones – the authors identified this gap between users and developers – the EUD gap. The authors believed this was the cause of frustration among some users, who perhaps did not have the motivation to learn how to program, but nonetheless wished for more control of applications and functionality in their cell phones. With the main purpose of designing a system meant to support users in such a task, this thesis has also looked at how such a solution could function in a larger, EUD-related perspective. Apart from presenting a final design proposition, the authors find indications for that there is, indeed, an interest among users to customize the functionality of mobile phones. Findings, possible implications, and recommendations based on the authors' own experience of developing the system, are finally presented, both out of an academic and a commercial perspective.

Table of Contents

Figures and Tables

Important terms

Android	An open operating system for cell phones.
Back-end	The system as it works “behind the scenes”.
Cloud navigation	Often called a tag cloud. It is generally a list of tags or search words used on a site. The tags' or links' font size or color generally indicate how popular a tag or link is.
CSS	Cascading Style Sheets, a language for defining web content layout.
Customize	This term is used in the meaning of the act of a user who modifies something to fit his or her needs.
EUD	End User Development, the act of letting end users develop, or edit, their own applications.
Flickr	Flickr is a hosting community for photos and videos.
Front-end	The part of the system that the users experience. Includes the GUI.
GUI	Graphical User Interface, the visual shell of an application which the users interact with.
HTML	HyperText Markup Language, a language used to define content for websites.

IS	Information System, commonly refers to software used to process information but can also refer to the broader sense of how information travels within an organization between people, data storage and processes.
Java	Java is an object-oriented programming language developed by Sun.
JavaScript	JavaScript is a scripting language used when creating web content. It allows a website to run code on the users' computers. Not to be confused with Java.
Mockup	A mockup is a graphical model of a system. It is similar to a prototype, but a prototype must offer a certain amount of interactivity or functionality.
PlayNow Arena	PlayNow Arena is Sony Ericsson's solution to selling music, games, themes and backgrounds to users of their mobiles.
RSS	RSS, which is short for Really Simple Syndication, is a system for publishing updates of web content, such as blogs or news. Using RSS reader software, users can automatically be notified when a web resource is updated.
UI	User Interface, the shell of an application which the users interact with. Same as GUI, except that it does not necessarily have to be graphical.

Please note that in this article, mobile phone, mobile devices, mobile, cell phone, and phone, are used interchangeably and bear the same meaning of cellular phone.

1 Introduction

The origin of this thesis has been to design a system for Sony Ericsson. Based on the popularity of of their tool Themes Creator, the company wanted a system that would allow end users of mobile phones to customize functionality and applications to a larger degree than what was possible – without the need for programming. The main focus was on how the flow and interaction between end user and system should be designed, in order to make the process of customization as effortless as possible. The authors were virtually given free hands, and were allowed to choose for themselves how to work, and what to make of the project. The progress of the project was regularly reported to Sony Ericsson, but overall, the company had a very passive approach – only sharing their opinions when specifically asked by the authors, and also leaving all decisions to be made by the authors. It was therefore decided very early on to investigate how such a project could contribute with something more than just a system design. That is, how it could create something of interest for other parties than Sony Ericsson.

During the initial investigation, it was discovered that the project was related to the research area of End User Development (EUD). The authors also discovered what they considered being a gap between programming end users, and end users without programming knowledge, a gap hereon refered to as the EUD gap. This EUD gap could be considered hindering for the programming-inexperienced end users, who perhaps wished for more control of functionality and applications, but who did not have the motivation to learn how to program. It was deemed interesting to study how the design solution for Sony Ericsson might be of aid in such a situation. Related to this, was also the question of whether or not this kind of user would at all be interested in the concept of customizing applications for mobile phones; a matter which was polled in user tests.

1.1 Purpose

The have been two purposes to this thesis. The main one, has been to design a system for Sony Ericsson that allows end users without programming knowledge to customize applications and functionality for mobile phones. The authors have looked at how such a system can be created to give maximum flexibility to users, while keeping it simple enough for anyone interested. The second purpose, has been to study how this design could aid in bridging the EUD gap. A subgoal to this has also been to find indications of whether or not the concept of customizing applications for mobile phones is something desirable by users.

While the solution is aimed at mobile phones, the authors believe this research could be applied to other areas as well, such as tools supplied with operating systems, and to other situations where a developer of a platform would want to supply customizable software for the users of said platform.

1.2 Research questions

The main research question of this thesis has been: How should a system be designed in order to best support non-programming users in customizing applications for mobile phones? This question relates not only to the end product, but also to the process of creating it. Related to this, is a subquestion, namely: Is there a demand for a service such as this?

Lastly, the thesis has also looked at how the authors' designed solutions could aid in bridging the EUD gap.

1.3 Delimitations

The authors have designed a system for Sony Ericsson that allows end users without programming experience to customize functionality for mobile phones. The system was, however, not implemented by the authors themselves, as this was beyond the scope of the project. Instead, a mockup prototype of the system was created and tested upon users, in order to study their impressions of the concept of user customization of applications for mobile phones.

Focus has been on the concept of the system and the user interaction. Emphasis has therefore not been on technical issues. Aspects such as mobile platforms and back-end architecture have, however, been dealt with, but kept on a non-technical level to as large of a degree as possible.

With regard to the subquestion – that is, if there is a demand for such a service – the authors are aware of the fact that a demographic market survey would have been the most appropriate in order to achieve a reliable answer to this question. That, however, was not possible with the time and resources available, and in order to not lose this question entirely, it was surveyed with the help of the user tests. While not entirely reliable, the answers did provide an indication.

1.4 Disposition

The authors believe there are two groups that might be interested in reading this report. The first one is naturally Sony Ericsson – and other businesses interested in offering customers increased customization possibilities – and the second one is researchers active in the End User Development area. For the sake of clarity, the disposition of the report is briefly explained below.

In short, this report consists of two parts. The main one is concerned with describing the process of designing a system for Sony Ericsson, and the result of this process. The second part treats the concept of End User Development, and how the system could aid in allowing users to bridge the EUD gap.

The overall disposition of the thesis can be seen in figure 1.1. It begins with this introductory chapter, followed by a theory chapter. The theory chapter presents a theoretical background for the system area, a part on interaction design, a part on modularization and components, and ends with theory on End User Development.

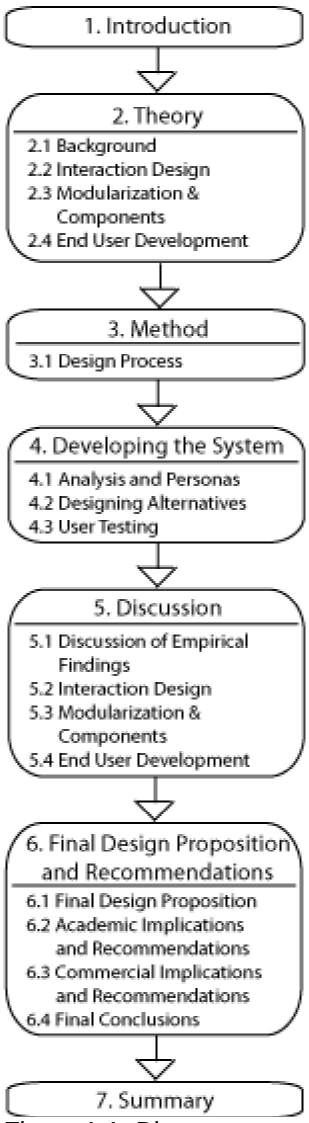
Chapter 3 describes the methodological approach and design process of the report, with motivations for why certain approaches were chosen.

Chapter 4 begins by presenting an in-depth description and motivation of the personas, followed by an equivalent presentation of the initial design alternatives, and the interactive prototype used in the user test. The chapter ends with the results of this user test.

Chapter 5 first discusses the implications of the results of the user test, followed by discussions on the use of interaction design, modularization & components, and End User Development in the project, respectively.

Chapter 6 presents a final design proposition for the system, and also discusses academic and commercial implications and recommendations, based on the authors' experiences in the project. It ends with a part on the final conclusions the authors have reached.

Finally, chapter 7 ends the report with a summary.


1. Introduction
2. Theory
2.1 Background
2.2 Interaction Design
2.3 Modularization & Components
2.4 End User Development
3. Method
3.1 Design Process
4. Developing the System
4.1 Analysis and Personas
4.2 Designing Alternatives
4.3 User Testing
5. Discussion
5.1 Discussion of Empirical Findings
5.2 Interaction Design
5.3 Modularization & Components
5.4 End User Development
6. Final Design Proposition and Recommendations
6.1 Final Design Proposition
6.2 Academic Implications and Recommendations
6.3 Commercial Implications and Recommendations
6.4 Final Conclusions
7. Summary
Figure 1.1: Diagram showing a brief overview of the report disposition

1.5 Disclosure Statement

This work has been made for Sony Ericsson, and thus the final design proposition may be biased towards this company; however, the authors have tried to the largest extent possible to stay unbiased, professional and academic throughout the study. This possible bias should also not affect the study of the customization concept in itself in this type of project, nor the result of it. The design propositions might, however, be influenced by the connection with the company for sake of practicality; working with an existing platform in mind is easier than working with an imaginary one. The platform itself is however of subordinate value, as long as it functions well. The authors considered the opportunity to work on a project such as this to outweigh the possible drawbacks of the private interests of Sony Ericsson.

As for the choice of working with specifically Sony Ericsson, the company was chosen for several reasons. Not only is it a big company close to the university campus, but it also has a lot of clients and a big market share. This, the authors hoped, would give access to a broad user base that would hopefully make the research more relevant (Sony Ericsson, 2008). The authors also believed Sony Ericsson's experience in the field could help attain relevant results by directing the authors' efforts towards issues Sony Ericsson had found important. Seeing as both authors had contacts at Sony Ericsson, it was relatively easy to get research access. The project and purpose of the thesis was decided and planned in cooperation with these contacts.

This close cooperation with Sony Ericsson did, however, mean there were special restrictions on the authors as researchers, and ethics had to be considered. The interviews with employees at Sony Ericsson had to be carefully conducted, translated and edited to exclude company secrets and strategy. The authors did their utmost to satisfy Sony Ericsson's restrictions while still keeping the research objective and academic.

2 Theory

When an investigation in search of a theoretical basis upon which to build the system had been conducted, three areas were chosen for this purpose, namely: Interaction design for the process of designing a user friendly system, modularization and components in order to ensure a solid, architectural foundation for the system, and End User Development in order to address the EUD gap. These parts are, after a presentation on the theoretical background for the system area, each presented in the following chapter.

2.1 Area Background

Before commencing the design of the system, the authors decided to look into the background of the area, which includes ideas on mass customization and End User Development. The following part is a result of that investigation.

Mass production is outdated and has been for some time. Already in 1993, Pine II wrote that mass customization is the new way to do business; that while mass-produced and custom-made used to be two opposites which could not be combined, this had started to change more and more. This was also true for IT, where the era of mass production had been transcended, and instead computers could be highly customized to fit the users' needs. Even the software inside the computer had evolved from being created for one customer to be created for all customers, being shipped with a multitude of user customizable features. The concept of mass customization has only grown in importance since then, especially in the manufacturing industry, much thanks to the opportunities IT, internet and e-commerce provide (Chang & Chen, 2009; Tseng & Jiao, 1998). The reason for this is that "customization can lead to a higher level of customer satisfaction" (Chang & Chen, 2009, p. 7), which in turn makes "customers [...] willing to pay premium price for their unique requirements being satisfied, thus giving companies bonus profits" (Roberts & Meyers, 1991 according to Tseng & Jiao, 2001, p. 687).

According to Piller (2005), mass customization can be divided into three levels: style, fit, and functionality. Style is concerned with customizing the looks and visual aspects of the product, fit relates to measurement tailoring in accordance with for example body size, and functionality deals with customization of such things as the cushioning of a shoe, power, output or upgradeability. This last aspect of customization is something that Piller (2005) means is often overlooked, and he considers it to be the least utilized of the three dimensions of customization. He continues to write that "Embedded [software] configurators could become a very promising new technology that would allow customers to continuously re-configure a product" (p. 322).

While Piller is mainly concerned with mass customization in regards to the manufacturing industry, the same concept he writes of could be considered applicable in the area of IT. Here, user customization of software has existed for more than a decade; ranging from the most common activity of changing given settings and simple reorientation of buttons, to the rather rare activity of writing complex macros (Oppermann & Simm, 1994). A study by Page et al. (1996) showed that 96% of users customized their word processor to some extent, and a later study by Kahler (2001) further supported that users customize their word processor. The goal of customization has mainly been to either make the program match the users' use patterns, to amend annoying or slow functions, or to alter a newer version of a program to look like an older version due to old habits (Mackay, 1991). In short, it is made to increase productivity and make it easier and smoother to use the program.[1] This user customization could be considered a simple form of End User Development (Fischer et al., 2004), where McGill (2004) has showed that EUD not only makes end user developers perform better with the product they have made, or participated in creating, but also makes them feel more satisfied with it. Rivera (2005) showed that this also holds true for UI content customization.

However, this kind of customization is, in accordance with Piller's (2005) thoughts on mass customization within manufacturing, overall on a very superficial level. It is mainly concerned with the looks and design of the products, and is at the most a matter of limited, internal attribute customization – so called parameterization (Lieberman et al., 2006) – where functionality and composition are static (Chang & Chen, 2009). This holds true for both customization of software (Page et al., 1996) and manufactured products (Chang & Chen, 2009).

In IS/IT, the opposite of this rather superficial customization would be users who themselves write programs or modifications to existing programs, an area which is the main concern of EUD research (Fischer et al., 2004). Most of the time it is simply a matter of a person with a regular job, who happens to be interested in, or know some, programming, and decides to write a program or program modification to make it easier for himself/herself to work (Fischer et al., 2004). Companies can, however, support this by acquiring systems specifically built to cater such needs (Fischer et al., 2004), and EUD research is continuously looking for new ways to lessen the degree of programming expertise required by users to make these programs or modifications. Lieberman et al. (2006) write:

> By now, most people have become familiar with the basic functionality and interfaces of computers. However, developing new or modified applications that effectively support users' goals still requires considerable expertise in programming that cannot be expected from most people. Thus, one fundamental challenge for the coming years is to develop environments that allow users who do not have background in programming to develop or modify their own applications, with the ultimate aim of empowering people to flexibly employ advanced information and communication technologies. (p. 1)

However, most of the time, the systems still require actual programming in one way or another, and in spite of many attempts to make it easier for non-programmers to make modifications or their own programs, such as visual programming (Kindborg & McGee, 2007) or natural programming languages (Pane & Myers, 2006), there is still a gap between these two extremes of superficial customization – or at the best limited parameter setting – and actual programming done by the users. A gap where users without programming knowledge might want to customize functionality, or even make their own simple applications, but do not have the time, motivation or resources to

1. While the authors are aware that these studies of customization may be somewhat dated, they have not been able to find any more recent studies. However, there are virtually no signs that the situation of customization would have changed to any greater degree since.

learn how to program. In other words, the EUD gap. This holds especially true for mobile devices, where customization is immensely popular – as indicated by the existence of countless customization services – but mainly limited to superficial content like ringtones or themes. However, users have a large need for more services and content in their mobiles than just ringtones and the like, as is shown by Mahatanankoon et al. (2004). While third party applications are widely available, these are often impersonal and not customizable, leaving users with close to no possibility of making or customizing their own applications if they do not have any programming knowledge; something which both Korpipää et al. (2006) and Subramanya and Yi (2005) have seen as well. It can thus be assumed that mobile users have a need, or at the very least an interest, for creating their own applications and customizing the functionality of their phones.

2.2 Interaction Design

> A number of terms have been used to emphasize different aspects of what is being designed, including user interface design, software design, user-centered design, product design, web design, experience design, and interactive system design. Interaction design is increasingly being accepted as the umbrella term, covering all of these aspects. (Preece et al., 2007, p. 9)

Interaction design is, according to Preece et al. (2007, p. 8), about "designing interactive products to support the way people communicate and interact in their everyday and working lives". In other words, it is about designing a system that supports the users in their tasks, and that the users find enjoyable, pleasant and productive. In the following parts, the interaction design process by Preece et al. (2007) is presented together with techniques from Cooper (2004), as well as user stories by Cohn (2004), followed by usability goals according to Preece et al.

The usability engineering life cycle (Nielsen, 1992) is a renowned article with 5543 citations (Google Scholar, 2009) about software usability engineering. In the article, the interaction design process is split into 10 identifiable elements, giving a slightly more detailed and complex model over the process than the one presented by Preece et al. (2007). However, the decision was made to use the model by Preece et al. (2007), since it is considerably newer and easier to work with, and thus more fitting for a project of this size. Seeing as Preece et al. refer to Nielsen's work, the authors believe this gives Preece et al. a certain academic validity, apart from the researchers' own academic prestige. While the models are different, they nonetheless share the philosophy and intent of the process.

2.2.1 Interaction Design Process

Preece et al. (2007) present in their book a simple interaction design life cycle model which they have derived from life cycle models, stemming from related research areas such as software engineering and Human-Computer Interaction, in addition to personal experiences. This iterative model consists of four steps, namely: Identify needs/establish requirements, (Re)Design, Build an interactive version, and Evaluate. These are presented in further detail below.

Identifying Needs and Establishing Requirements

The key to solving the problem is interaction design before programming. (Cooper, 2004, p. 16)

Before setting about gathering data and performing interviews, Preece et al. (2007) mean that one needs to focus on four key issues: to set the goals for the data gathering (i.e. for what purpose the data is being gathered), to consider the relationship between the data gatherers and data providers (in other words, maintaining professionalism), to triangulate (that is, using several different methods for gathering data), and make a pilot study (a small trial run of the entire survey in order to test its viability).

Once these are set, the data can start to be gathered in order to establish needs and requirements. This data can be gathered in a multitude of ways, mainly through different kinds and techniques of standard surveying methods such as interviewing, questionnaires, and observation. The data gathered should then be analyzed and structured in order to assess the needs and requirements that one must possess good knowledge of before starting the development process. Naturally, it is not expected that all needs and requirements will be discovered the first time this step is iterated; more will certainly be uncovered throughout the course of the development process. (Preece et al., 2007)

Related to this stage are the techniques Cooper (2004) suggests in order to help making requirements less abstract. He calls these personas, goals, and scenarios.

Personas are "*hypothetical archetypes* of actual users" (Cooper, 2004, p. 124), meant to exemplify a future user of the system being designed. Cooper argues that instead of speaking of "the users" in a general term that changes its meaning depending on the situation – for example the users being fully capable of understanding a semi-advanced task at one point, but degraded to being IT-incompetent at another – it is much more beneficial to the design process if one creates one or a few personas. The most important aspects about these are that they are precise and representative (Cooper, 2004). Personas, to the developers, must become real people, and thus require names, jobs, looks, skills, a personal life, and more. The personas must also represent a group of actual potential users, which is why they are made according to stereotypes.

Cooper means that by using personas instead of asking real users, the problem of the real user deviating from the norm can be avoided, apart from the fact that a persona is much easier to handle. Grudin and Pruitt (2002) disagree with this and mean that personas should be based on, and created with the help of, as much quantitative and qualitative information about, and from, the users as is available. Seeing as such an extensive data gathering was beyond the scope of the report, the authors felt that Cooper's approach was more appropriate. The purpose is nonetheless – in short – to design for one person, and not a generic mass of people, thus gaining a much clearer view of "what the product must do–and can get away with *not* doing" (Cooper, 2004, p. 131).

"The essence of good interaction design is to devise interactions that let users achieve their practical goals without violating their personal goals" (Cooper, 2004, p. 150). A goal is the purpose for which a persona engages in a task. For example, if a persona wishes to write a novel, then the finished book is the goal, while the hours spent in front of a word processor writing, is the task. While tasks change along with technology, goals do not. By focusing on goals instead of tasks, Cooper means that better and more appropriate designs will be created, because to the persona, different tasks are merely a way of reaching her goal, and she does not necessarily make a distinction between these.

Goals are divided into personal goals, corporate goals, practical goals, and false goals. Personal goals are universal and general, and mainly circulate around such things as to not feel stupid, not to make mistakes, or to have fun. Corporate goals are the equivalent of personal goals to the corporation developing a product. These include, for example, to increase the profit, increase the market share, or to provide better services than the competition. Practical goals "bridge the gap between the objectives of the company and the objectives of the individual user" (Cooper, 2004, p. 157). Cooper exemplifies these as avoiding meetings, dealing with the clients' demands, or handling the clients' orders. Lastly, false goals are goals that are not actually real goals, for example for a program to be memory efficient, to run on a certain platform, or to be easy to learn (Cooper, 2004).

While Cooper (2004) does provide fully sufficient coverage of personas and goals, the extent to which he explains the practical use of scenarios is lacking. While this could be complemented with an external source such as for example Carroll (2000), Carroll's scenarios are not as synoptic and easily used as Cooper seems to suggest they should be. Furthermore, the nature of the task also makes it very difficult to implement scenarios in the way Cooper (2004) intends – with daily-use, necessary-use and edge-case scenarios – which is why the authors have chosen to replace scenarios with an adaptation of user stories. User stories are mainly used in Agile Software Development such as eXtreme Programming, and consist of one or two sentences representing a system requirement written on a paper card by a future user of the system (Cohn, 2004). Compared to formal requirement documents, they provide an easy way of formulating user requirements, and are used throughout the whole project for planning and prioritization of system functionalities (Cohn, 2004). An example of a scenario would be: "As a heavy user of the system, I regularly need to be able to print documents quickly and easily."

Redesign

This phase is divided into two parts: the conceptual design and the physical design. In the conceptual design process one designs a conceptual model, defines what the product should be able to do, what it should look like, and how it should perform these tasks. The physical design is more concerned with detailed specifications of the final product, such as how the menu should be designed, what colors that should be used, what the icons will look like, or what sounds that should be played upon completion of certain tasks. The most important part of this phase is that one designs, or at least considers, alternative solutions to virtually everything. (Preece et al., 2007)

Building an Interactive Version

Building on the designs developed in the previous phase, this phase focuses on making interactive versions of these designs in order to let users test and evaluate them. These interactive versions do not necessarily have to be software-based, but could for example be very simple and paper-based prototypes (Preece et al., 2007). Prototyping is beneficial to the design process as it can be produced much faster and cheaper than a real and fully functioning program, thus allowing for user testing and evaluation earlier. This in turn highlights problems and design errors earlier, making these substantially cheaper to correct than they would be at a later point in time (Cooper, 2004).

Evaluate

Using the interactive prototypes that were built, this phase evaluates the experience the prototypes produce for the users. It looks at many different criteria dependent on the nature of the product being designed; for example how well it meets the requirements posed earlier, how easy it is to use and learn, how appealing it is to use and look at, and so forth. It is important to note that evaluation in a way is integrated throughout the whole interaction design process, due to the constant focus on considering alternative designs and choosing the best, as well as the iterative nature of interaction design. (Preece et al., 2007)

2.2.2 Usability Goals

Knowing for what purpose a product (here onwards referred to as system) is being developed, is part of the interaction design process. Preece et al. (2007) consider this to be best dealt with using so called usability goals, which they break down into the following six subgoals: effectiveness, efficiency, safety, utility, learnability, and memorability. An imaginary word processor will be used to exemplify the goals.

Effectiveness is, quite simply, how effective a system is; i.e. how well it performs the task it is meant to perform (Preece et al., 2007). It is a very general goal, but in the case of a word processor, it could be exemplified as the following: In order to be effective, a word processor should at the least be able to handle large amounts of text, and preferably support basic text editing functions. If the program freezes when a document reaches 5000 words, then it is quite clearly a rather poor word processor, and far less effective than one able to handle an infinite amount of words, since one will not be required to make a new document every time the word count reaches 4999.

Efficiency deals with the product while in use (Preece et al., 2007); does it efficiently support the users in the tasks that the users want to perform? A word processor that enables the users to write more quickly with for example the help of keyboard shortcuts, could be considered more efficient than a slower counterpart lacking such a feature.

Safety is related to issues such as prevention against accidental deletion of files or crashing a program, as well as being able to undo things that were not meant to be done. It is also related to providing users with the possibility of exploring the system without risk of causing havoc (Preece et al., 2007). In the case of a word processor, safety is for example the system's built in function for auto-saving a document, or the undo-command.

Utility deals with the aspect of functionality (Preece et al., 2007). A word processor must naturally have basic functionalities like support for writing and editing text as well as saving documents in order to be called a word processor, but a word processor that also provides functions for text formatting, printing, and automatic spelling checks certainly offers more utility than one not providing these functions.

Learnability has to do with the learning curve of a system (Preece et al., 2007). How easy is it to learn how the system functions? How long does it take? Do the users think that it is worth the effort? A word processor in this case is an example of system in which it is very easy to learn the basic functionality. One merely opens a new document and starts typing. If one is familiar with the computer and keyboard to begin with, this is no problem. However, word processors these days often come with a multitude of functions of various complexity. A system that provides a tutorial or help document for users that wish to learn how to use this is a more learnable system than one that merely expects users to find it all out for themselves.

Memorability builds on the previous stage of learnability (Preece et al., 2007). Once a system is learned, memorability is related to how easy the use of a system is to remember. Especially if a system is used rarely, this goal is of major importance, as users should not have to relearn an entire system merely because they use it infrequently. A word processor can support this by for example having logically grouped icons that are easy to understand.

2.3 Modularization & Components

Modularization is the idea that a complex system can be made simpler by breaking it down into smaller parts – modules – which work together to make up the larger system. Baldwin and Clarke (2000) state that modules should be interdependent within themselves and independent across each other. In essence, a module should be made up of what is vital to its functionality while keeping the connections to other modules as weak as possible.

Baldwin and Clark (2000) mention three key terms when it comes to modularization: abstraction, information hiding, and interface. When a system becomes too complex, one divides it into smaller pieces; each piece being hidden behind an abstraction, which has a simple interface that can be seen as a form of access-point. This hides the actual information in the piece behind a simple front-end. The other parts of the system do not have access to the information in the piece; instead they communicate with it through the interface. This means that as long as the interface is kept the same, the information in the module can be changed without any impact on other modules.

A common way to modularize a system is to create it using components.

> A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties. (Szyperski, 2002, p. 41)

Each component represents something – an object or some form of functionality – and it should be possible to use them independently, but also connected together by their interfaces. A component is a unit of deployment, meaning that a component will always be deployed as-is. One cannot deploy just a part of a component. The knowledge of how a component is implemented should not be needed in order to use it. Components can thus be added or optimized with relative ease and without any impact on the system as a whole. (Szyperski, 2002)

Wang and Qian (2005) express 5 principles of component-oriented software engineering and programming:

Components represent decomposition and abstraction: As a technique for modularization, components build upon the idea that complex problems should be decomposed, and details of implementation should be hidden by abstraction.

Reusability should be achieved at various levels: Not only should the components themselves be reusable, but the specifications and implementations can also be reused.

Component-based software development increases the software dependability: Components will be used in various situations and optimized. As the components have been used before, one can trust the components to be secure and stable.

Component-based software development could increase the software productivity: Assembling previously made and pretested components is faster than rewriting functions for each application.

Component-based software development promotes software standardization: By standardizing interfaces, components can be used in a "plug-and-play" manner, similar to USB-connected hardware on computers.

According to Wang and Qian (2005), a component has a name and a collection of properties, methods, and events. Properties are variables concerning the component's state and attributes. Methods are the services the component offers. Events are actions the component can initiate. When connecting components, it is often the case that an event from one component calls a method in another component.

2.4 End User Development

EUD (End User Development) is defined by Lieberman et al. (2006, p. 2) as "a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify, or extend a software artifact. "

The idea is to grant power over the software to the users, in an easy and approachable way. EUD is a growing field, the number of end user developers growing constantly. It is also a wide research area, encompassing such disciplines as HCI (Human Computer Interaction) and AI (Artificial Intelligence), as well as natural and visual programming. (Lieberman et al. 2006)

Another concept in EUD is the belief that one can go from design-before-use to design-during-use. Instead of trying to design the system from the start to be perfect for the end users, the idea is to let the end users design the system to be perfect for their individual needs while using it. A precondition for this development, is that systems are built with the purpose of being flexible, as well as letting users modify them beyond simple means, such as parameterization. (Lieberman et al. 2006)

2.4.1 Approaches

When it comes to End User Development, there are several different approaches, such as component-based development, visual programming languages, simple scripting, and programming by example.

Components in End User Development are parts of a program that have made beforehand, which can be controlled by defined interfaces (Won et al. 2006). Users would thus learn how the component is used, instead of having to learn how it is programmed (Won et al. 2006). By connecting and mixing components, an application can be created (Won et al. 2006). This is the approach focused on in this thesis; though it is mainly treated in the freeform alternatives (*Min*, *Xiang* and *Wu*).

Visual programming languages are programming languages based on visual syntax instead of textual syntax. The goal of such a focus is to make the programming easier; instead of having to learn how a computer perceives a problem, the programmers explain their ideas in images. Kindborg and McGee (2007) have published an article on how visual programming based on analogical representations can make programming easier. The goal of an analogical approach is to have the source code of a program look like the run-time end result, much like how a map is a representation of the world it depicts. The article by Kindborg and McGee (2007) focuses on how the concept of comic strips can be used in visual programming, since comic strips, just like a visual programming language, use something static to describe something dynamic.

Scripting languages are similar to traditional programming languages, but have a different goal. While traditional programming is supposed to create algorithms and data structures from the simplest abilities of the computer, scripting languages connect such simple components together. Examples of scripting languages include Perl, Python and Ruby. Such languages can vary significantly in learnability and power. In general, they sacrifice speed of execution for speed of development. (Ousterhout, 1998)

Programming by example, also called programming by demonstration, is when the programmer exemplifies a rule, and the computer analyses it to automatically generate the code. For example, the programmer might drag a train object over a rail object in order to show the computer that trains should only travel across rails. The limit to how useful programming by example is, is how hard it is for the computer to generalize. The recorded steps have to be generalized by the computer in order to work in other contexts than the one exemplified by the users; this is what makes programming by example different and more powerful than macros. Getting a computer to generalize is, however, not a simple problem. If one attempts to make the generalization flexible, the risk of the computer generalizing incorrectly increases. (Lieberman, 2001)

3 Method

The research conducted in this article aims to develop an IS system that allows end users to customize applications for mobile phones. It also intends to investigate if such an IS system could bridge the EUD gap, and if there is an interest among users for such a concept. The process began by developing the system for Sony Ericsson, using the user interaction approach defined by Preece et al. (2007). The system designed was later tested on potential users in order to find information on how useful the system was, and how useful the concept of customization in mobile phones was for end-users. When the system design had been finalized, the authors analyzed and discussed the insights they had attained during the project. One can thus make the case that this research was both inductive and deductive (Jacobsen, 2002). The design of the system used a deductive method where empirical findings on interaction design and component-based approaches were used. The discussion and analysis, however, was inductive; the empirical basis for the authors' arguments being the project itself.

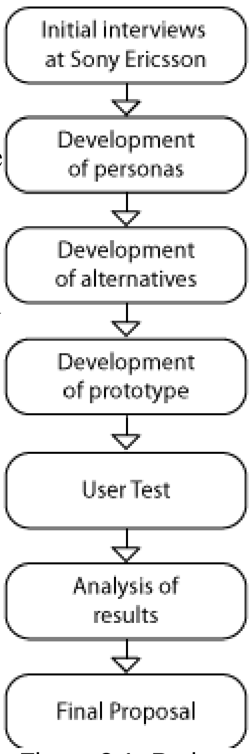


Figure 3.1: Project process overview

Figure 3.1 shows a map over the process of the entire project. It begins with the initial interviews with employees at Sony Ericsson – conducted in an effort to understand the problem area – and ends with the final proposition and its possible implications.

To develop the system, theories and previous research was needed. This was found at the library of the University of Lund using Lovisa, from the University of Lund's repository of scientific articles, ELIN, Google's scholarly search engine scholar.google.com, and Google itself. The search words used were interaction design, component, End User Development, modularization, mass customization, and mobile customization.

3.1 Methodological Approach

The research conducted can be considered both qualitative and quantitative. The initial data gathering was mostly qualitative since the authors had limited knowledge of the problem area, and the authors needed to know how the key employees at Sony Ericsson understood and interpreted this. After all, the main purpose of the project was to design something deliverable to Sony Ericsson; it stands to reason that the authors needed to learn what the company wanted and expected. Jacobsen (2002) supports the use of qualitative measures when the goal is to understand how people interpret and understand a situation. The user tests, however, were conducted in a quantitative manner since the literature recommended the use of closed-question questionnaires for

this purpose (Preece et al., 2006; Dumas & Redish, 1999).
The research assumed an individualistic approach (Jacobsen, 2002) in that it drew upon the authors' personal experiences of developing a system, and attempted to draw parallels to other areas of system development. A holistic approach (Jacobsen, 2002) would have been interesting, but the authors felt that a tangible system design was needed to get reliable results. However, the authors believe they may have opened the door for further research within other companies, in order to determine how general their conclusions are.

Seeing as the authors themselves developed the system, it was not possible to assume a distant approach to the research, as their effect on the empirical findings could not be minimized. On the contrary, their own work was as much part of the research as the evaluation of the system, and the approach in designing it. According to Jacobsen (2002), many researchers consider an intimate approach to research superior, as it is never possible to completely counteract the researchers' impact on the research conducted, thus, this is not considered to be a large issue.

3.2 Design Process

The following parts describe the entire design process of the project. It starts with initial data gathering, and moves on to establishing requirements. Afterwards, it is presented how this information was used to create simple mockups and a prototype that demonstrate alternative solutions. Lastly, it is presented how the user testing was conducted.

3.2.1 Initial Data Gathering

Building on the interaction design process proposed by Preece et al. (2007), the authors commenced by "identifying needs and establishing requirements for the user experience"; however, this stage was divided into two parts – one data gathering part, and one analysis and requirements establishing part. This due to the scope of the requirements establishing activities, which are explained in part 3.2.2. The data gathering was conducted through interviews with three key employees of Sony Ericsson; all who through their work as product owners of third party developer and/or consumer tools, possessed a good understanding of the EUD gap that this report deals with. One of these key employees wished to remain anonymous, and required that the interview was not included in the report. The name of the employee has therefore been replaced with "the third key employee" in the two places it is referred to. An alternative approach in this case might have been to conduct a market survey among potential users, to see how big the EUD gap truly was, and whether users were interested in a tool that would allow them to customize functionality in their cell phones. However, at this point in time, the concept of what was needed to fill the EUD gap, and the scope of it, was still on such a highly conceptual level that it was deemed impossible to formulate it in a way that would make potential users understand what it actually was, and how it would work. It was therefore decided that actual users should not be involved until a tangible system proposal had been designed.

A semi-structured approach was, after much consideration, chosen as the most appropriate form of data gathering for the initial interviews with the Sony Ericsson employees. This is supported by

Jacobsen (2002) who writes that open interviews are the most appropriate when the interviewees are few, and when what each individual interviewee says is important; both these aspects being highly relevant in this case. Jacobsen also indicates that open interviews may be structured, if certain aspects of the interview have to be put into focus. Seeing as specific answers were desirable concerning which theoretical basis to build on, as well as to grasp the scope of the project, this focus was needed; hence the semi-structured approach. Oates (2006) expresses that unstructured and semi-structured interviews are unfitting when the researcher wants to generalize the results. As there was no need for generalization in this case, this issue was deemed irrelevant.

These semi-structured interviews roughly followed the interview guide (appendix 1) content-wise, however, the structure of the interviews could not be kept the same due to individual deviations in the understanding of the problem area. The interviews were recorded, and the recordings were transcribed, translated, and sent to each interviewee for verification and approval. Corporate secrets have been removed from the transcriptions (appendices 2, 3 and 4).

Relating to the four key points that according to Preece et al. (2007) have to be decided before the data gathering begins (explained in 2.2.1), with regards to the goal of the data gathering, this was clear from the beginning: to establish an initial base of requirements on which to base the first, simple mockups. The relationship between the interviewees and interviewers were kept on a professional level, as between employers and consultants. However, in the case of triangulation, several different methods for gathering data were not used, due to the nature of the task and the very limited number of people that were to be interviewed at this stage. Also, a pilot interview was not conducted. It would have been fruitless since the questions were supposed to be little more than a starting point in the discussion with Sony Ericsson, regarding their expectations on the project.

The interview (appendix 1) starts by asking the interviewee about his work within the company. The purpose of the question was mainly of an introductory nature, but also gave the authors a context in which the system was to be developed. Two questions are then asked in order to uncover the hopes the interviewee has for the system to be designed. Well aware that most of these hopes could not be realized by this project, the authors felt that the answers could give a direction as to the actual goal of the system to be developed. These questions were followed by specific and more concrete questions, aimed at illuminating issues that had been identified during the initial discussions with Sony Ericsson, such as the issue of salability and the issue of multiple mobile platforms. These questions also had the purpose of providing guidance towards the design of the system. The questions were designed in order to not restrict the answers; for example, by using vocabulary such as “expect” or “perceive”. This was also meant to avoid leading questions (Seidman, 2006).

3.2.2 Analysis and Requirement Establishment

In the second stage, dealing with analysis and requirements, the techniques presented by Cooper (2004) – personas and goals, as well as user stories – were used in order to concretize the requirements and needs found through the structuring and analysis of the interviews. While not being without drawbacks or risks, much like any other techniques, the use of personas, goals and user stories have been shown to be a very effective and powerful tool (Grudin & Pruitt, 2002; Pruitt & Adlin, 2006; Cohn, 2004) in the design of interaction interfaces. The main focus was Cooper's interconnected version of personas and goals, as he offers a very concise and easily managed overview of these techniques (as opposed to Grudin and Pruitt (2002), who focus solely on personas). As mentioned in the theory chapter, Cooper's (2004) scenarios were not used. Instead, the authors were inspired by user stories from agile development, and used an adaptation of these in the work. They were found to be very simple and easy to use, thus following the general principles of Cooper while providing a solid and practical alternative to scenarios. While in agile development the user stories are written by actual future users (Cohn, 2004), these had to be written by the authors themselves. This is naturally not as ideal as user stories written by real users, but in the lack of such, it was determined better than none. Also, seeing as they were replacing scenarios, which were written by the authors themselves, this was not deemed an obstacle. The user stories were simply a way of formulating the persona requirements in an easy and effective way. In agile development, user stories are also used in a more administrative way, such as prioritizing and planning the amount of time needed for implementation of the functions (Cohn, 2004). As the system being designed would not be implemented within the scope of the project, this use of user stories was not deemed necessary.

Once the personas had been developed, they were sent to Sony Ericsson in order to find out which of these Sony Ericsson considered to be best in line with the main focus of the project, and to compare this with the authors' own opinions. At this point in time, the company also involved a fourth, newly hired, employee in the project, whose job it was to deal with precisely such issues as consumer and prosumer tools (appendix 8).

3.2.3 Initial Design Propositions

In the third stage of the design process, a few alternative, simple, low-tech mockups were created with the help of the personas; the major focus being on the conceptual design. These mockups were initially spawned on a white board, and later refined in a digital drawing program. At this point they were demonstrated to the same key employees within Sony Ericsson that were involved in the project; this was done in order to evaluate the mockups and receive feedback on the proposed solutions.

3.2.4 Interactive Prototype and User Test

The fourth stage dealt with making an interactive, digital prototype; a solution proposal that built on the feedback received in the previous stage, as well as what was deemed feasible within the limited time and resources of the project. The prototype (appendix 20) was made by further refining the mockups of one of the designs from the previous stage (*Mandarin*, appendix 11). It was turned interactive with the help of a slideshow program. In this stage, preparations were also made for the user test that would be conducted using the interactive prototype. This involved creating a user test scenario (appendix 17), as well as creating a questionnaire (appendix 19) which would be filled in by the users once they had completed the test scenario. Although the user base would be relatively small, a post-test questionnaire is recommended when it comes to user tests as it provides easily quantifiable, hard data (Dumas & Redish, 1999). The purpose was to find out how usable and user friendly the proposed system was, and to find out whether such a system would be desirable. Before each test, the users would be informed about the test to be conducted, asked to sign a form of informed consent (appendix 17), as well as to fill in a pretest questionnaire about the users' background (appendix 18). This would verify their qualification for participating in the test, provide better validation for the test itself (Dumas & Redish, 1999), as well as divide the users into two different target groups. The participants were also asked which language they would prefer to have the questionnaire in – English or Swedish – in order to avoid linguistic misunderstandings.

The questionnaires

The questionnaire was created in accordance with Oates (2006). The questions were kept as short, relevant, unambiguous, specific and objective as possible (Oates, 2006). The majority of the questions were closed, in order to not counter the point of the questionnaire delivering quantifiable data. However, they were on occasion complemented with open sub-questions, for example where the authors wanted to find out where in the system the users felt uncertain (appendix 19, question 7a) – if they felt uncertain – or what kind of functionality the users might have missed (appendix 19, question 17a). As for the format of the questions – with the exception of the open questions – these were made into 4-point Likert scale questions (Oates, 2006), where the answer called for deciding for example how well the users liked or disliked something, or how easy they thought something was to use (appendix 19, questions 1 and 2). However, a 5-point scale was used when questions required a possible neutral answer (appendix 19, question 4). Simple yes or no questions were constructed when the Likert scale could no be used effectively (appendix 19, question 14). The yes or no questions were often complemented with an open question, in order to gain a better understanding of the users' issues, if they had any. The reason for using a 4-point Likert scale is that, while the 5-point is the most commonly used form of the Likert scale (Cummins & Gallone, 2000), due to the limited size of the study, it could not be afforded to have a no-choice option. The reason for this is that a no-choice option not only indicates nothing – and thus would be harmful to the study as the answer as such cannot be used in the analysis – but also tends to draw attention from decisive answers in questions where users are uncertain (Dhar & Simonson, 2003). While a 4-point scale might be considered insensitive due to the few choices provided (Cummins & Gallone, 2000), a study has showed that increasing the number of points from 5 to 7 or 10 does not affect the mean outcome of the study (Dawes, 2008); thus a 6-point or 8-point scale would likely not be more beneficial. Furthermore, the study was meant to provide indicative answers and not sensitive ones, and it was felt that a 4-point scale would be the most appropriate in this case due to its simplicity.

The questions were structured into three main groups: 1) introductory, general questions related to usability, such as what the users thought of the system on a whole, and how easy it was to use, 2) frame-by-frame questions about the system, 3) market related questions to find out if the users would appreciate such a system, if it existed in reality. The only exceptions were questions 17 and 17a (appendix 19), which were general in nature, but placed after the frame-by-frame questions in order to not be confused with questions 11 and 11a. This provided a clear structure and logic to the questionnaire, and also secured a certain amount of content validity (Oates, 2006).

Pilot study

In accordance with Preece et al. (2007), Oates (2006) and Dumas and Redish (1999), both a brief pretest and a pilot study were conducted in order to verify the construct validity. The pretest – a consultation and discussion with other active people in the field – was held in a seminar with the supervisor of this thesis, and six other students four days before the real study. The pilot study, in turn, tested the questionnaire on a user that matched the target group two days before the real study.

The questionnaire was then altered slightly, on basis of the feedback received from the pretest and pilot study. The reliability of the questionnaire was however not tested, mainly due to the small scope of the project and the small size of the questionnaire study.

User Test

In practice, two usability tests were conducted. The first one, with four test subjects provided by the company, was held at Sony Ericsson's compound. These test subjects were, however, employees of Sony Ericsson, and were found to not match the target group as well as could be wished for, which is why the decision was made to conduct a second usability test. This increased the total number of test participants to eleven, and also provided more reliable data. The second test was performed outside of company compound, and with test subjects chosen by the authors themselves – mainly consisting of friend and acquaintances matching the target group – but apart from this the two tests were conducted in the same way. The test subjects were monitored while they were allowed to go through the predefined scenario of creating a blogging tool with the prototype, after which they were asked to fill in the questionnaire. As for the number of test subjects, Preece et al. (2007) recommend 5-12 users, while Dumas and Redish (1999) recommend 6-12. This placed the user test in the upper part of this range, providing a good amount of users without becoming too cumbersome to manage.

User Test Result Analysis

Once the tests had been conducted, the users were divided into two groups – one simple group and one advanced group – according to the information provided in the background questionnaires. People with programming and/or web design experience were placed in the advanced group, and the people without were placed in the simple group. This allowed for interesting comparisons between the two groups. The data from the post-questionnaires were juxtaposed in a spreadsheet program, in order to get an easily manageable overview of the results.

3.2.5 Final stage

In the sixth, final stage – after analyzing the data from the interactive user tests – adjustments were made to the prototype in accordance with the findings, and a final system design and strategy for dealing with the EUD gap was proposed, along with some recommendations for future research.

3.2.6 Usability Goals and Design Principles

Several usability goals were found to be of importance to the design project. The main usability goal that was focused on was learnability. Seeing as consumers are generally not educated in information technology and computer usage, although some are interested in those fields (Mårtensson, appendix 2), it can be assumed that their skill-levels will vary widely. The system must cater to all of them. It can be expected that users will be uncertain whether or not they are fit to use the program, therefore it is very important that the users quickly feel they have a good understanding of the system.

Customization of applications will most likely not be something that users do every day, or even every week. Therefore, memorability became another core usability goal. The users should feel safe to return to the system after not having used it for months, even if they have a different cellphone model than they had the first time.

Security is another important usability goal, seeing as users may have sensitive data on their telephones. To as large of an extent as possible, the authors wanted to use constraints to make sure the users could not delete, or somehow share, such information unless they explicitly wanted to. This issue was also important when it came to making the users feel safe in using the system itself; it should be very difficult, or impossible, to fail in the process of making an application.

Effectiveness, efficiency and utility were not the main focus of this project. Basic functionality, coupled with ease of use, was the main concern. Naturally, the authors nonetheless tried to be creative and design a flexible solution where features could be added later; giving the system more utility and making it more effective. The authors believed the system would be considered efficient if it could be made easy to learn, despite that this might make the system somewhat slower to use. As application customization would not be a daily event, and as the program was meant to be quick in itself, this would hopefully not pose much of a problem.

3.2.7 Component-based Approach

It was decided to proceed with a component-based approach, as the project manager explicitly stated that there were benefits to work with modularized systems (Mårtensson, appendix 2). The authors believe, in accordance with Wang and Qian (2005), that a modularized system based on components would give the system advantages in the areas of maintainability, reuse and overall dependability. Since the applications created by the system would have to be tailored for the various models of cellphones, and the various embedded operating systems, it was important to have a system where support could easily be added for new models, without having to change the rest of the system's architecture. However, the authors also believe that a component-based approach is less important for end users, seeing as modularization assumes a need for reuse (Wang & Qian, 2005). The modularization efforts were therefore focused on the architectural part of the system; that is, the back-end.

4 Developing the System

This part will explain how the system was designed in detail, and on what basis. It starts by presenting how personas were used to analyze the problem area, and how they were used to find solutions which were exemplified in design alternatives. One of these solutions was further developed into the prototype used in the user tests. Finally, the user test results are presented.

4.1 Analysis and Personas

After the initial interviews with key employees at Sony Ericsson, three personas were created to represent three types of users that had been identified: the young customizer Kate (appendix 5), the relaxed, practical user Logan (appendix 6), and the professional developer Brian (appendix 7). These personas were created in accordance with well known stereotypes, which makes them representative for whole groups of users, and distinct from one another (Cooper, 2004).

Kate was created with young customizing cell phone users in mind. She was based mainly on the interviews with Mårtensson (appendix 2) and Petterson (appendix 4), motivated by comments such as that the target user should be a user of the popular social networks like Facebook and MySpace (appendices 2 & 4). Kate belongs to a target group which today very actively customizes their cell phones when it comes to the look-and-feel – for example themes and ring tones – but who tomorrow will hopefully be customizing the functionality of their phones as well, for example with custom-made social networking applications. That is, as long as these are quicker, easier, and more fun to make than acquiring an existing application. Kate would be using the system mainly for her own sake.

Brian, on the other hand, was based on the interview with the third key employee (appendix 3), who unlike Mårtensson (appendix 2) and Petterson (appendix 4) was mainly concerned with reducing the time and resources today required from developers to make a third party application. Thus, Brian represents a target group which already creates applications for mobile phones, and possesses considerable programming skills, but who would prefer to create these applications in considerably less time. The main reason for him to use the system would be to sell the applications created. This places demands on the system to be advanced enough to suit his needs, while still being enjoyable to use.

Logan is a blend of Kate and Brian. Not in terms of competence – where his is on a level similar to Kate's – but in terms of reasons for using the system. The target group he represents is one that would use the system for its practical benefits; the ability to gain access to services in their own cell phones that perhaps would not otherwise be available, and the possibility to provide services to the common public, something that could not have been done without the system.

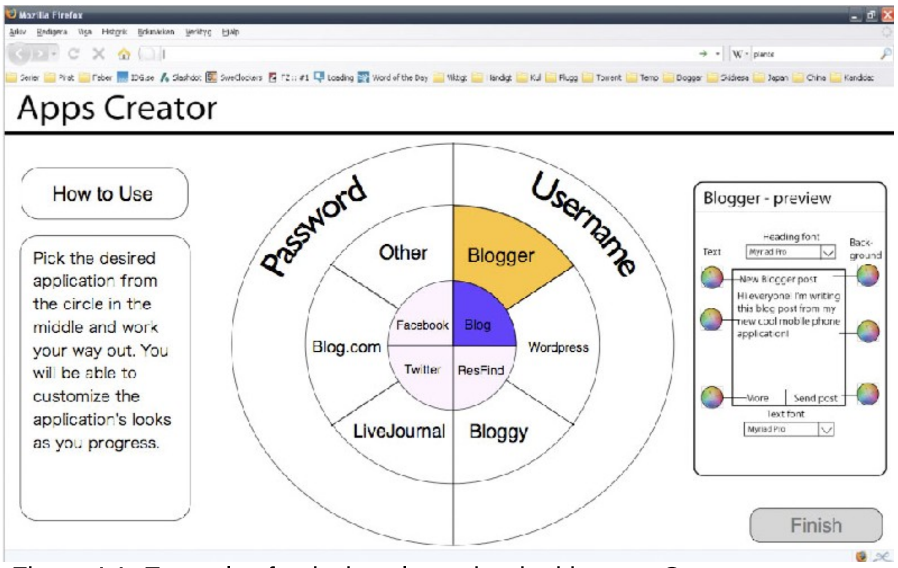
There were two main reasons for creating these personas. The first one was to concretize the target users. By providing three examples, feedback could be received from Sony Ericsson as to which of the personas they considered the most important. In this case, Logan was considered the long-term focus of the project, and Kate the short-term (appendix 8). This was also in accordance with the authors' own opinions in the matter.

The second important goal was that the personas made way for easier and clearer communication regarding the design alternatives, thus simplifying discussion and criticism. Based on the personas and what they likely would have wanted to create and use, several applications could be imagined. This provided examples to work from, when deciding what functionality to include in the system.

The personas became less important once the basic requirements were identified; especially Brian (appendix 7), seeing as only one of the key employees showed any will to emphasize such a user (appendix 3). However, the personas nonetheless remained important, and came up in meetings and discussions for the duration of the project.

4.2 Designing Alternatives

With the personas in mind, several design alternatives were created. When designing the alternatives, the main goal was not the look-and-feel of the system, but rather its functionality, and how this would be presented to the users. The alternatives were designed in brainstorming sessions using whiteboards, later refined in a digital illustration program.



Pick the desired application from the circle in the middle and work your way out. You will be able to customize the application's looks as you progress.

New Blogger post
Hi everyone! I'm writing this blog post from my new cool mobile phone application!

Figure 4.1: Example of a design alternative, in this case Cantonese

The alternatives were split into three areas: The back-end alternatives – *White Shark* (appendix 9) and *Manta Ray* (appendix 10) – which presented the architecture behind the system and how the created applications would work on the phone; the wizard alternatives – *Mandarin* (appendix 11) and *Cantonese* (appendix 12) – which show how a template application can be created in just a few easy steps; and the freeform alternatives – *Min* (appendix 13), *Xiang* (appendix 14), and *Wu* (appendix 16) – where a mode is presented meant to allow the users to be more creative in the development and customization of the application. The back-end alternatives were designed first, so as to give technical constraints when designing the other alternatives; the authors did not want to create an alternative which would not be technically feasible in reality.

The *Xiang* and the *Min* alternatives both have visual editors included in their designs, but since these visual editors would be very similar, they are treated as one.

Table 4.1: Overview of design alternatives

System Category	**System Codename**	**System features**
Back-end	*White Shark* (appendix 9)	Architecture based on components, optimized for various platforms.
	Manta Ray (appendix 10)	Architecture based on web content, such as HTML and JavaScript.
Wizard	*Mandarin* (appendix 11)	Wizard which allows a user to customize visuals and basic functionality in a few simple and short steps.
	Cantonese (appendix 12)	Same functionality as *Mandarin,* but designed to look more interesting and innovative by using a pie-chart design.
Freeform	*Min* (appendix 13)	A component based editor focusing on simplicity.
	Xiang (appendix 14)	A component based editor focusing on flexibility and functionality.
	Wu (appendix 16)	A component based editor which is a mix of *Min* and *Xiang*, giving the user slightly more flexibility and functionality while keeping the system simple enough for a beginner to use.

4.2.1 Back-end Alternatives

As mentioned, two back-end alternatives were designed: *White Shark* (appendix 9) and *Manta Ray* (appendix 10).

In *White Shark* (appendix 9), the system creates a configuration file that defines which components that make up an application; these could be buttons, text fields and similar components. As users connect to the back-end to download the application, the back-end looks at the configuration file to see what components are needed. The components are saved on the back-end, sorted by phone model. By checking the users' phone model at the time of access, the system can collect the correct components for the specific phone model, and pack them together into a complete application, tailored by the configuration file and the users' phone model. Other users could then find the application on PlayNow Arena (if the creating user has chosen to share it), where they can get links to obtain the application from the back-end.

In *Manta Ray* (Appendix 10), the system creates web content that makes up the application, such as HTML, CSS and JavaScript. When the users access the back-end to download the application, the back-end checks the model of the phone, and customizes settings in the web content such as the display size of the mobile. The back-end then compresses the configured web content into a special package, which the phone can run as an application. This provides a platform independent solution – assuming that a browser engine is created and distributed to the telephones, thus enabling the application package to run. As in *White Shark*, other users would be able to access the back-end from PlayNow Arena.

The first goal to be set up was that the system had to hide the fact that there are several mobile platforms; the users should not have to develop different applications for different platforms, and should hopefully not even need to know which platform they are personally using – be it Java, Windows Mobile or Android. This goal was met with both architectures. In *Manta Ray* (appendix 10), the whole problem is sidestepped by using a web content approach that depends on the specific engine in the phone, and thus not on the platform. In *White Shark* (appendix 9), this is achieved by the fact that the company behind the product will be forced to produce all components themselves, and make sure they work on all phones; but at least the users would be able to be oblivious to the technical workings of the mobiles. The negative side of the *Manta Ray* solution is that the web engine for the mobiles would have to be created and distributed, as it would be impossible to run the applications without it. This might, depending on the implementation, require users to actively get hold of said web engine. *White Shark*, on the other hand, requires the distributors of the system to code all modules and components themselves, a heavy workload, and a technical solution that may hinder third party developers from coding their own applications.

An interesting advantage of the *Manta Ray* (appendix 10) approach is, that since web content is an international official standard, many users will be familiar with its composition. This would allow more advanced users to code their applications outside of the system, similar to, for example, Mozilla's XUL project (Mozilla, 2009), thus enabling third party developers to add applications to the database.

4.2.2 Wizard Alternatives

The two wizard alternatives, *Mandarin* (appendix 11) and *Cantonese* (appendix 12), were both made with absolute simplicity in mind. In a matter of minutes and just a few steps, the users – in this case being aimed at mainly Kate – should have a finished application, ready to be exported to the telephone. *Mandarin* meets this need in a classic way known to, and recognized by, most computer users. Once they are logged in, they are greeted by a welcome screen that offers them a number of possible applications to create, for example based on the most popular applications. They are also given the option to go directly into freeform mode, with consideration to more advanced users. Choosing for example the blog application – an application for writing new posts to one's own existing blog – the users will first be asked to select the blog service provider their blog is hosted on. They will then be asked to fill in their username and password, in order for the application to know which account it should connect to. Once the bare minimum technical issues are out of the way, the users will be given the opportunity to customize the looks of their application. If they do not wish to do this, they can press next immediately, and will in such a case have a standardized look on their application. There was a discussion of whether this step should be included at all – for the sake of having as few steps as possible – but seeing as Kate is such a vivid visual customizer, she would most likely find the App Creator boring if she was not able to customize its visual aspects.

The final step in the *Mandarin* (appendix 11) wizard consists of: a preview of what the finished application will look like, the option to go into freeform mode with the currently made application, and naturally, the ability to finish the application, in which case it will end up in the users' "My Apps"-list; ready for export to their telephones. The "My Apps"-list could, for example, be incorporated in PlayNow Arena. *Mandarin* thus offers a logical and recognizable sequence of steps – most users will be familiar with the concept of wizards – that meets the requirements of effectiveness, learnability, efficiency and memorability (Preece et al., 2007). The safety aspect is also met, as the wizard is fail-safe; the users will not be able to delete anything by accident. The worst thing that could happen is that the users accidentally close the web browser, in which case the system would be able to remember the settings entered, enabling the users to continue where they left off when they log in again. Utility will naturally be rather lacking in the wizard; this is however not its purpose, and utility will be better met in the freeform mode.

With *Cantonese* (appendix 12) the authors wanted to leave the safety of an easily recognizable interface, in lieu of a more enjoyable and experimental user experience. After all, if the experience is not enjoyable, there is very little chance of the tool being used to any large extent (Mårtensson, appendix 2). Thus, the pie chart inspired design came to mind. Its sequence and number of steps is much the same as *Mandarin* (appendix 11); the difference lies in the way it is presented. The users will at first only see the explanatory box on the left, and the innermost circle providing them with the various applications. Note that in a real system, the number of choices would be more than the four available in the mockup. The users thus choose one of the applications, which in this example is the same blog application used in *Mandarin*. When an application is selected, the second circle will twirl and expand into place, providing the users with the choice of blog service providers. Once this choice has been made, the third circle will twirl and expand into existence in the same way the second did, asking the users to fill in their username and password. Having filled these in, the visual editor (in this case the same as from *Mandarin*; however, this could also be made into a fourth circle), is then displayed, followed by a final preview step as in *Mandarin*, once the users click the Finish-button.

The *Cantonese* (appendix 12) design was somewhat of a gamble, seeing as the users might feel insecure due to its differences in comparison with classical user interfaces. However, considering that this part of the system was mainly for Kate, it was perceived as a viable alternative; after all, Kate is not the type to be afraid to try new things (appendix 5). With regards to the usability goals (Preece et al, 2007), *Cantonese* meets these much like *Mandarin* (appendix 11) did, seeing as the number of steps, and their sequence, is the same. Safety is also maintained, thanks to the simplicity of the wizard. Once again, only utility is somewhat lacking.

4.2.3 Freeform Alternatives

If Kate was the main persona for the wizards, then Logan was the equivalent for the freeform alternatives. His technical expertise was roughly on the same level, but he would be more interested in making custom applications that the wizard did not support, for example, by making a surfing application that checked the weather at given surfing spots and recommended the best one.

A large issue with the freeform alternatives was how to keep the system simple enough for a non-developer such as Logan, yet at the same time offering significant functionality. A middle-ground had to be found. It was decided that the system should build on functional components that were somehow interconnected. Not only was this the EUD approach the authors had chosen to work with, but it also provided a good conceptual model for users since the back-ends were also planned to be component-based. The question was how large and encompassing these components should be, and on what technical level. The larger the component, the less freedom to make applications precisely as the users want them; the smaller the component, the more complex of a system, and the higher of a technical competence threshold to use it. Initial experiments with components such as RSS-reader, GPS-location, and Camera-input were made, however, these were all deemed too technical when considered for use by Logan or Kate. While most likely having heard of, and perhaps knowing roughly what an RSS-feed or GPS-location is, neither Logan nor Kate would know how it actually functioned, nor how to use it properly – at the least not without significant instructions. The choice was therefore made to lift the components to a higher level of abstraction. After all, if Logan wished to make a weather application, then he would not be thinking "I must start by aggregating several RSS-feeds from the correct weather stations", but rather "I want to make a weather application", and start looking for the weather component. Thus, instead of having technical components such as RSS-feed and GPS-location, components would already be aggregates of various technical components themselves, and merely require a certain amount of parameterization; for example, a weather component consisting of an RSS-reader, which can take in various weather-related data from several weather services, based perhaps even on the users' current GPS-location.

Min (Appendix 13)

The first design proposition for the freeform alternatives was *Min* (appendix 13). *Min*, in a way, built on the *Mandarin* wizard, in that its main focus was still very much simplicity, only offering more functionality than the wizard. It is based on three tabs offering different functions, namely Design, Functionality, and Code. Design (appendix 15) offers visual customization to a much larger extent than the wizard, Functionality offers the same for functionality, and Code is meant as a tab for developers – for example Brian – who wish to look at and/or change the code of the application. Code should, however, be seen as somewhat of a parenthesis; in fact, it was the first thing that was scrapped in the follow-up meeting with Sony Ericsson, due to the complexity it added not only to the possible implementation of the system, but also to the users who were not developers. It also violated the safety goal (Preece et al., 2007), seeing as users could potentially ruin the current application by making a wrong turn in the code tab.

The functionality tab of *Min* (appendix 13) is based on the concept of layers and frames. The users begin with the so called main frame, namely, the first screen of the application. They then choose a functional component from the sorted and grouped list on the left side of the screen, and are able to set the parameters such as – in the weather component case – location, time, and which weather values to display. This is then connected to the main frame, along with any other components the users might add; for example, a background component in between the main frame and the weather component. The users could also add other frames, and define how these should be accessed from the main frame in order to have more functionality. By clicking an existing component, information about it would be displayed in the upper right corner, and the relevant settings could be changed in the lower right part of the screen. This design proposal thus maintains a relatively simple level of interaction, while offering significantly more freedom of choice in comparison with the wizards. A drawback is that the utility could still be considered somewhat cramped in comparison with small, technical components; however, simplicity is deemed more important in this case. With the exception of the code tab – which, as mentioned, was scrapped at a later point – the system also maintains a high level of safety, not allowing the users to do anything that would make the application cease functioning; at the worst, the users would set a parameter wrong, in which case they could merely change it back. Accidentally deleting a component would naturally be reversible. As for learnability and memorability, a simple step-by-step tutorial, as well as explanatory comments – upon hovering with the mouse or clicking a question mark – would ensure easy learning and thus not require relearning upon later use; hopefully, the system is intuitive and logical enough to use even without a tutorial. Efficiency will naturally be somewhat lower than the wizard, but is still kept at a high level thanks to the minimum number of steps required to add functionality, and the easy drag-and-drop interface. The same thing is applicable to effectiveness.

Xiang (Appendix 14)

An issue that had been existent throughout the course of project – thus far – was the fact that all the experts interviewed wanted the applications created by users to be possible to sell on some sort of online market place (appendices 2-4), such as PlayNow Arena. It was therefore a large problem that, even though *Min* (appendix 13) offered a fair amount of freedom in the creation process, it was still not advanced enough to create truly unique applications. This meant that a market place would most likely be swarmed with almost identical applications. Eventually deciding that this issue was beyond the scope of this project, *Xiang* (appendix 14) was created in order to visualize this clash between usability, freedom of use, and salability. The design proposition is therefore of a rather technical nature, having smaller and technical components such as RSS-feed and GPS-location. This offers greater creational freedom, but also places the program on a completely different level in terms of prerequisite technical competence. In short, it could be seen as a design proposition built for aiding Brian in shortening the time and resources required to make cell phone applications, in comparison with the time and resources required to write everything in code (as expressed by the third key employee, appendix 3); an issue beyond the scope of this project since the authors aimed at bridging the EUD gap from the end-users' perspective. Thus, *Xiang* is more of a design proposition showing what not to make in this project, than a viable alternative.

Min & Xiang Visual Editor (Appendix 15)

The *Min* and *Xiang* Visual Editor (appendix 15) works in a very simple way in comparison to the *Xiang* (appendix 14) Component tab. Though drawn as part of the *Xiang* design, it also serves to show how the visual editor of *Min* (appendix 13) would work on a rough functional and conceptual level. It shows all the functionality that has been added to the application in the left side pane, a visual preview of the application in the main window, and the properties of the currently selected component or function in the right side pane. In the left pane, the users can turn the functions that have been added on and off, thus choosing what should be shown in the display. The preview window itself supports drag-and-dropping for reorganizing the way the visual elements are presented, and in the right pane the users can change the settings and parameters of the currently selected function. All this gives the users considerably more freedom of choice when designing the application – in comparison to the wizard – although the utility could still be considered rather cramped in comparison to programming. However, the simple interface makes for easy learning, and minimizes the need for relearning. Safety is also kept high, as this visual editor uses on-off buttons in order to show or remove things from the application, and the drag-and-drop is simply for reorganization, not deletion. While this may be somewhat hindering to effectiveness and efficiency, considering the use and overall scope of the visual editor in this case, this should not pose much of a problem.

Wu (Appendix 16)

In an attempt to streamline usage and to make *Min* (appendix 13) more flexible, the third alternative *Wu* (appendix 16) was designed. In *Wu*, the component and visual editors of *Min* have been combined into one editor, where functionality and visual elements are combined and placed on a preview screen of the eventual program. On the left side of the editor, functional components can be added, such as a weather component. This component is edited to contain relevant information; for example, the weather component has its location decided. After this initial setup, the users can drag various functions from the component. In the case of a weather component, it can be such functionality as the current degrees in Celsius at the given location, or the humidity in percent. These functions have to be connected to visual elements, which are added at the right part of the editor. Like functional components, these visual components are picked from a palette of components, and basic parameters such as color and font are edited. The elements are then dragged-and-dropped to the preview window. By dropping a function from a functional component onto a visual component, the two combine into an operational component; a visual representation of a function. This can be text showing the current humidity, a picture showing the last seen picture on Flickr, or a part of a song from Last.fm's latest hits.

Operational components are normally shown in a way that indicates which visual components and which functional components that have been connected. By clicking the preview button of the editor, the view can instantly switch to show a preview example, where operational components are displayed in the way they will look when the application is actually used on a mobile phone. This solution allows the users to quickly see what their application will look like, without making operational components harder to edit; something which would be the case if an operational component did not show which functional and visual components that had been combined to create it.

As with *Min* (appendix 13), *Wu* (appendix 16) relies on previously made components that have already been set up to contain relevant functionality. Similarly, there might be a risk that applications created in *Wu* are too similar to one another. However, the authors felt that the approach allowed the users to create very individual applications, as the functionality of several components can be displayed in one frame. One could thus take a picture from Flickr and display on the same page as the current temperature in Sydney. This flexibility allows users to be more creative. While the applications might not be worth paying for, at least they would be very personal, and could hopefully be created to meet specific needs.

While *Wu* (appendix 16) lacks the utility of *Xiang* (appendix 14), it is considerably easier to use. It has a slightly steeper learning curve than *Min* (appendix 13), but seeing as visual and functional editing are both contained in one window, it was felt that the users might consider it easier to get a good overview of the capabilities of the system. The authors believe this one-window solution makes the editor better in the areas of learnability, memorability and efficiency. The users would quickly learn how to use that one window, and there would be no other windows to be concerned with. Less windows to remember equals better memorability. Having everything in one window, without making it cluttered, also saves the users time as they will not have to switch between views. The utility and effectiveness of *Wu* is similar to *Min*, but improved. There is more flexibility, and more advanced applications can be created.

4.2.4 The Prototype (Appendix 20)

The prototype created was simply a more colorful version of the *Mandarin* (appendix 11) alternative. It was created to look more like a proper web application than simple concept art, although the alternative is of course still just a mockup, and not supposed to look like a real implementation.

Technically speaking, the prototype consisted of the refined pictures of the *Mandarin* (appendix 11) alternative, put into a slideshow presentation with invisible buttons placed over the interactive parts of the images. This allowed for the quick creation of a prototype, which in turn enabled the user tests to commence. The problem was that only the intended functionality worked, possibly making the prototype slightly easier to use than an actual system. However, as the authors observed the users' interaction, it would have been noticed if the constraints helped the users. Therefore, the authors do not believe this harms the results of the prototype tests. Seeing as text fields could not be implemented in the slide show program, it was vocally explained how the users would insert their username and passwords. However, most users understood this without any explanation.

During the pilot test, several small issues with the prototype were found, which were corrected before the real user tests. For example, more help text was added to the steps. A few of the application names in the welcome screen were also changed, as they were found to be vague. Some applications had had concrete names like “Facebook” and “Twitter”, while other applications were named in a sense which implied they were in fact categories, such as “Blog”. In the final prototype, these were changed so that all of them had the sense of being categories.

The applications in the welcome screen were picked to be representative of the abilities of the system. The RSS-Reader was specifically added to see if, and how, users would react to it.

4.3 User Testing

Derived from the background questionnaires (appendix 18), two groups of users were defined; the advanced users, who would likely be interested in more advanced functionality, and the simpler users, who would want the system to be as easy to use as possible. The main difference between the two groups, is that the advanced group includes programmers and people who know basic HTML, while the simple group consists of users who have no knowledge of such technologies. The advanced group mainly consists of the employees from Sony Ericsson that participated in the user test. The results of the questionnaires can be seen in table 4.2.

Table 4.2: Showing the answers from user test questionnaires, sorted by question and group

Question (Possible Answers)	**Advanced User Group**					**Simple User Group**					
	1	2	3	4	10	5	6	7	8	9	11
1. What did you think of the program?											
(1-4)	2	3	3	3	3	3	3	4	3	4	3
2. How easy or difficult was it to understand what you were supposed to do in the program?											
(1-4)	3	3	3	1	4	4	3	4	3	4	3
3. How easy or difficult did you find the program to use?											
(1-4)	3	3	3	3	4	4	3	4	3	4	3
4. What did you think of the number of steps required to make an application?											
(1-5)	3	3	3	2	3	2	3	2	3	3	3
5. Did the buttons do what you expected them to do?											
(Y/N)	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	N
6.Did you ever feel that there was not enough help documentation in the program?											
(Y/N)	N	Y	Y	Y	N	N	N	Y	N	Y	Y
7. Did you ever feel uncertain of what you were supposed to do?											
(Y/N)	Y	N	Y	Y	N	N	N	N	Y	Y	N
8. Did you notice the process bar (which showed which step of the process you currently were in) that was displayed in every step?											
(Y/N)	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y
8a. If yes, did you find it helpful?											
(Y/N)	N	Y	N	Y	Y	Y	Y	N	N	Y	Y
9. What did you think of the number of alternatives available in the main menu?											
(1-5)	4	1	5	3	3	4	3	3	3	3	3
10. How clear did you find the names of the applications to be?											
(1-4)	2	2	3	3	3	3	3	4	3	3	3
11. Did you miss any kind of application?											
(Y/N)	Y	Y	Y	N	N	N	N	N	N	N	N
12. What did you think of the number of choices in the blog service menu?											
(1-5)	2	2	2	3	3	3	3	3	3	3	1
13. How did you feel about entering the username and password of your blog?											
(1-4)	2	3	3	3	3	4	3	4	3	3	2
14. Would you prefer to enter the username and password of your blog in your cell phone each time you wrote a post?											
(Y/N)	Y	N	N	N	Y	N	N	N	N	N	Y
15. How interesting did you think it was to be able to change the visual appearance of your application?											
(1-4)	2	4	1	4	3	4	3	4	3	4	3
16. Did you understand how the visual tool worked?											
(Y/N)	N	N	N	N	Y	Y	Y	Y	Y	Y	Y
17. Are you satisfied with the functionality that is offered in the program as a whole?											
(Y/N)	N	N	Y	Y	Y	Y	Y	Y	Y	Y	N
18. Would you use such a program if it existed for real?											
(Y/N)	N	N	Y	Y	N	N	Y	Y	Y	Y	N
19. How big of an implication would the availability of such a tool have on your purchase of a new cell phone?											
(1-4)	1	3	1	1	3	1	2	3	2	2	1

4.3.1 General (Questions 1-8)

Overall, the simple users considered the system to be good and easy to use and learn. The advanced users, though not as positive as the simple users, similarly found it good and easy to use and understand. Both groups found the amount of steps required to create an application to be just about right, and they agreed that the buttons in the system worked as expected. Both groups also agreed that there were times where more help documentation could have been useful. However, during the observation, it was noted that many users simply did not read existent documentation because it was not presented properly. The majority of simple users never felt uncertain what to do, but 3 out of 5 of the advanced users did. Almost every user noticed the process bar, but only a slight majority found it helpful.

4.3.2 Welcome Menu (Questions 9-11)

As can be seen in the table, both groups found the number of alternatives in the welcome menu to be just about right. The advanced users found the names of the applications to be a bit less clear than the simple users, but both groups generally found the names to be clear. Not a single simple user found any specific application to be missing, but 3 our of 5 of the advanced users did.

4.3.3 Blog Service (Questions 12)

Both groups found the blog service list to be just slightly less comprehensive than they would have liked. Only one person said there were “too few” choices to pick from, although the majority of advanced users said there were “somewhat too few”.

4.3.4 Username/Password (Questions 13-14)

The majority of all users considered the input of username and password to be safe, and preferred to enter the username and password in the creation process rather than being asked for it each time they used the application. The advanced users were slightly more open to the idea of entering their usernames and passwords each time than the simple users.

4.3.5 Visual Settings (Questions 15-16)

Although both groups found it interesting to edit the visual appearance of the application, the simple users found it slightly more interesting than the advanced users. With the exception of the Sony Ericsson employees, all test subjects found the visual tool easy to understand.

4.3.6 General Functionality and Market Related Issues (Questions 17-19)

The majority of users were found to be satisfied with the overall functionality of the system, especially the simple users. Where the majority of simple users would use a system such as this if it existed, the majority of advanced users would not. The plurality of users considered the potential existence of a system such as this, to have no impact at all on their purchase of a new mobile phone.

4.3.7 Observational Results

Observational notes were kept as a complement to the post-survey questionnaires in order to cover things the questionnaire could not cover, such as users forgetting to fill in something they thought of earlier on, or things that came up in discussion. The notes mainly provided further support for many of the issues that were brought up in the questionnaires, but also highlighted a few other issues; for example the fact that quite a few users did not bother to read the instructions on screen, or that it took a while for some users to understand the visual tool – although once they did, they found it very easy. A list of all the observational notes can be seen in appendix 22.

4.3.8 User Test Flaws

In spite of utmost attempts to keep the user tests professional, relevant, and without glitches, a few issues still came to the authors' attention when all the tests had been conducted. The most important one, was perhaps that not all users fully understood the context in which the scenario took place; one user, for example, thought that the user details screen for the blog account was the login for the system itself. Although the context – that the user, according to the scenario, had already logged in to the platform in which the system existed – was explained at the beginning of each session, it seems this was not enough, and further explanation was sometimes needed during the test itself, in order to elucidate these issues. Therefore, in hindsight, a login screen should perhaps have been included before the users were allowed to interact with the actual prototype; merely in order to have the users understand the context better. Related to this is also the issue that some users did not seem to understand, that the scenario of creating a blog application was merely an example of what the system could be used for, as well as the fact that some users thought they were supposed to fill in the questionnaire as if they were still acting as the fictional person in the scenario. Further clarification and explanation beforehand might have remedied both these issues. As it happened, this was explained to the users as soon as the authors understood that there was a problem.

Lastly, the questionnaire was found to be somewhat blunt in certain aspects, mainly with regards to questions 8, 14, and 16. Question 8 did not ask if the users thought the process bar was harmful or annoying, and question 14 did not address the possibility if whether the system asking for the users' account details to be saved, would be of any interest. Question 16 did not take into consideration that perhaps the users initially felt uncertain, or that it took some time, before they eventually understood how the visual tool worked. While the issue of question 14 unfortunately went unnoticed until the end of the user tests – with the exception of one user who specifically asked for this solution – questions 8 and 16 were in fact covered by the observational notes, therefore not being an issue. Overall, the authors feel that the issues encountered were dealt with in the best possible way, considering the circumstances, and that they should not affect the validity of the results of the user test.

5 Discussion

The following part begins by discussing the empirical findings in further detail, as well as their implications on the final design proposition. This is followed by discussions on the use of components, interaction design, and End User Development, in this project.

5.1 Discussion of Empirical Findings

5.1.1 General (Questions 1-4)

Questions 1-4 were meant to test the overall satisfaction and user interaction with the program, and the results from these indicate that, when it comes to the general impression of the program (Question 1), how easy it was to understand what to do (Question 2), and how easy it was to use (Question 3), the users were satisfied. Based on this, the conclusion was drawn that the program was worthy of further refinement, instead of being scrapped completely and replaced with something new. As for question 4 – what the users thought of the number of steps required to make an application – most users considered it to be “just about right”. This was somewhat surprising, as the authors had strived for the application to contain as few steps as possible in order for it to be as easy as possible. Three people – two from the simple group and one from the advanced group – had circled that there were “somewhat too few steps”. However, interestingly enough, none of these users thought the program lacked any functionality (Question 17), and the observational notes shed no light on this matter either. This makes it difficult to understand what kind of functionality any possible extra steps would contain. Due to this, and the relatively small size of the deviation, this issue was not taken into account in the final design proposition.

There were, however, a few other deviations from the norm as well. In question 1, for example, one user from the advanced group thought the program was “bad”[1], but the same user also thought the program was both easy to understand and use. The reason for this may be that the user wanted to be able to do much more than the program allowed him/her to do; something which is supported by the authors' observational notes (appendix 22). This issue was taken into account in the strategy (see part 6.2) proposed to deal with the gap. It did, however, not affect the final design proposition, seeing as it contradicts the point of the program's ease of use, which the same user, after all, thought was satisfactory. In question 2, once again, one user – also from the advanced group – thought the program was less than “easy” to understand, and considered it being “very difficult”. The reason for this, is that the user thought the program itself was merely an introduction with instructions for the real program, which the user expected would come afterwards (as can be seen by the questionnaire

1 All other users thought the program was either “good” or “very good”.

comments in appendix 21, and the observational notes in appendix 22). The misunderstanding sprung from the notion that the prototype looked like mere images – which they were – an issue the authors believe would be remedied automatically if the program is actually implemented. In that case, it would no longer look like images, but rather like the interactive interface it would be. Questions 3 and 4, in turn, had no deviating results, and all users thought the program had been either "easy", or "very easy", to use, and had circled either "somewhat too few", or "just about right" for the number of steps.

5.1.2 Buttons (Questions 5 & 5a)

In question 5 – did the buttons do what the users expected them to? – only two users answered "no", one from the advanced group and one from the simple group. Both of these were confused by the visual editor; a step that, according to the observational notes (appendix 22), was also the cause of confusion for a couple of other users. This visual editor was therefore remade in the final design proposition.

5.1.3 Help Documentation (Questions 6 & 6a)

Question 6 – whether there was not enough help documentation – returned the somewhat alarming result that more than half of the users thought that this was the case. In the advanced group, 3 out of 5 thought so, and in the simple one, half of the users did. The comments received in question 6a from the people who thought so, did not manage to shed much light on the matter; instead it was the comment from one user who answered "no" to this question that explained it, together with the observational notes. This comment says that "It was there but could've been clearer" (appendix 21). The observational notes supported this, and showed that while in many cases there were seemingly adequate instructions, many users simply did not read them, especially when the text volume was larger. This issue was therefore dealt with in the final design proposition.

5.1.4 Uncertainty (Questions 7 & 7a)

Question 7 showed that 5 out of 11 users at some point felt uncertain what to do. What is interesting about this question is the difference between the groups. In the advanced group, 3 out of 5 were uncertain at some point, while in the simple group, the equivalent number was 2 out of 6. The reason for this is unknown, but the authors are glad that it is not the inverse, seeing as the prototype program was aimed at the simple group. This difference could, however, be related to the results of question 16, where all four users that were employed by Sony Ericsson answered that they did not understand how the visual tool worked, while all of the other users understood it; an interesting result that the authors do not know the reason for. It can be speculated that perhaps the tool is too simple for advanced users, or that they are used to more advanced programs, and therefore read more functionality into the visual editor than what is really there; thus becoming confused when it does not work the way they expect it to.

Returning to question 7, the comments in 7a show that 3 out of the 5 people who felt uncertain at some point, did so in the visual editor. One of the two remaining ones was the user who thought the whole program was an introduction, and the last one wondered if the application he/she was making was meant for his/her own personal use, or for the use of other people. However, as the observational notes showed, the main reason for this last user's uncertainty was also the visual editor, seeing as he/she did not understand why one would want to customize the looks of an application if it was only for personal use. This is a very difficult thing to amend with the program, although greater care could, and most likely should, be taken when explaining to users which applications are meant for personal use, and which are not. Overall, however, the authors feel that the confusion regarding the visual editor was an issue large enough to warrant a remake of the tool.

5.1.5 Process Bar (Questions 8 & 8a)

When it comes to the process bar (Questions 8 and 8a), there was only one person from the simple group who did not notice this. Observational notes showed that this did, however, not affect the user's interaction with the program in any way, and question 8a showed that a majority of users did find the process bar helpful. In the case where users noticed the process bar but did not find it helpful, the observational notes showed that this was mainly because the users felt the creation was so short that it did not warrant a process bar; they did, however, not find it disturbing or annoying. This led the authors to believe that it was right to include the process bar, although arrows would have been better suited for this cause than the boxes and lines that were in the prototype, as was suggested by one user. Such a thing would better show that the process bar represented a flow, and would avoid implying that the bar was clickable; something which one or two users thought.

5.1.6 Welcome Menu (Questions 9-11)

In the welcome menu, most users thought that the menu had “just about the right” number of alternatives (Question 9). Two people – one advanced and one simple user – thought it had “a few too many” alternatives, one advanced user thought it had “too many”, while another advanced user thought there were “too few”. The latter advanced user was a programmer, very similar to the persona Brian, who wanted much more functionality in the program, and therefore also more alternatives in the welcome menu. Once again, this was not taken into consideration in the final design proposition, as it contradicted the very nature of the program itself.

As for the users who thought there were “a few too many”, or “too many”, alternatives, two of them were from the advanced group, which is not the main aim of the program, leaving a small enough deviation to be ignored. The welcome menu would still be radically remade, though mainly in accordance with the results from questions 10 and 11, and the observational notes. Question 10 showed that almost everyone thought the names of the applications were clear. There was, however, an interesting difference between the advanced group and the simple group, in that the advanced group had two users who thought the names were “unclear”, while all users in the simple group thought the names were “clear” or “very clear”. This could be due to the fact that the advanced users have a better understanding of all the things that the categories in the main menu could denote and include, while the simple users perhaps understood what the words meant, but did not contemplate further on their implications. If this is true, then even simple users would most likely be confused when they start interacting with the other applications, being surprised by what the application actually does.

Question 11 showed that only three people from the advanced group missed any kind of application. What they missed were non-internet based applications and image processing applications – the creation of which the system should naturally support – as well as one user who wanted to make games and animations. Unfortunately, these two things are beyond the scope of the system, due to the radical difference between simple applications and animations and games. Although none of the simple users expressed that they missed any kind of applications, the authors still believe the users might not realize the benefits from a certain application, until they discover that the application could actually be made. The authors think that the commercial success of a tool such as this heavily relies on there being a greater number of applications available. This would thus appeal to the needs and wants of as many users as possible. Also, the observational notes showed that as much as three users from the simple group, who thought the names were clear, still did not understand what an RSS-reader was, supporting the decision to lift the general applications to a non-technical level. Based on these facts, it was deemed that the welcome menu needed a radical change.

5.1.7 Blog Service (Question 12)

In question 12, most users thought that there were just about the right number of alternatives in the blog services menu, with the exception of three advanced users who thought there were “somewhat too few”, and one who thought there were “too few”. Seeing as the creation of a blog application is merely an example of the numerous applications that the system should be able to create, this does not have much of an implication when it comes to the overall system. It is merely concluded that, in accordance with the previous paragraph, the system should support as many blog services as possible in order to be able to cater to the needs of as many users as possible.

5.1.8 Username/Password (Questions 13-14)

The results of the username/password screen does have a rather large implication on the system as a whole, seeing as many applications would be internet based and require the users to fill in their account details. While the mean value of question 13 showed that users felt it was safe to fill in their username and password, two users – one from the simple group and one from the advanced group – still felt that it was unsafe. Considering the safety aspect related to – albeit temporarily – storing username and password on a central server, the authors felt that this was an issue large enough to warrant attention. Upon closer examination, it was concluded that there was in fact no technical need for the username and password to be entered in the creation process of the application. Instead, the authors were appealed by the proposition put forth by one of the testers: to remove the account detail handling from the application creation process, move it to the cell phone, and ask the users the first time they enter account details if they wish to save them (appendix 22). This way one avoids the security issue of storing usernames and passwords on a central server. The solution allows users who do not wish to enter their account details every time they use the application to save the details, yet does not force more security-prone users to save their details anywhere.

In Question 14, even though only 3 out of 11 said they would prefer to enter their account details in their telephone every time they use the application, the authors believe the mentioned solution satisfies both types of users, yet still makes the system much safer to use.

5.1.9 Visual Settings (Questions 15-16)

Question 15, clearly shows that users are interested in customizing the looks of their application, even if it is only for personal use. There is, however, a significant difference between the advanced group – where one user thought it was “uninteresting”, and one thought it was “completely uninteresting” – and the simple one – where all users thought it would be either “interesting” or “very interesting”. One of the advanced users who thought it was “uninteresting” or “completely uninteresting”, was the one who did not find any value in personal customization of visual aspects, and the other one would have preferred to control what the blog post would look like on the blog, instead of the just the looks of the application in the telephone. The first issue has already been discussed in the paragraph on deviations in questions 1-4, and the second issue, while theoretically possible, is of a technical nature and mostly related to the blog application itself; not so much the system on a whole. The possibility of changing the visual aspects of the actual blog post should, however, be taken into consideration when a real back-end blog application is developed for the system.

Question 16 was discussed in relation to question 7, and will therefore not be discussed any further.

5.1.10 General Functionality (Questions 17 & 17a)

Question 17 showed that 8 out of 11 users were satisfied with the overall functionality that the system offered. Two out of the three users that were not satisfied were from the advanced group, and the third one was from the simple group. One of the advanced users misunderstood the question, and answered that he/she would like to be able to see previous blog posts in the blog application – an issue which, much like the possibility to change the visual aspects of the blog post, should be taken into consideration when a real back-end blog application is developed. The other advanced user wished for “More creativity when it comes to graphics”, and the simple user wanted to be able to add his/her own videos, images, and external links, as well as making text underlined, bold, or italic. The same user also did not notice the function for changing fonts. All of these things provide further support for the needed remake of the visual tool that can be seen in the final design proposition.

5.1.11 Market Related (Questions 18-19)

When it comes to the market related issues, only a scarce majority of the users said they would have used such a tool if it existed for real. There is, however, a large difference between the advanced and simple group here. Merely 2 out of 5 users in the advanced group said they would have used such a system if it existed for real, while the equivalent number in the simple group was 4 out of 6. The comments from the users in the advanced group, who said they would not use such a tool, varied widely. One user said he/she felt unclear of what he/she was supposed to do, and what the results would be; another user said “I can write my own java midlet [application] instead” (appendix 21). The third of the no-sayers in the advanced group said that he/she still preferred to do most such things on the computer, but if he/she were to become more mobile in the future, then a tool such as this might be of interest. With regards to the first of these comments, this is hopefully easily solved by better instructions and clarification in the system as to the whats and hows – this will be dealt with in the final design proposition. The second comment is beyond the scope of the system – its aim is not to replace classical programming, but to enable non-programmers to customize existing applications, and thus also to customize the functionality of their telephones. The third comment is also beyond the scope of the system, as this is an issue that lies with the personality and habits of the user itself. The system can merely hope to provide enough variety among its applications so that even such a user would find something of interest – an argument further supporting the remake of the welcome menu.

As for the two simple users that would not use the system, the first one of these wrote “I don't blog”, resulting from a misunderstanding, thinking the system was only for making blog applications such as the one in the scenario. This issue would most likely be remedied with a new welcome menu that more clearly showed the vast variety of applications at the users' disposal. The second user wrote “I'm busy having a real life instead of a cyber one”, indicating that he/she did not realize the system was not only meant to create applications for web use, similar with the advanced user who missed non-internet based applications. Once again, a new welcome menu, which better shows the variety of applications that can be made, should be able to address this issue. As for the users who would use such a system if it existed (6 out of 11), the authors consider this number to be adequate to support the hypothesis that users in the gap would like to have this kind of functionality; especially when taking into account the simplicity, and very limited functionality, of the prototype.

Lastly, question 19, with most users saying that such a system would have none, or a small, implication for them when buying a new cell phone, indicates that on a strategic level, the system should most likely not be used to market new cell phones, but should rather be marketed itself with the help of new cell phones, aiming at providing complementary services that add extra value to the purchase.

5.2 Interaction Design

In designing the system, the authors discovered that the interaction design process (Preece et al., 2007) was a great asset. It provided a well-needed structure for the design process, yet allowed enough freedom for the creativity to express itself. The first stage – “Identifying needs and establishing requirements for the user experience” – was, as mentioned in the method chapter, divided into two parts: one data gathering part and one requirement establishing part. In practice, it was merely a symbolical division, in order to clarify the process; especially as the authors decided to use personas in order to establish the requirements. These two stages provided a solid base to stand on for the rest of the project, which together with the following steps allowed for effective creation of initial design propositions, an interactive prototype, and a final design proposition. Without this design process, the project would most likely have gone astray, and eventually failed to produce anything of value. One can naturally claim that it is very difficult to know if it was this specific method that allowed for the successful completion of the project, or if any similar systems development method would have sufficed. That is, however, a question beyond the scope of this report, and the authors believe that a method which is not as experimental and evolutionary as the one proposed by Preece et al. (2007), would have had a disadvantageous effect on the project, or at least the product of the project. This likely holds true for similar projects and products.

Especially the use of personas in combination with goals (Cooper, 2004) and user stories (Cohn, 2004) were tremendously useful in the early design process. By being described in such a detailed way as they were, Kate, Logan and Brian truly became like real people, thus providing a basis for very fruitful discussions, and the inclusion and exclusion of many different features and parts of the preliminary designs. For example, when Brian was still considered a potential candidate for use of the system, his existence motivated the inclusion of the Code-button in the *Wu* design. Likewise, the focus on both Kate and Logan made the authors realize that the system would have to be lifted above the level of technical components that had first been in mind, something which in the user tests turned out to be a very good choice, as quite a few did not know what an RSS-reader was.

When the initial design propositions were developed, the use of mockups was also an invaluable tool. By first sketching very crude versions on a whiteboard, the first, basic concepts could be created, refined, and eventually finalized in a digital drawing program. The whiteboard meant that the authors could draw, redraw, and erase each other's ideas, which was very beneficial to creativity and the designs. The mockups were thus an easy and quick way to conceptualize ideas and design propositions, and allowed for more time designing, as well as less time drawing, describing or building. They were also effective in communicating the concepts to other people. Likewise, by simply combining a refined version of one of the preliminary designs with a slideshow program, the authors were allowed to spend more time being creative, and less time building an interactive prototype. An initial plan was, in fact, to create a functional prototype in Java. However, it was realized that this would require considerably more time than a slideshow, while only offering marginal benefits; a time investment that could not be afforded within the time limits of the project.

The use of usability goals (Preece et al., 2007) in the design process was beneficial as well. Partly in the creation of personas, where it enabled the discussion of which usability goals that would be the most important to which personas, but mainly, it helped by providing a clear goal and aim for the system, in a similar way that the personas did. This held true especially when the designs were evaluated in relation to the usability goals. Had it not been decided to focus on learnability, memorability and safety, instead focusing on for example efficiency and utility, or not using usability goals at all, the system would most likely have looked considerably different. Especially *Mandarin* might have become encumbered by additional features and steps; this does however most likely hold true for *Wu* as well.

It is the authors' belief that no single one of these factors is solely responsible for the successful completion of the project. On the contrary, it is the combination of them all that is the most important part, including the component-based approach, which, as mentioned, had several practical implications.

5.3 Components

In the final system proposal, components were assigned a rather downplayed role, but the approach has nonetheless been relevant to the project as a whole. Both proposed architectures, *Manta Ray* and *White Shark*, are built in a component-based manner, where reusability and modularization is important and supported. The authors found component-based solutions to be very flexible when designing the various freeform alternatives as well, especially in the *Wu* alternative. The freeform alternatives are can therefore be considered a form of the component-based approach in EUD (part 2.4.1). Thus, the component-based approach would be relevant to the developing company in the form of the back-end, and relevant to the end users in the form of the freeform alternatives.

If the system was to be implemented, the authors believe a component-based architecture such as *Manta Ray* or *White Shark* would grant the providing company flexibility when it comes to adding features and opening the system up to other developers. In an architecture where each application has been made beforehand, and the application simply customizes itself from a configuration file, problems could arise from this lack of modularization. For example, if a company behind a product such as this would want to let other developers create applications, the developers would have to learn how the company itself makes applications customizable; e.g. how a configuration file should be written. Otherwise, the new applications would not be compatible with the system, and would lack customizability. In a component-based architecture, however, the distributor would define how each component could be customized independently, while in a situation where a component was not made to be customized, it would only affect that particular component and not the whole system.

The three principles of modularization proposed by Baldwin and Clark (2000) (as discussed in part 2.3), have all been incorporated in the designs. The interface to the components is almost invisible to the users; even in the *Wu* alternative where direct manipulation takes place. The functionality of the component is shown to the users, but how it works is hidden; the company in control of the system can therefore update and change the components in any way they wish without the users noticing, as long as the same functionality is available through the interface. The system is also abstracted; the users creating an application do not need to know whether the back-end is *White Shark* or *Manta Ray* based, or whether the components are implemented in Java or some form of web content, such as HTML. The users only have to focus on what functionality they want, and the technicalities of the system implementation are left to the providing company.

By applying the five principles by Wang and Qian (2005) presented in the theory chapter, one sees that:

Components represent decomposition and abstraction: In both *Manta Ray* and *White Shark*, the problem of creating flexible applications is solved by splitting the functionality into components, making each component a small problem instead of each application a big problem. In *Wu,* as well, functionality is abstracted into components, which group functionality and visuals in a logical matter.

Reusability should be achieved at various levels: In *Wu*, if one knows how to use a component, one can use it in several situations, without having to relearn how it works. In a back-end like *Manta Ray* and *White Shark*, one component could be used to create several applications, and the same specifications and documentation for those components would still be valid, for example, the documents explaining how the component could be visually customized.

Component-based software development increases the software dependability: This, too, is true in both *Wu* and the back-end alternatives. As users work with *Wu* components, problems and limitations would be found, and these would then be possible to improve, thus constantly making the system more dependable. As applications are created automatically in the component-based back-ends, problems would be logged and could be solved by the distributor, making the whole back-end more reliable. Every new application made from old components would be guaranteed to work as well as the previous applications.

Component-based software development could increase the software productivity: Instead of having to code a new application from scratch, users – using *Wu* – and the distributor of the system – using the back-end – could create new applications by simply creating new combinations of components.

Component-based software development promotes software standardization: In a back-end like *Manta Ray*, both the creation and composition of several components could be standardized. By doing this, third party developers could be allowed to create their own components, making the system very flexible. Each new component would create a lot of new possibilities, as it is connected to old components. In a system like *Wu*, the functionality the users have access to would grow exponentially as a result of such standardization.

5.4 End User Development (EUD)

The authors believe that the designed system has formed into something which is less flexible and complex than actual EUD, yet more interesting and functional than what normal end users are accustomed to. Where the *Wu* alternative leans more towards proper EUD, the wizard proposals lack the ability to create completely new functionality, and are therefore better seen as a form of parameterization, in accordance with Lieberman et al. (2006). The reasons why none of the EUD approaches mentioned in the theory chapter were used, are several. Early on, it was decided that scripting languages were too advanced for end users to learn; it is the authors' belief that most end users could not tell the difference between a scripting language and a normal programming language. The authors therefore place scripting in the extreme stages of EUD, whereas the aim was for the other end of the EUD gap identified in the background. Visual programming languages were considered to be difficult to develop, and a fitting one could not be found. One could however argue that the *Wu* alternative, and especially the *Xiang* alternative, contain elements of visual programming, where simple structures and functionality can be created by interacting with visual elements. In both alternatives, combining functionality gives an analogous representation of what the application will look like when used on the mobile phone; much like Kindborg and McGee (2007) recommend. Programming by example was not found to be fitting for the system, seeing as it is difficult to define the examples. How would one show a computer to take an RSS-feed from a website and turn it into proper weather information? Even if one could make an interface where users were able to perform such examples, how would the computer generalize such an action? The *Wu* alternative was found to give similar functionality, without having to develop a working programming by example system. There would also be difficulties in testing such a system with end users.

The authors believe this simple kind of EUD is needed when bridging the EUD gap identified in the background. Proper EUD is highly beneficial in situations where employees at a company can be taught how to customize and create new functionality for the company's system (Lieberman et al., 2006). In such a situation, the employee is paid for learning how to perform these tasks, and it is also something the employee may use very often. In the case of an end user like Kate (appendix 5), however, the end user most likely does not have the motivation to learn real EUD techniques like natural or visual programming. Such a user might simply give up and never experience the creation of personal software, or the practicality of customizing functionality. Seeing as this type of user is the focus of this thesis, it was decided to have a much simpler system as the final design proposition. However, the authors believe that the user could hopefully be guided over the EUD gap by gradually offering increased possibilities at the cost of complexity. For example, by starting with the *Mandarin* alternative, and slowly converting to the *Wu* alternative. When the user feels at home in the *Wu* system, proceeding to more advanced EUD techniques should not be as big of a step.

6 Final Design Proposition and Recommendations

This chapter begins by presenting the final design proposition – *Mandarin 2.0* – followed by recommendations and conclusions on the academic and commercial implications of the project. It ends with the final conclusions drawn by the authors.

6.1 Final Design Proposition

As can be gathered from the discussion on the findings of the user tests, several parts of the prototype were found to need improvement. As a result of these improvements, the authors feel that the final design proposition is considerably easier to use, and much more flexible. Virtually every step of the process has been improved, and superfluous steps have been removed.

Appendix 23 contains images of the final design proposition. First of all, it was decided to change the welcome screen completely. It now includes more applications, and could easily be extended with even more. Instead of using a category system – where a user picks a type of application and then general options – it was decided to list individual applications directly, while keeping them in categories that could be browsed if the user did not know exactly what specific application to use. In lieu of having an undefined list of applications with a “more”-button, two lists were made: Popular applications and Recommended applications. The popular applications list is, of course, based on the most often created applications. The recommended list, on the other hand, is meant to be maintained by the distributor of the system, based on applications they believe users would want to use, should the users know more about how the applications worked, and that they existed. For example, it was found that end users generally do not know what an RSS-reader is; however, if they did, they might find it to be very practical. By using the recommended list, the distributor could promote such applications.

Tag cloud navigation was included for two main reasons. First of all, it gives a good indication of what applications that are available, and also which search terms that are the most popular, thus letting users find the applications currently most popular in the community. Secondly, cloud navigation is a modern and popular technology (Aouiche et al., 2008) that can make the system more interesting to use.

Another new feature is the category bar. It is a sidebar of buttons, each button a category of applications. If the user clicks a button, that button jumps to the top of the list and a box opens under it, containing all the applications of said category. When expanded, this box might become too small, in which case, the list would have to be changed. For example, another dynamic list could be contained within that box. To reset the list and close the box, the user could either click the button again, or click at the “Reset list” button at the bottom of the list. This feature was added to save space, and, like the cloud navigation, the authors feel that it makes the system more interesting and appealing.

When an application in the welcome screen is clicked, a new step is presented. Many users in the user tests seemed confused about what the applications did, and could not based on the names alone understand the purpose of the application. Was the blog application to be created for the users' personal blogs or for others? Could the application work with pictures? Could it both read and write to a blog? This new step allows the users to learn about the application before they create it. They can read what the application is meant to do, see a list of features, and also look at some screenshots of various parts of the application. These screenshots display both the functionality of the application, and the application's visual customization features. The authors believe that basic functional customization could be done on this screen if needed, such as having the users choose if the blog application should read posts, write posts, or both. However, it was decided that a new user test would be needed to determine if users would want such functionality, which is why it was not included in this design proposition.

By clicking the “Create this app!” button, the users start the actual customization process. The following step might vary depending on what application is being created, but in the case of the Blogger application example, the users are immediately introduced to the visual tool. The username and password step was removed, as it was realized it was not needed. In the new system, the username and password of the users are entered in the mobile phone instead, asking the users if they would like to save the account details the first time they enter them. It should be noted that customization of functionality could be involved in other applications, even if the blog application is restricted to visual customization. For example, the RSS-Reader application would include a step to enter the source of the RSS-feed.

The visual tool was changed in several ways. Where it previously showed every element that could be edited all at once, the new one simply shows a clean preview. By hovering the mouse over elements of this preview, the ones that can be customized are highlighted, and the users could easily see which elements that are interactive. Clicking an element brings up a box where customization options are listed. Instead of having a text telling the users how it works, the user tests indicated that most users found the visual tool simple to understand once they tried clicking it, apart from the fact that several users simply did not read the instructions. Capitalizing on this, the text was removed and replaced by a simple image. This image encourages the users to click in the preview window, and as they do, they should realize how the tool works as it is updated in real-time while the users edit the options. This new approach also allowed for more customization being added, without cluttering the initial view. For example, the users could previously only decide a color for the background. In this new tool, they could decide to upload a picture from their computer as well, and more features could be added in the future if deemed beneficial.

The last step was only changed slightly. Some users seemed confused as to what the Finish button did, and the authors realized that there was no need for the users to finish anything. The users expect to get the application to their phone as it is completed, thus the tools for adding it to the mobile phone were placed in this step. The final design proposition allows the users to get hold of the application in three ways: getting a link via SMS, manually entering a link in the telephone, or downloading it to the computer and sending it to the telephone via USB or Bluetooth.

The navigation bar was also changed. Observational notes showed that many users did not take notice of it, and some did not even understand why it was there. The bar was simplified in order to show where in the process of creation the users are, without implying any functionality; some users believed the previous navigation bar was made up of clickable buttons.

6.1.1 Architecture

In the final design proposition, the underlying architecture has not been indicated. This is due to the fact that both *Manta Ray* and *White Shark* would work; one could even use other, simpler architectures. For example, every application in the system could be created as a Java application, which uses a configuration file created by the system to define looks and simple functionality. If that is the case, the distributing company would merely have to create one Java application for each customizable application. As a result, however, the architecture would place rather high demands on third party developers if the system was to be opened up. Also, a system like *Wu* would not work with such a back-end, and new features would be more difficult to implement at a later point. Furthermore, if third party developers are ever to be allowed to extend the application library, they would have to be taught the specific standards of how to make the Java applications customizable. For these reasons, the authors believe that a more long-term solution such as *Manta Ray* should be used, or even *White Shark* if *Manta Ray* is not possible.

6.2 Academic Implications and Recommendations

EUD is a science mainly concerned with the upper part of the EUD gap, something which is shown by its focus on trying to make it easier for end users to program, and, as shown in the theory chapter, not to remove the programming entirely. EUD could thus be considered to make out a solid foundation for a bridge on one side of the EUD gap. However, a one-sided bridge will never hold. It is the authors' belief that in order to allow more people to participate in End User Development, and software development in general, a bridge that spans the entire EUD gap is needed. This would allow people to seamlessly move from one end to the other, continuously evolving and learning new things along way, perhaps eventually programming themselves. Such a seamless flow from one end to the other is most likely not, realistically speaking, possible – there will no doubt be holes and unfinished parts in the bridge, for example, in moving from a purely graphical interface to code of some sort. Hopefully, these could be made and kept small enough for a person to jump over with a little effort and willpower, in comparison with the giant leap of faith and determination that is required today in order to cross the EUD gap. While the authors are well aware that they have far from bridged the EUD gap with the solutions proposed in this report, they do believe that they have laid the first couple of bricks for the foundation on the side of the EUD gap opposite to EUD. It is the authors' hope that something like *Mandarin 2.0* could be used as an initial stepping stone of the bridge, perhaps leading to something similar to *Wu*, which in turn might lead to some approach of EUD. Some of the holes in the bridge would most likely still be too large with such a solution though, especially between *Wu* and EUD. There is yet a long way to go, and it is the authors' hope that *Mandarin 2.0* and *Wu*, or similar solutions, will be further tested and refined by future research, and that the EUD gap will be lessened even further.

6.3 Commercial Implications and Recommendations

As is supported by the user tests, there is a certain want and need among end users to customize mobile phone applications and functionality; something which the authors take as an indication that there is motivation among end users to start bridging the EUD gap. It can be speculated that many users will, as they start moving across the EUD gap, realize the possibilities of doing so, and in turn be motivated to advance onto more complex tasks; some of the users perhaps even becoming proficient programmers in the end. This would be a good thing for users and companies alike, including the IS area as a whole. The users are given more freedom of choice, the companies can successfully leave more and more development and production to end users – which cuts down on costs – and the IS area will receive more interest and attention from the general public.

However, if a company is to develop any such product commercially, there must be a market for it. This is best explored with the help of a larger demographical survey, something which has been beyond the scope and resources of this report. The usability study did, however, include a part meant to initially probe this market related issue. The authors are well aware that a usability study conducted on 11 people does not nearly compare to a large demographic survey, and is far from generalizable; however, they believe it indicates that there could be a market for this, and that it is worthy of further investigation. Assuming that this is the case, this report has identified a few general guidelines that the authors believe could be of help in the commercial development of such a product.

First of all, close cooperation with future potential users in the development process might seem like a truism today, but the authors believe it is truly vital to the success of such a product, and therefore worthy of mention. Without it, a commercial product meant to enable non-programming end users to customize functionality, or make their own applications, would most likely fail, simply because the users would not understand the interface or the way the system works.

Secondly, working with usability goals, personas, and in an interaction design oriented way, may not be as crucial as working closely with the users, but could still be very beneficial to the end product. Especially in the beginning of the development when users cannot be involved as there simply is nothing to show – although, much like Preece et al. (2009) write, users should naturally be involved as early as possible – the authors believe that, in accordance with Cooper (2004), personas are of great assistance by providing that important someone to design for, instead of just designing for the general mass.

Thirdly, working in a component based way – especially in the back-end – will severely lessen the need for maintenance and rework. This holds especially true for the mobile industry, where new models that must be supported by such a system are continuously released.

Finally, once such a product is launched, it is also crucial that is has as large of a repository of functional components as possible; thus offering a wide variety of high quality applications in order to have something of interest to virtually anyone that finds their way to the product. A close integration with a platform for online services, such as PlayNow Arena, would most likely be highly beneficial to both the platform, and the product. While at the launch, the product will naturally not cover everyone's needs, continuously added functionality and applications are of great importance. An effective way of doing this would be to open up a part of the system for third party developers, who could make their own customizable applications, and to add these to the repository. However, some form of control of what is added to the repository would most likely be required, in order to make certain that all applications would work seamlessly both with customization and in the phones. Also, the back-end would have to be constructed in a way that makes this fairly easy for third party developers to do, for example by using something similar to *Manta Ray*.

In relation to the possibility of third party developers, the matter of economical incentive should also be discussed. After all, these developers would need a reason to develop applications for the product. While the authors believe that much of the success of a product such as for example *Mandarin 2.0* lies in that it is free, the possibility of charging users for applications always exists, in which case third party developers could sell the applications they create. There is also the possibility of adding support for advertisement banners in applications or the product itself, or dividing products into those that are free and those that are not.

Something similar to *Wu* should also be considered as a possible part of a finished product – for example in form of an advanced mode or the like – in order to, as mentioned in part 6.2, enable users of *Mandarin 2.0* to move closer to writing actual code, and perhaps in the end becoming third party developers themselves.

While *Cantonese* has been left in the shadows of this report – not so much because the authors do not believe in it, as that it would not have been possible to build a satisfactory prototype of it to test – the authors would nonetheless like to mention it. In accordance with Mårtensson (appendix 2), the authors believe that a product like *Mandarin 2.0* must be enjoyable to use, lest the users will not feel inclined to take the first steps of actually trying the product, much less continue using it. It is therefore well worth considering experimenting with creative alternatives to the classical wizard that *Mandarin* is, such as for example *Cantonese*.

As for the economical incentive for the company developing the product itself, the authors believe that at least when it comes to the mobile industry – which has been the aim of this report – the availability of a product such as *Mandarin 2.0* might add sufficient surplus value to cell phone users to make the product economically viable, perhaps even more so if the product is free of charge. The product would build a user base and community around the brand of the distributor, the economical worth of which is difficult to calculate, but which is very important nonetheless (Brandweek, 2009). While the user tests indicated that the availability of a product such as this would have little to no relevance on the users' choice of telephone, if the product is marketed together with new telephones – instead of for example marketing a new telephone with the help of the product – and users try it out, the authors believe that it might cause the users to be more satisfied with their purchase, which in turn might lead them to stay loyal to the brand. These are, however, merely speculations based on personal experiences from the project, and as mentioned, further studies would be required in order to validate this. It is the authors' belief, though, that there is a future in this area.

6.4 Final Conclusions

In relation to the main research question, the authors have presented a final design proposition which they believe non-programming end users could use in order to customize applications and functionality for mobile phones. While it is impossible to know if this solution is "the best" design for the task, the authors do believe they have managed to design a system which meets the usability goals set for the project – learnability, memorability and security – and which allows programming-inexperienced users to use it with relative ease. Further user testing would nonetheless be required if the system was to be implemented.

As for the design process, the authors are of the opinion that a user-centered approach, and preferably even an interaction design approach, is highly beneficial to a development project if it is to produce a user-friendly system. While, as mentioned in the discussion, it is impossible to know if a different approach would have rendered a less user-friendly system, that is an issue beyond this thesis; the authors can merely speculate based on their personal experiences, and believe that this is the case.

With the help of the user test questionnaires, the authors believe they have found indications that the concept of customization of applications for mobile phones is something desirable by users. While it is most likely not desirable by all users, it does seem to be of interest to an amount of users large enough to warrant further investigation.

Lastly, as discussed in part 6.2, the authors are well aware that the final design proposition far from bridges the EUD gap by itself, but believe that, together with a more advanced solution such as *Wu* to follow it, it could aid in building a bridge that perhaps stretches partway across the gap. Future research and commercial solutions could add further pieces to the bridge, thus making it stretch even farther; successfully lessening the gap until it is small enough to be crossed by anyone who wishes to do so.

7 Summary

First, contact was established with Sony Ericsson, and an assignment was defined: the design of a system which allows end users to customize applications for mobile phones. By interviewing key employees at this company, the vision of such a system was realized. Following the interaction design process suggested by Preece et al. (2007), several alternatives for a system were created, including two architectural options. By demonstrating these alternatives to Sony Ericsson and receiving feedback, an alternative was chosen to be used in user tests. An improved, interactive version of the chosen alternative was created, and user tests ensued. In the user tests, two user groups were identified: advanced users and simple users. Results of the user tests were analyzed with this in mind. Several problems were found with the initial design, and a final design proposition was created with consideration to these issues.

The project conducted produced several interesting insights concerning the design of IS for end users. It was found that a system such as the final design proposition could indeed be simple enough for regular users to learn and use. The authors therefore believe that the final design proposition is a good example of how a system aimed at end users could be designed, when companies want to empower their consumers. The authors found that the use of personas and usability goals, together with a sound user interaction design process, could offer several benefits to developers in the same way these aspects helped the authors in their work of designing the system.

The report has presented how the EUD gap can begin to be bridged by the design solutions it proposed, in the context of a system which lets non-programming end users customize software for mobile phones. The authors believe a system like the final design proposition could be a first step in bridging the EUD gap; enabling end users without interest in programming and systems development to customize their applications. The authors see this as a contribution to the area of End User Development; not so much as a form of EUD, but as a way to get end users interested in the concept.

It was also found that a system such as this, or at least the concept of simple customization on a big scale, is something many users would likely find enjoyable and useful. The report therefore provides recommendations for commercial implementations, based on the authors' experience from the project.

The authors found that flexibility in customization and eventual development comes at a high cost of learnability, therefore proposing a strategy that allows end users to slowly work their way over the EUD gap, while keeping the difficulty curve as gentle and gradual as possible. The authors do not believe that one system could allow users to move from interested consumer to producing developer; several systems with increasing complexity and flexibility would, however, allow users to advance as far into customization and development as they feel is interesting and compelling.

Appendix 1 – Interview Guide

About the interviewee

What kind of work do you do at Sony Ericsson?

Questions about the problem area and the system we are to design

What is your perspective on the system to be designed?

If you could decide without any limitations how the system would work, how would it look and which functions would it contain?

Concrete questions, depending on what answers we get on the above two questions

Do you feel users should be able to make money on what they create, and how would this work in practice?

What do you think of the possibility for cooperation between users? Will users work alone or, for example, be a part of an open community which supports group development and sharing?

Which target audience do you feel the system aims at? How broad and deep should it be?

What prior knowledge will be assumed by the system, and how advanced tasks should the system support?

How open should the architecture be? What possibilities should users have regarding editing and adding to the system back-end?

How much time do you expect users to spend developing an application?

On what platform do you expect development to take place?

How do you perceive the system will handle various cellphone models? Not just all the regular Sony Ericsson cellphones, but also upcoming, new operating systems.

How will the system handle different national languages?

How is safety and application quality handled?

Appendix 2 – Interview with Viktor Mårtensson

About the interviewee

What kind of work do you do at Sony Ericsson?

Viktor is chief over a group called Tools and Documentation. He has a small group – at the moment three people – with product owners of the tools that are aimed at developers and creators outside Sony Ericsson. Being a product owner means to be in charge of defining a product, conceptualizing it, having it built, delivering it, and somehow marketing it.

Questions about the problem area and the system we are to design

What is your perspective on the system to be designed?

Viktor starts speaking about the problem area, where the English term is consumerizing development. That means making development and software development into a consumer activity. Traditionally, development of software, games or similar things for the telephone happened through regular programming where a number of programmers were allowed to create a game or something that you install. This costs relatively much money. However, Sony Ericsson has a tool called Themes Creator, which enables non-developers to make something that you can install in your telephone, or even sell. According to Viktor, it's a very popular tool, and so they thought they could make more tools like that.

Viktor continues that you could take an RSS-reader and define how you could drive consumerization development, revolving around this RSS-reader. He means the question is how to let a technically advanced user build an RSS-reader, why she would like to do that, and how to implement, design and build the software for such a system. Not to mention how to deal with the fact that there are hundreds of different cell phone models on the market today.

He contemplates as to whether or not the system should support other telephone systems. That perhaps such a prosumer tool would create different applications for different platforms; a Java application for the Java telephone, a Windows application for Windows, and so forth. So as long as you have entered a certain number of parameters, then what the actual application will look like, and what it runs on, should not matter. For example, the Sony Ericsson Z320 does not run the same things that the UIQ-telephone does. And that, he means, is a typical decision that should not be exposed to the creative hobbyist, more than what is absolutely necessary. The actual implementation is not interesting to the user.

Upon being asked if the system is only meant for Sony Ericsson telephones or for other brands as well, Viktor responds that it is primarily for Sony Ericsson, but that there are no real drawbacks if it works on other brands as well. It would however be positive if the developed product worked better, or had some cool features on Sony Ericsson's telephones. For example, Viktor says that they have proprietary APIs that can be used to integrate the applications closer to the phone than what is commonly possible on other brands. If avaliable, these APIs can be used to enhance the program, but it would still work without them

If you could decide without any limitations how the system would work, how would it look and which functions would it contain?

Seeing as there are many people who blog, and many of these are not technicians, Viktor says he wants to make it possible for them to develop some kind of app, or some kind of product, that they can market on their blog. They should somehow be able to write ”Hello world, hello everyone. By the way, I've added this new feature to my blog. Click on this link or send this sms to your phone and you can now read my blog easy in your phone”, or something similar. It is important that it should be easier to get that product into the phone than it is to get an RSS-link into the phone. The person making the product could for example either download a file that she would put on her server – which she would then link to her blog and which people could surf to on their telephone – or paste a special URL that leads to a Sony Ericsson server that keeps track of the application she has made. Viktor means that, in such case, when users click the link, the server goes “Ah, this is that telephone, so let's output the correct client for that”, or “This is a Z320-telephone, and then we take this client – configured in the way the developer made it – and output it”. He continues that anyone should be able to make it, it should be “dirt easy” to market, and it should be “dirt easy” to get it into the phone. Preferrably, Viktor would see that the applications are sent out via SMS with a link, or MMS.

It also has to be possible for a really untechnical person to make an application, while still enabling more advanced users to spend time in it, Viktor means. Thus, in the same way that the untechnical person should have an RSS-reader in five minutes, then the more advanced user should be able to spend two hours with the tool and still find those two hours meaningful; by configurating, adding small icons, and the like.

Concrete questions, depending on what answers we get on the above two questions

Do you feel users should be able to make money on what they create, and how would this work in practice?

Viktor thinks you should be able make money of what you create, and in such a case you would put it up in some kind of online store; there are existing channels for that. As for how to actually make money of it, he does not know, but points out that we should not put much effort into what an application seller would look like, but maybe how the system would work together with such a sales channel.

What do you think of the possibility for cooperation between users? Will users work alone or, for example, be a part of an open community which supports group development and sharing?

Viktor does think that it is an interesting idea, but points out that we should focus on the simple user first, and the really tech savvy people later. He does not see that development happens with other people. While it is possible that one could import graphics that someone else made when one asked for it, the creative process itself happens with the single user. The community is built by providing the tools, and that you connect these tools to some form of chat forum where people can give each other tips and pointer and put up screenshots.

Which target audience do you feel is the aim of the system? How broad and deep should it be?

The advanced consumer is the target group, says Viktor. He highlights bloggers as a pretty good example of that because the term covers quite a lot; this dues to the fact that even though there are more technicians than really simple users blogging. If you think about everyone, then anyone who blogs should be able to use the tool.

What prior knowledge will be assumed by the system, and how advanced tasks should the system support?

Seeing as they are prosumers, Victor claims they should be able to surf well. The user should not be surprised by clicking a button and choosing an image from their own file system. People should be able to find the tool after all, and that in a way is a good filter, he means – if they can find the tool then you should be qualified enough to use it.

As for how advanced things one should be able to make, Viktor refers to it as being of a secondary nature. The important thing is to start by doing something simple. Seeing as engineers love to pack applications with features, he says; the trick lies in capping the correct amount of features, or not expose the user to all the features from the start. It could perhaps start with one or two buttons in the beginning and then haves an options menu that says Advanced Mode, Super Hard Mode, etc.

How open should the architecture be? What possibilities should users have regarding editing and adding to the system backend?

One should be able to make a really open system, but Viktor does not know how or where one should open it. He says you could divide the tool into two systems. One system that takes the parameters for an application – the colors, the fonts and the images used, and generates a .jar for Java, HTML for a Facebook frontend, etc. The other system is where the user would sit and decide between the colors, choose the wallpaper image, and so forth. If that is the case, one could then imagine an open part inbetween, so that someone else could write the UI that you design your application in, he says, but also points out that he does not know if that is appropriate or even something that one would want.

It is also aired that there are, however, benefits for Sony Ericsson to work with modularized systems.

How much time do you expect users to spend developing an application?

The aim are the prosumers, says Viktor, meaning that these are people willing to spend a couple of hours on building a product. Because of this, the system aims at the span between 30 minutes and a few hours of work. However, he points out, the easiest product to make should be really quick to make. If it takes 30 minutes to make the first product then the user is never going to finish it. So, Viktor says, perhaps it should take 30 seconds; once you press a URL then you automatically get a suggestion for how it'll look, and if you then press Publish then it's done. Afterwards, if you want, you can add a new background color, new wallpaper image, new font. At its quickest it should more or less be instantaneous, because then the user will see that it really works, and get a sense of accomplishment. Viktor feels it is important that users feel this sense of accomplishment if we want them to continue using the system.

On what platform do you expect development to take place?

Viktor believes in the PC for development. He points out that while the whole industry says it is an obvious web app, he still considers the best user experience to come from heavy applications, but he does realize there is a problem in that the user would have to download it, install it, and then eventually they would forget about it. While it would be cool if the development happened in the cell phone itself, he says, that should not be the first step. Porting a web application to the cell phone is, however, not so big of a step, thus it should probably be a web based system.

How do you perceive the system will handle various cellphone models? Not just all the regular Sony Ericsson cellphones, but also upcoming, new operating systems.

Viktor states the system would have to handle them all. There will constantly be new telephones in the future and they will have to be included. Once a user makes the investment of making an application – even if it is only 15 minutes – then she should not have to remake it just to make it work on a new model.

How will the system handle different national languages?

Viktor points to EFIGS [English, French, Italian, German, Spanish] plus Chinese as being the usual target. However, he ponders whether the user must make a single-language version, or whether she could put in multilanguage support from the beginning. He points out that it should be kept simple though, so even if the tool would be available in several languages, then the generated RSS-reader should not have to be multi-language because the actual content is in just one language.

How is safety and application quality handled?

Viktor argues that safety and application quality might not be relevant if the application only takes 30 seconds to create. If the application is to be sold, it would be relevant however. Viktor brings up that a statistical guess could be applied to find if the application may or may not work on a phone, but in the end the user would somehow have to try it out themselves, possibly using an emulator.

Miscellaneous

Upon being asked if we could put the user test on the Internet, Viktor answered that he did not want that, and that Sony Ericsson would provide test subjects for a user test.

Appendix 3 – Interview with third key employee

This interview has not been approved by the third key employee, who wishes to remain anonymous, and therefore it cannot be included in the report.

Appendix 4 – Interview with Jonas Petterson

About the interviewee

What kind of work do you do at Sony Ericsson?

Jonas works under Viktor and is product chief for the tool Themes Creator. He also works in a few other areas such as Capuchin, a Sony Ericsson technology created to allow a mixture of Flash/Flash Lite and Java, thus allowing developers to create an application where the interface is made in flash and the ”brains” are made in Java.

Questions about the problem area and the system we are to design

What is your perspective on the system to be designed?

Jonas' picture of what we are doing, is that we are trying to come up with one or several suggestions as to what Sony Ericsson can do in the area of prosumers; trying to find some form of service or tool that will allow a advanced consumers to build their own applications, or create their own solutions, which can be loaded to the mobile phone. He does not yet really have an image of the system himself.

If you could decide without any limitations how the system would work, how would it look and which functions would it contain?

Jonas brings up an example they are working with at the moment; how users working in Themes Creator will be able to buy Capuchin components such as, for example, a clock component and use those in their themes. He's pondering over where to draw the line between technology and possibility. The developer should not be forced to understand how the component works; he or she should just buy it and easily integrate it into the theme being created. He sees us creating some form of tool which can allow the user to create something like a widget with drag-n-drop possibilities, where one can take an RSS-reading and drop into a picture, then maybe some form of banner. Status messages from Facebook or Twitter are also good examples of expected functionality. He states that in such a situation, one can both make it technically advanced but hard to use, or very simple where the system hides the complexity from the user. Jonas tells us how Themes Creator has solved it by means of levels of complexity. The user can for example use a simple three-step wizard, or look directly at the XML-code.

Jonas discusses that a dream scenario would be a system where the fact that there are different mobile phones built on various platforms could be hidden from the user. A user of this system wouldn't have to think about how there is something called Windows Mobile, something called Android and so on; it should be completely crossplatform. Of course, it should be very visually attractive.

Concrete questions, depending on what answers we get on the above two questions

Do you feel users should be able to make money on what they create, and how would this work in practice?

Jonas is of the belief that one can make money on basically anything. He tells us that before he began working for Sony Ericsson, he worked for Disney mobile where they sold simple Disney wallpaper applications which automatically download Disney wallpapers for the mobile. Really simple things, but users paid for it. He argues that users generally buy services, not physical products. If the price is at the proper level, users will pay for things they could in theory do themselves. However, he does feel users will in all probability mainly develop for their own personal needs.

Jonas would prefer if Sony Ericssons current channels like Playnow Arena were used as the actual marketplace for these kinds of applications.

What do you think of the possibility for cooperation between users? Will users work alone or, for example, be a part of an open community which supports group development and sharing?

Jonas thinks that cooperation should be part of the system in some form of community. That is, if you're building something and want to present how it was done and such, you should be able to. He hasn't yet decided whether that should be a new community though, or if it should simply be a part of PlayNow Arena or Sony Ericssons development site, or some other system already in operation.

Which target audience do you feel the system aims at? How broad and deep should it be?

Jonas aims for a consumer who uses various social networks such as Facebook or MySpace; someone who is decently used to utilizing various services. People who have just learned how to text messages are not the target. Maybe someone who knows basic HTML, but Jonas doesn't feel one has to go that far. If he is to define a group of people to target, he'd say it is more or less every person from the Western world younger than 40 years old, so it's a big group.

What prior knowledge will be assumed by the system, and how advanced tasks should the system support?

Jonas would prefer a system which has no prerequisite knowledge of such things as programming. Some form of drag-n-drop solution would be optimal. A way to create mobile applications in a similar way to some tools where users can create their own websites by drag-n-drop.

Jonas hasn't really spent time thinking about how advanced the tasks should, and would like to see our suggestions about what could be possible.

How open should the architecture be? What possibilities should users have concering editing and adding to the system back-end?

Jonas believes the system should start out completely closed with the argument that Sony Ericsson often makes tools a bit too complicated, something which could scare new users away. He believes a slimmed down, closed environment would be good in the beginning. Later, some APIs could be created to open up the system. That is, however, is a future issue.

How much time do you expect users to spend developing an application?

Anything from a mintue to a few days. Jonas feels a user should be able to get a good impression of the system is less than a minute.

On what platform do you expect development to take place?

According to Jonas, the trend in the industry today is to create web applications, but he feels it should only be done if it's possible to do it in a really smooth way, AJAX being a possibility. If the system can't be created in an optimal way as a web application, he would rather see it developed for PC as installable software.

How do you perceive the system will handle various cellphone models? Not just all the regular Sony Ericsson cellphones but, also upcoming, new operating systems.

Jonas claims the only real demand is that new mobile phone models should be possible to easily add to the system. He mentions a system Sony Ericsson is currently using, a database with technical specifications for all their mobile phones. If the system could integrate with that database and automatically customize applications to fit phones in that system, it would be a good solution.

How will the system handle different national languages?

Jonas tells us that it's a matter of who the target user is. If it is a prosumer, the tool should be available in all languages where Sony Ericsson mobile phones are being sold.

How is safety and application quality handled?

Jonas believes it would be very hard to automate quality checking. It would have to be some form of filter, real humans controlling what users try to sell or share, especially when money is involved. He does not believe the actual system should be limited though, only the act of sharing it with others.

Appendix 5 – Persona Kate

Name: Kate Anderson

Age: 17

Gender: Female

Location: San Francisco, California. Lives with her parents and younger brother Ryan (12) in an American dream style villa.

Social status: Boyfriend – Keith (18), since 4 months. They met at the mall.

Education: Mainly studies social sciences, and is interested in languages. Studies Spanish. Goes to school but is not top of her class. Does fairly well and is happy with it. Likes going to school to meet her friends.

Interests: Boyfriend, friends, street dance, using cellphone, browsing.

Cellphone: Sony Ericsson W595.

Telephone use: Lots of texts/MMS to friends. Calls a lot as well. Customizes cellphone a lot by buying and/or downloading ring tones, themes and MP3s. Uses it as an MP3 player. She Twitters from her phone daily.

Food: Eats anything her mom serves, but likes sushi in particular.

Ringtone: Beyonce – Halo.

Background: Picture of boyfriend.

Technical competence: Can Google, can upload images to Facebook, is good at surfing, but no programming experience. Uses Facebook, MySpace and such regularly.

Computer: Laptop that she got for her 17th birthday with Windows Vista.

Browser: Uses Internet Explorer 7.0

Spare time: Goes to cafés with friends. Spends a couple of hours a day by the computer, socializing over Aim or Facebook. Youtubes a lot. Hangs out with her boyfriend. Shopping with friends. Occasionally watches a movie with friends/boyfriend. Practices street dance at the local dance studio twice a week.

Drivers license: Yes, but boyfriend prefers to drive. She has no car of her own.

Connection to Sony Ericsson: Likes her phone but didn't pick it specifically because of the brand. Doesn't know much about the actual company. Not really interested in technical brands. Has visited Sony Ericsson website once to download PC-suit in order to transfer songs to the phone.

Goals: Kate wants to make her phone feel personal and special. She would like to enjoy herself while customizing her applications, and impress her friends with how good and special her phone looks and works.

User stories:

- Kate can take an application she likes and change it to fit her style (ability to transfer applications from the phone to the system?).
- Kate can save a look-and-feel she likes and apply it to different applications.
- Kate can easily share her applications with her friends.

Use of system:
Kate will probably not be very interested in creating her own applications in terms of functionality. Instead, she wants applications she already uses to fit her phone. She will probably want to get a Twitter application for example, and edit it to look personal. She would probably not be interested in personally making a Twitter application, unless the process is extremely simple and almost automated.

Appendix 6 – Persona Logan

Name: Logan Tarrant

Age: 33

Gender: Male

Location: Melbourne, Australia, in the suburbs near the ocean; 1-floor house.

Social Status: Married to Michelle Tarrant (32). Owns a dog, a golden retriever.

Work: Has his own small surfing store where he sells surfboards and equipment, teaches surfing, and holds tours of good surfing spots. Wife helps out but works at a beach café. He also has an online store for his company where he blogs about surfing. Supplies weather information for various surfing spots. He has a few empoyees at the store.

Interests: Surfing and anything that pertains to surfing. For example chilling out on the beach with his wife and buddies, drinking a beer by the fire. Traveling. Swimming.

Cell: Sony Ericsson C510.

Telephone use: Taking pictures, mainly for his blog (he was of course not satisfied with the Blogger-locked automatic pic-to-blog functionality in his cell phone). Uses it as a business phone.

Food: Organic food and Thai. Obviously a big fan of BBQ on the beach.

Ringtone: The Ventures – Pipeline.

Background: A picture of the beach during summer.

Technical competence: Finds his way around his phone rather easily. Has some basic knowledge of HTML. Mainly uses his computer for business and blogging, to a certain degree social networking.

Computer: Windows Vista on a desktop PC.

Browser: Was convinced by his friends to use Firefox 2.0, hasn't bothered updating since.

Spare time: Surfing, anything surf related. Likes to travel with his wife.

Connection to Sony Ericsson: Doesn't really know anything about the company.

Goals for using system: Mainly, to spread surfing to more people, because it is such a wonderful activity. The best activity in the world even! Secondly, to increase number of visitors to his site and blog, and thirdly, to increase the turnover and profit of his surfing shop.

User stories:

- Since Logan want to be able to be out surfing as much as possible, he wants to spend as little time as possible constructing applications, that also means it has to be easy to learn and use.
- Logan is not entirely sure what an RSS-feed is, but he has heard it is good to have on one's blog. He wouldn't mind learning more about it, and he'd like to be able to have one in an application without too much fuzz.
- Logan would really like to be able to upload images to his blog straight from his cell phone in an easy and timely fashion.
- Logan wants to get the application into his cell phone quickly and easily, and to other people's cell phones as well.
- Logan would like an application to make it possible for people to read his blog, and access his web store from their cell phone, maybe even put orders or sign a form of interest for lessons and tours.
- Logan would like to have weather info in his cell phone from different surfing spots, and perhaps even suggestions for the best place right now, given certain circumstances.
- Logan cares how his application looks, naturally, but the most important part is that it works. He doesn't really want to spend much time making it look good.

Use of system (several of these could be combined into one):

- Application for personal use for blogging from his cell phone, or at least directly uploading images.
- Application to make it easier for other people to read his blog, for example via RSS.
- Application for getting weather information from various surfing places, or even suggestion for the best one, from his site into the cell phone; for public use.
- Application for making his web store available via the cell phone.
- Application for signing up for surfing tours of lessons.

Appendix 7 – Persona Brian

Name: Brian Gardner

Age: 27

Gender: Male

Location: London, UK, in a fairly nice 3-roomer in Greenwich.

Social status: Single.

Work: Runs his own small software development company. Works a lot. Has a bachelor of science in software engineering from the University of London, which he finished 4 years ago. He then worked at a web development company as a consultat making webpages for various customers. Quit 2 years ago and started his own company, BDesign. Works from home. Has created small web applications and adapted web pages for use in mobiles and on Facebook. Nothing popular but did generate some income. Main source of income as a web developer for various small to medium sized companies. Has some cool ideas for iPhone or Android apps but hasn't had the time to look into the possibilities any further.

Interests: Music (punk/rock, especially likes NoFX) with political lyrics, owns a guitar but lacks the ability to play it, web development, waiting impatiently for the release of CSS 3.0. Likes to create extensions for Firefox, mostly for private use. Really wants to make it big with his own company, thus the workoholic attitude.

Cellphone: Sony Ericsson G900.

Use: Calender, checking emails, simple web surfing, business calls, checks his adapted websites. Especially likes the pen supported note-taking.

Food: Italian (homemade pasta), he likes to cook.

Ringtone: NoFX – Sticking in my eye.

Background: His company's logo.

Technical competence: Good skills in C++ and Java. Experienced web developer (XHTML, CSS, JavaScript, PHP, MySQL). Quick learner when it comes to computers. Familiar with Facebook API.

Computer: Stationary PC and laptop with Windows XP.

Browser: Firefox 3.0 for personal use, checks his websites in most big browsers.

Spare time: Mainly works. When he allows himself to take an evening off, he goes to the pub with his old working buddies, or his friends from college.

Connection to company: Considers Sony Ericsson a safe and stable choice for standard cellphones.

Goals for using system: To have fun while working, to make it big with his company, and in order to do that, he must make more money. He wants his company to evolve and become something he can be proud of.

User stories:

- Brian can add advertisement to his applications from various data sources, just like a website.
- Brian can put the application up for a set price on a website where people can pay to get access to it.
- Brian can quickly create small applications for new markets and see how many people download and use his application. (Some form of statistics are collected)
- Brian can stop people from editing his applications.
- Brian wants to be able to make something simple quickly, but also wants to be able to spend some more time in order to make it more advanced.
- Brian doesn't want to be hindered by lack of functionality or a stupid GUI when he creates apps.
- Brian wants to have fun while making an app; or at the very least, it shouldn't be frustrating.
- Brian needs to be able to choose for himself how his application should look, and what it should do. Maybe even how it should do it.

Use of system: He will most likely make apps for other companies and be paid that way. However, he may very well also create apps that in themselves have enough value to be sold [can't really come up with an example], so that he can make money off them directly. Heck, maybe even a game or two [most likely beyond scope of system].

Appendix 8 – Mail Correspondence with Albin Olofsson

The following is a translated brief of the e-mail that was received as feedback on the personas from Sony Ericsson. It was sent by Albin Olofsson, a recent employee who works with prosumer tools under Viktor Mårtensson.

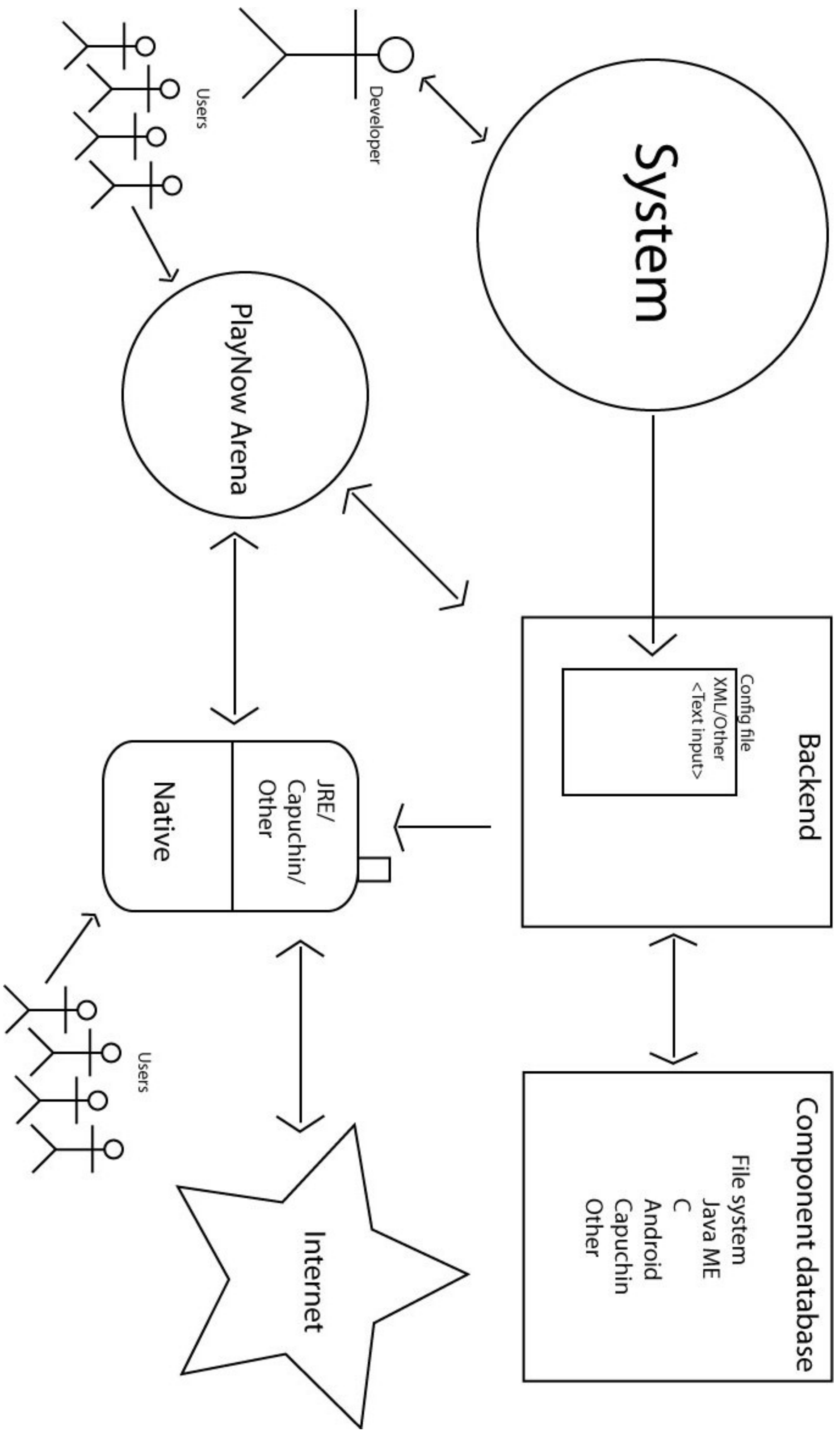
Albin has identified two segments of prosumers:

1. Those who wish to customize their telephones, in which case it has to be quick and easy to do so. This is best represented by Kate.
2. Those who want to create applications that use different content. These could be entrepreneurs who wish to spread/make available their business, or creative individuals who have ideas about what you could make with a mobile phone and its various components. This group is best represented by Logan.

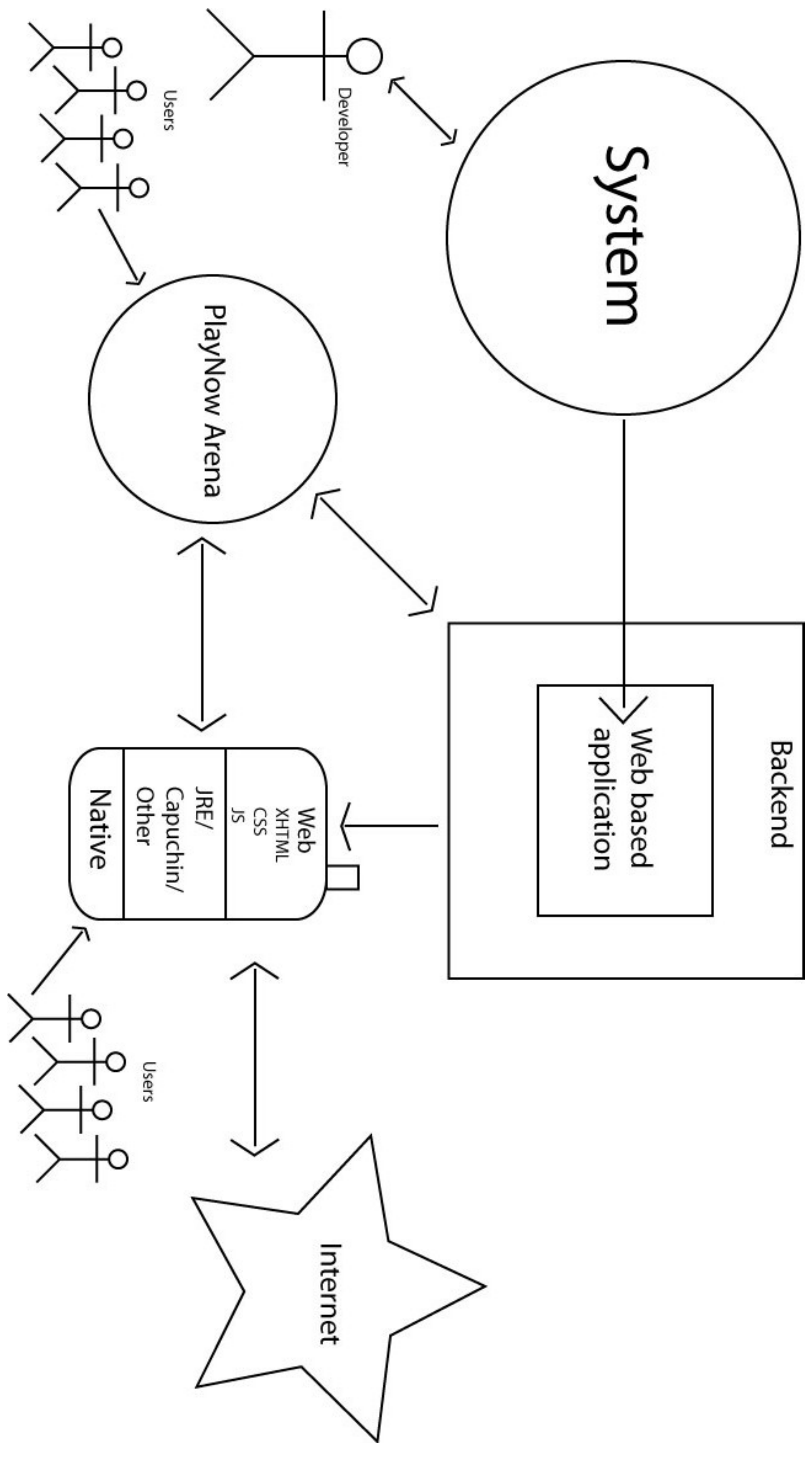
Albin means that in the short term, it is probably easier to create tools that Kate can use, but that it is also valuable, in the long run, to give Logan the possibility of creating applications that suit his business and interest. For this reason, he thinks that we should focus on Logan.

Finally, Albin writes that he would like to see a tool which can build applications that use modules such as camera, GPS or RSS. He does however feel that it is important that it is not too difficult to create the application, and Logan should be able to create his first application fairly quickly.

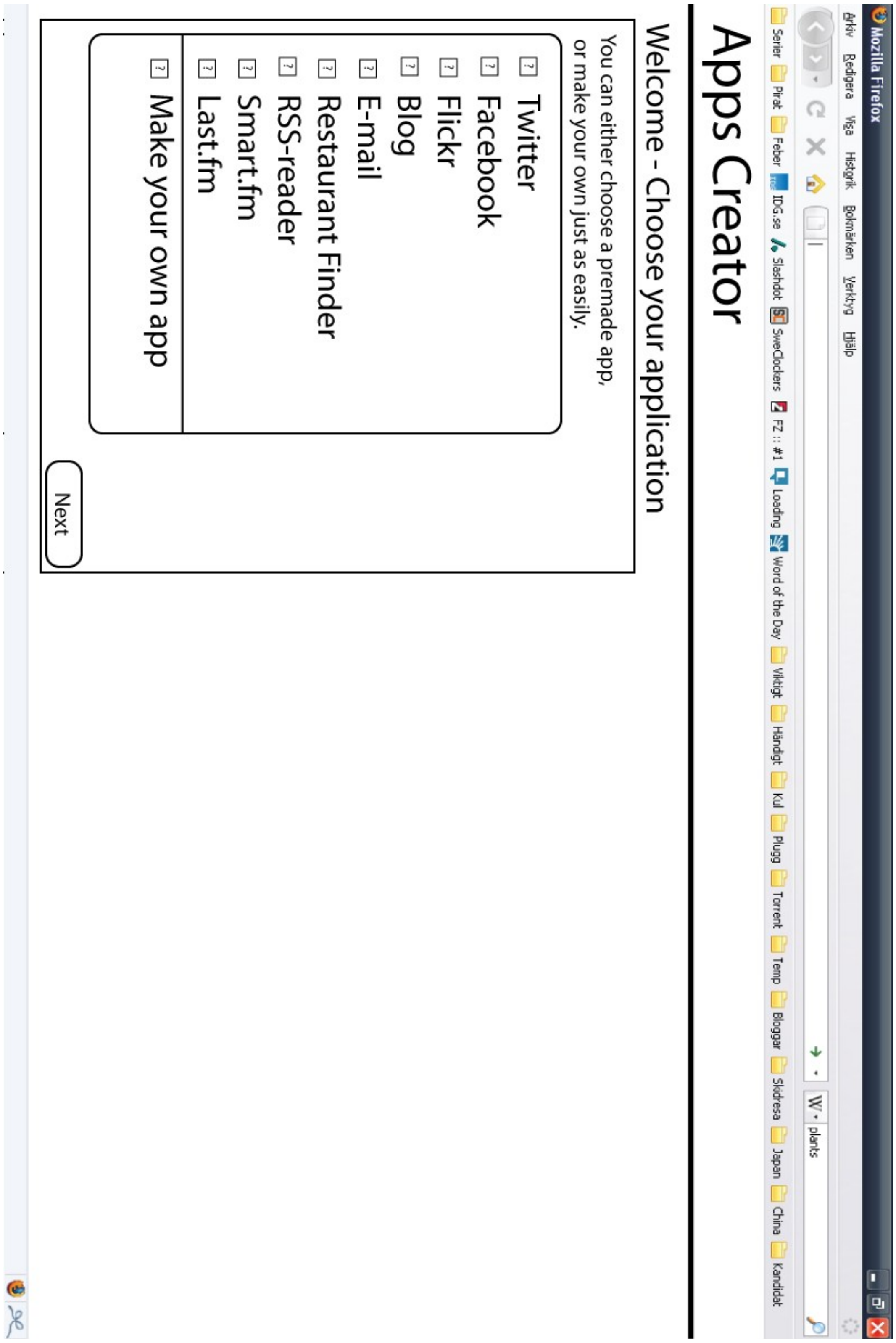
Appendix 9 – Design Proposition White Shark



Appendix 10 – Design Proposition Manta Ray



Appendix 11 – Design Proposition Mandarin Screen 1



Apps Creator

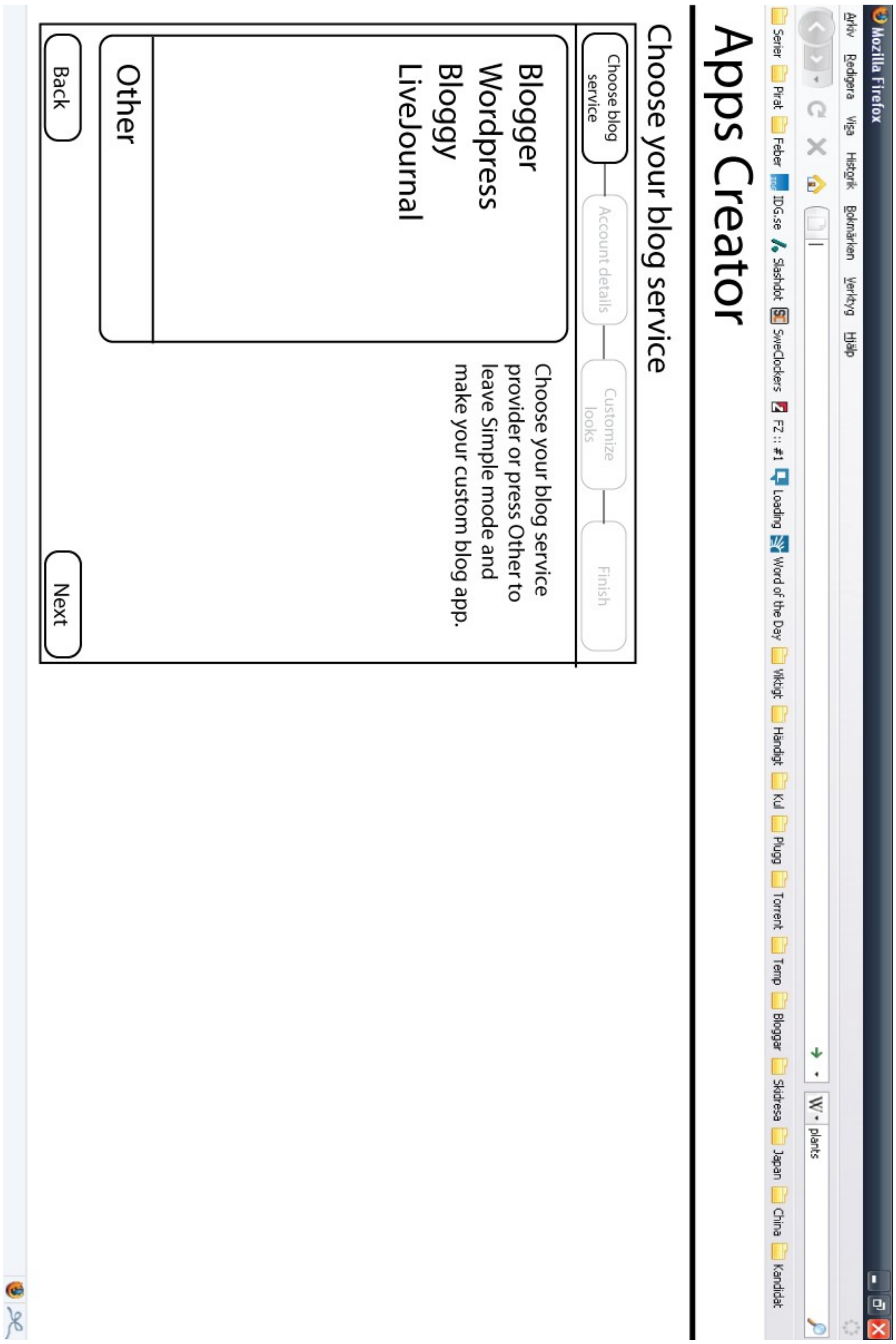
Welcome - Choose your application

You can either choose a premade app, or make your own just as easily.

- Twitter
- Facebook
- Flickr
- Blog
- E-mail
- Restaurant Finder
- RSS-reader
- Smart.fm
- Last.fm
- Make your own app

Mozilla Firefox
Arkiv Redigera Visa Historik Bokmärken Verktyg Hjälp
Serier Pirat Feber IDG.se Slashdot SweClockers FZ :: #1 Loading Word of the Day Viktigt Händigt Kul Plugg Torrent Temp Bloggar Skidresa Japan China Kandidat
plants
Next

Appendix 11 – Design Proposition Mandarin Screen 2


Mozilla Firefox
Arkiv Redigera Visa Historik Bokmärken Verktyg Hjälp
Serier Pirat Feber IDG.se Slashdot SweClockers FZ :: #1 Loading Word of the Day Viktigt Händigt Kul Plugg Torrent Temp Bloggar Skidresa Japan China Kandidat
plants
Choose blog service
Account details
Customize looks
Finish
Back
Next

Apps Creator

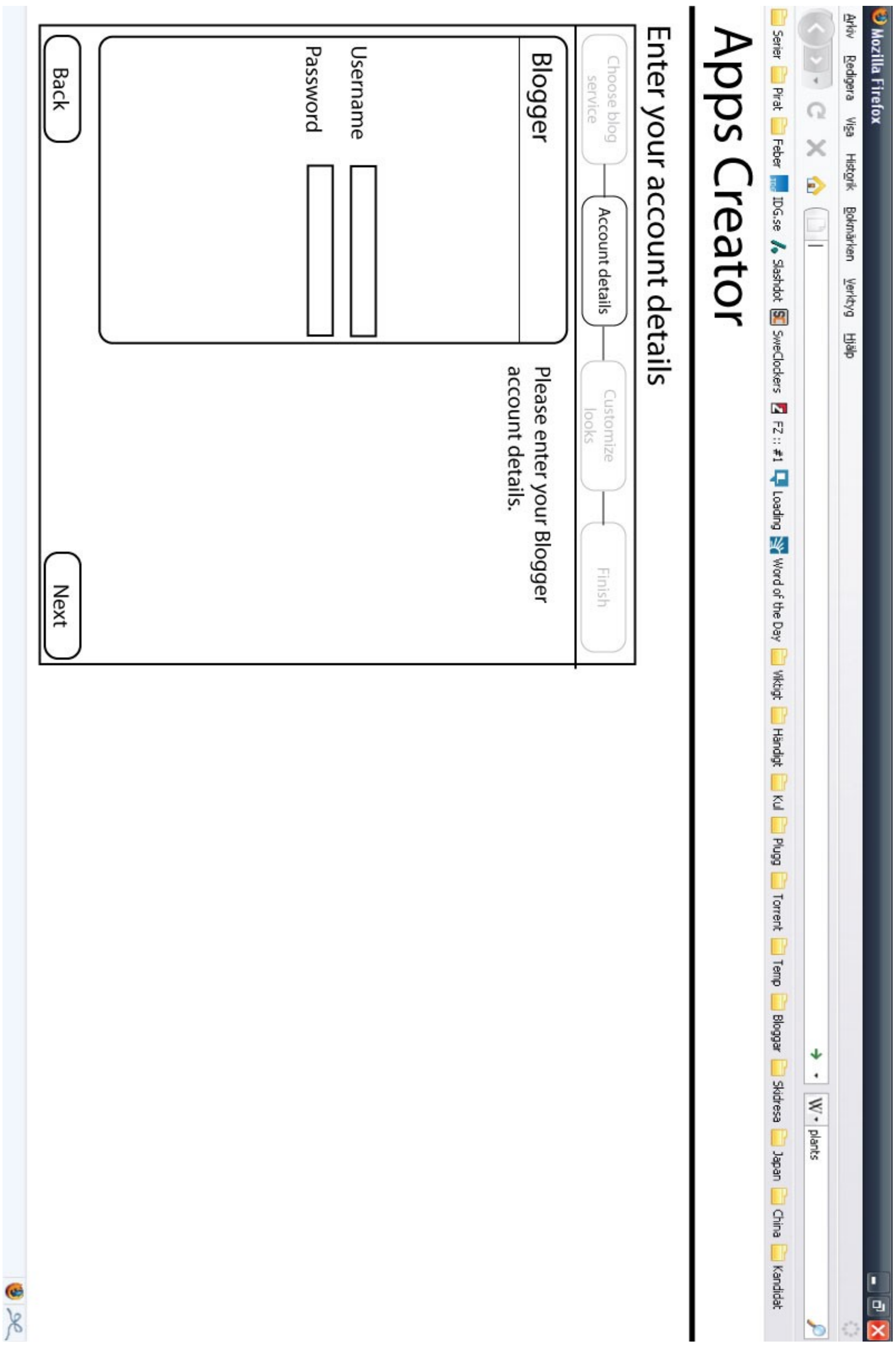
Choose your blog service

Blogger
Wordpress
Bloggy
LiveJournal

Other

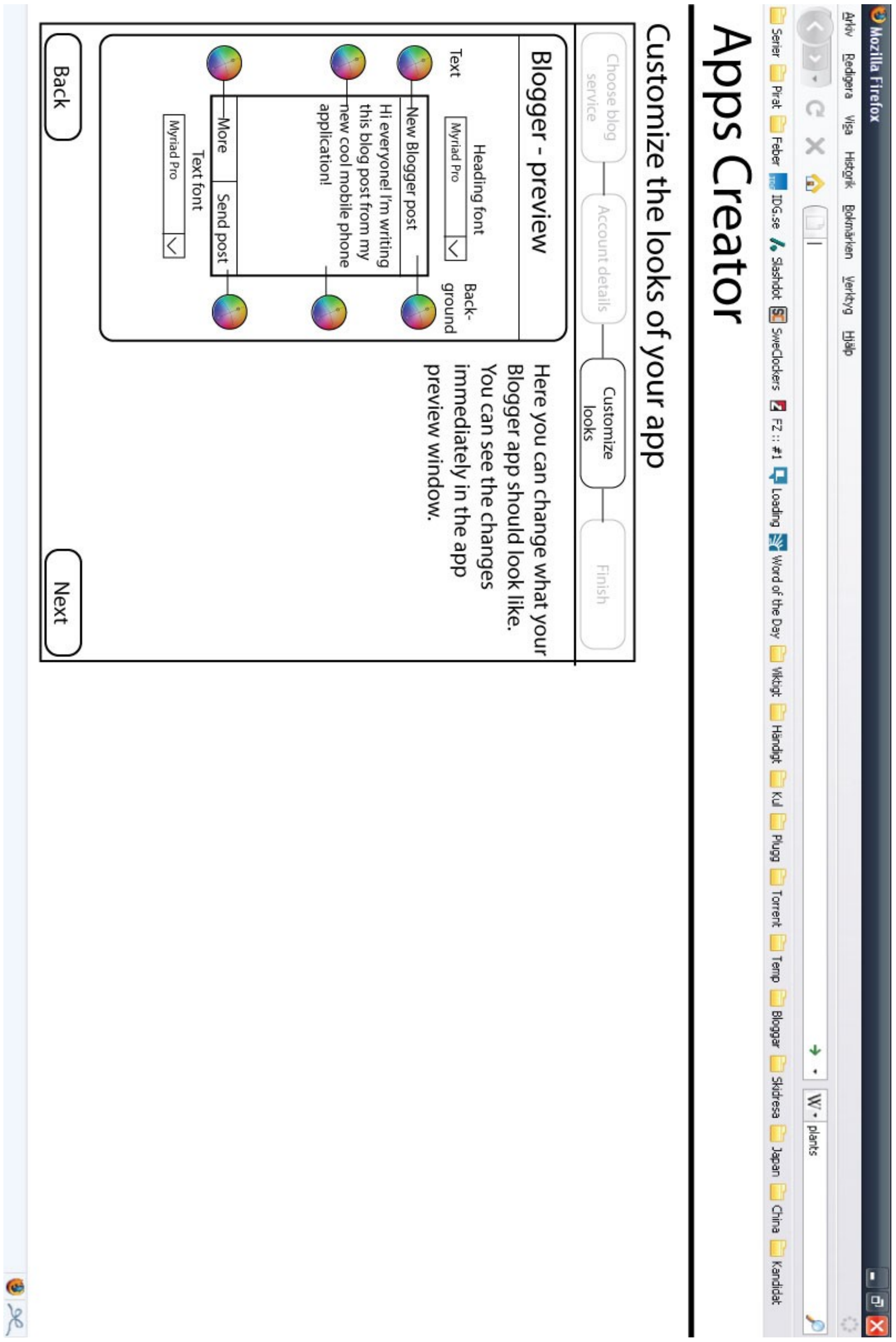
Choose your blog service provider or press Other to leave Simple mode and make your custom blog app.

Appendix 11 – Design Proposition Mandarin Screen 3



Please enter your Blogger account details.

Appendix 11 – Design Proposition Mandarin Screen 4


Mozilla Firefox
Arkiv Redigera Visa Historik Bokmärken Verktyg Hjälp
plants
Serier Pirat Feber IDG.se Slashdot SweClockers FZ :: #1 Loading Word of the Day Viktigt Händigt Kul Plugg Torrent Temp Bloggar Skidresa Japan China Kandidat
Choose blog service
Account details
Customize looks
Finish
Blogger - preview
Heading font
Myriad Pro
Text
Back-ground
New Blogger post
More
Send post
Text font
Myriad Pro
Back
Next

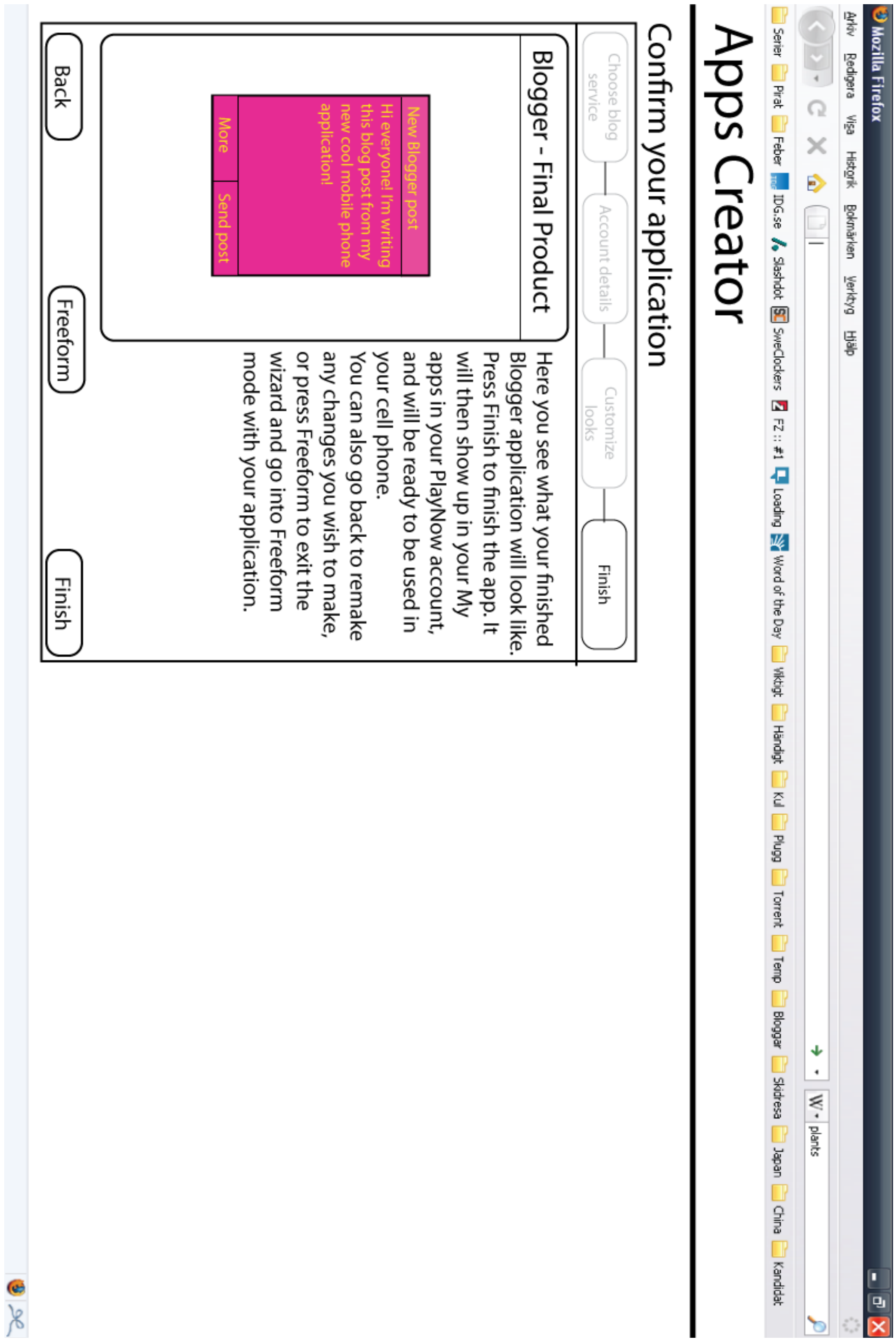
Apps Creator

Customize the looks of your app

Here you can change what your Blogger app should look like. You can see the changes immediately in the app preview window.

Hi everyone! I'm writing this blog post from my new cool mobile phone application!

Appendix 11 – Design Proposition Mandarin Screen 5


Mozilla Firefox
Arkiv Redigera Visa Historik Bokmärken Verktyg Hjälp
plants
Serier Pirat Feber IDG.se Slashdot SweClockers FZ :: #1 Loading Word of the Day Viktigt Händigt Kul Plugg Torrent Temp Bloggar Skidresa Japan China Kandidat
Choose blog service
Account details
Customize looks
Finish
More
Send post
Back
Freeform
Finish

Apps Creator

Confirm your application

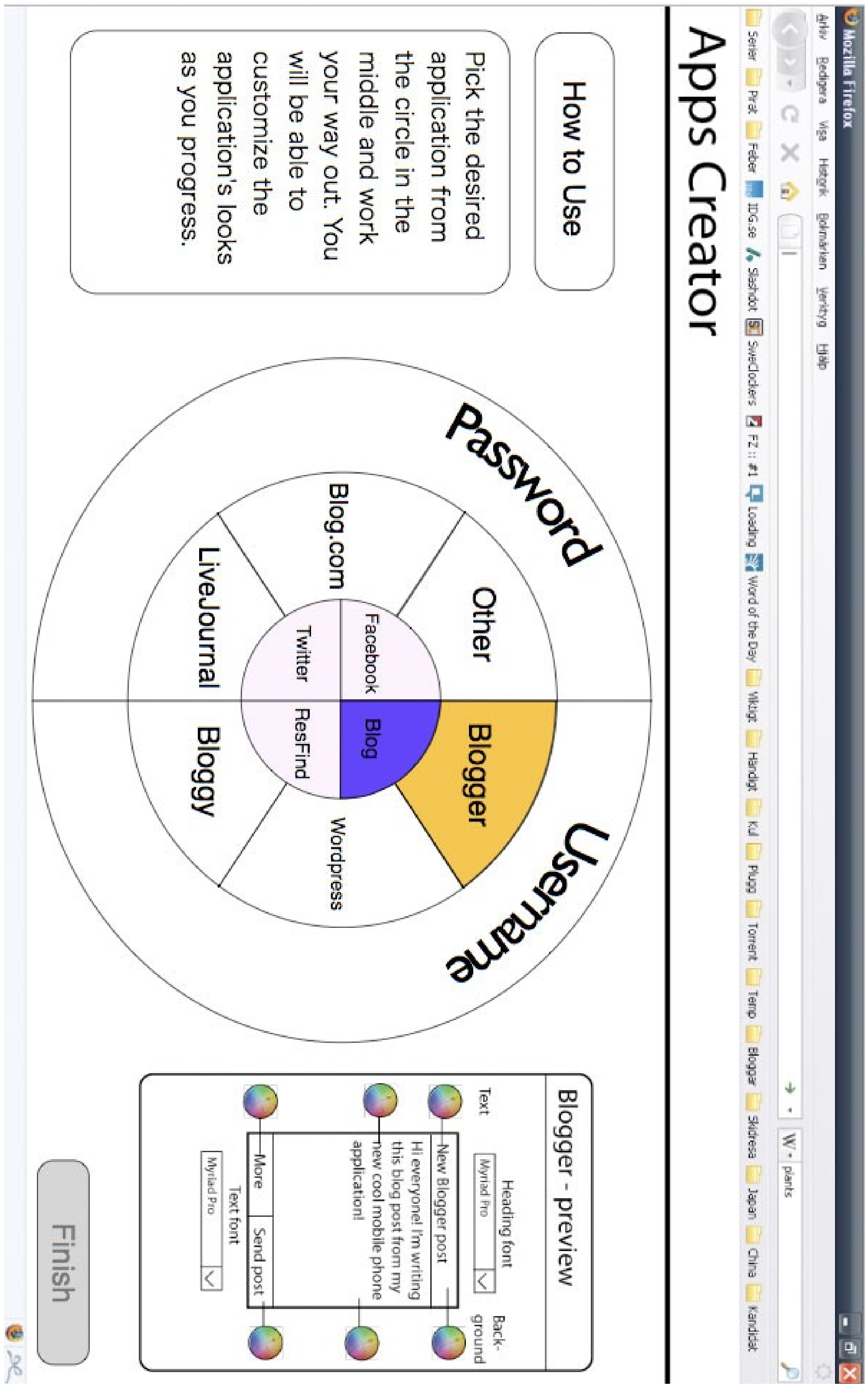
Blogger - Final Product

New Blogger post

Hi everyone! I'm writing this blog post from my new cool mobile phone application!

Here you see what your finished Blogger application will look like. Press Finish to finish the app. It will then show up in your My apps in your PlayNow account, and will be ready to be used in your cell phone.
You can also go back to remake any changes you wish to make, or press Freeform to exit the wizard and go into Freeform mode with your application.

Appendix 12 – Design Proposition Cantonese

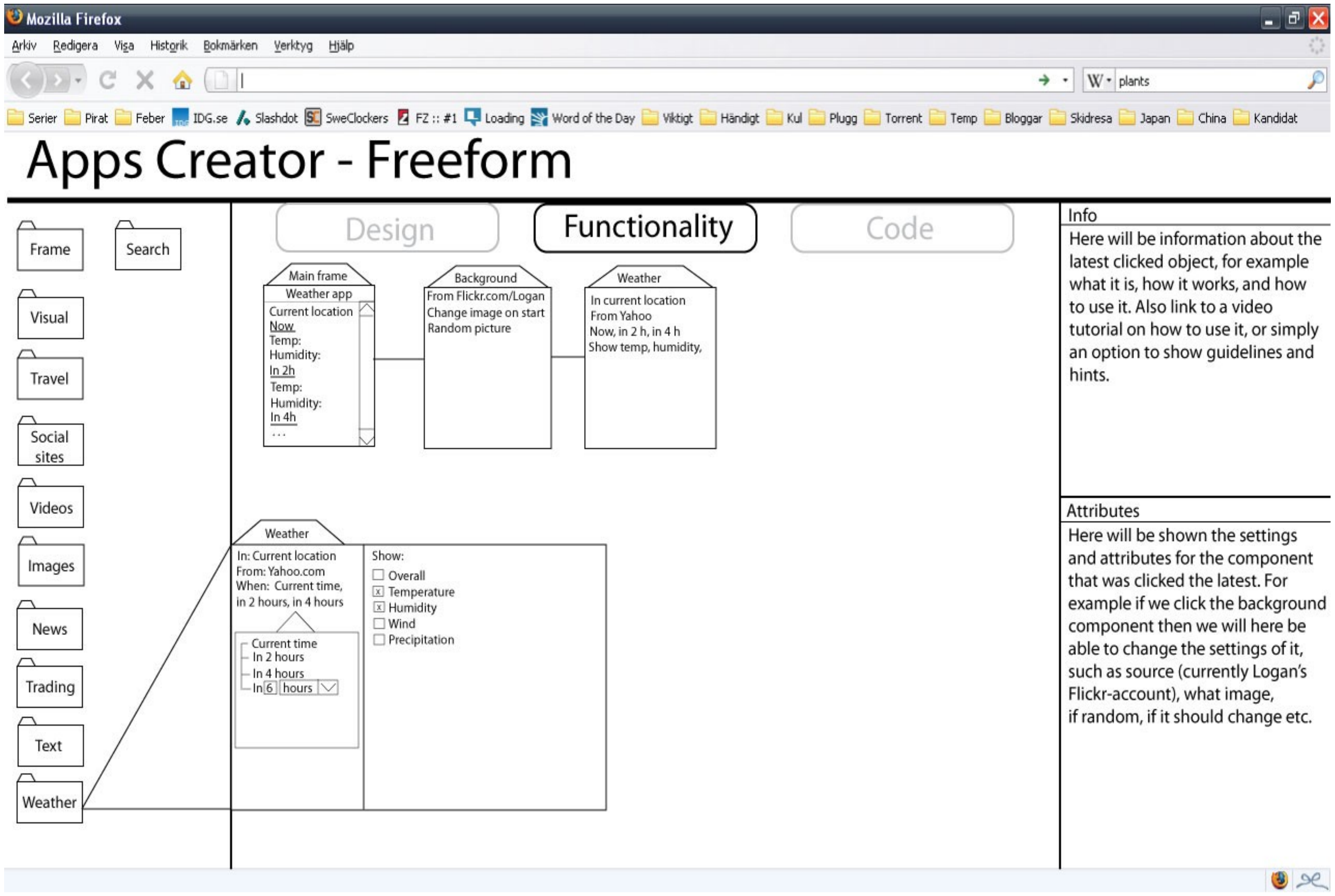


Pick the desired application from the circle in the middle and work your way out. You will be able to customize the application's looks as you progress.

New Blogger post

Hi everyone! I'm writing this blog post from my new cool mobile phone application!

Appendix 13 – Design Proposition Min



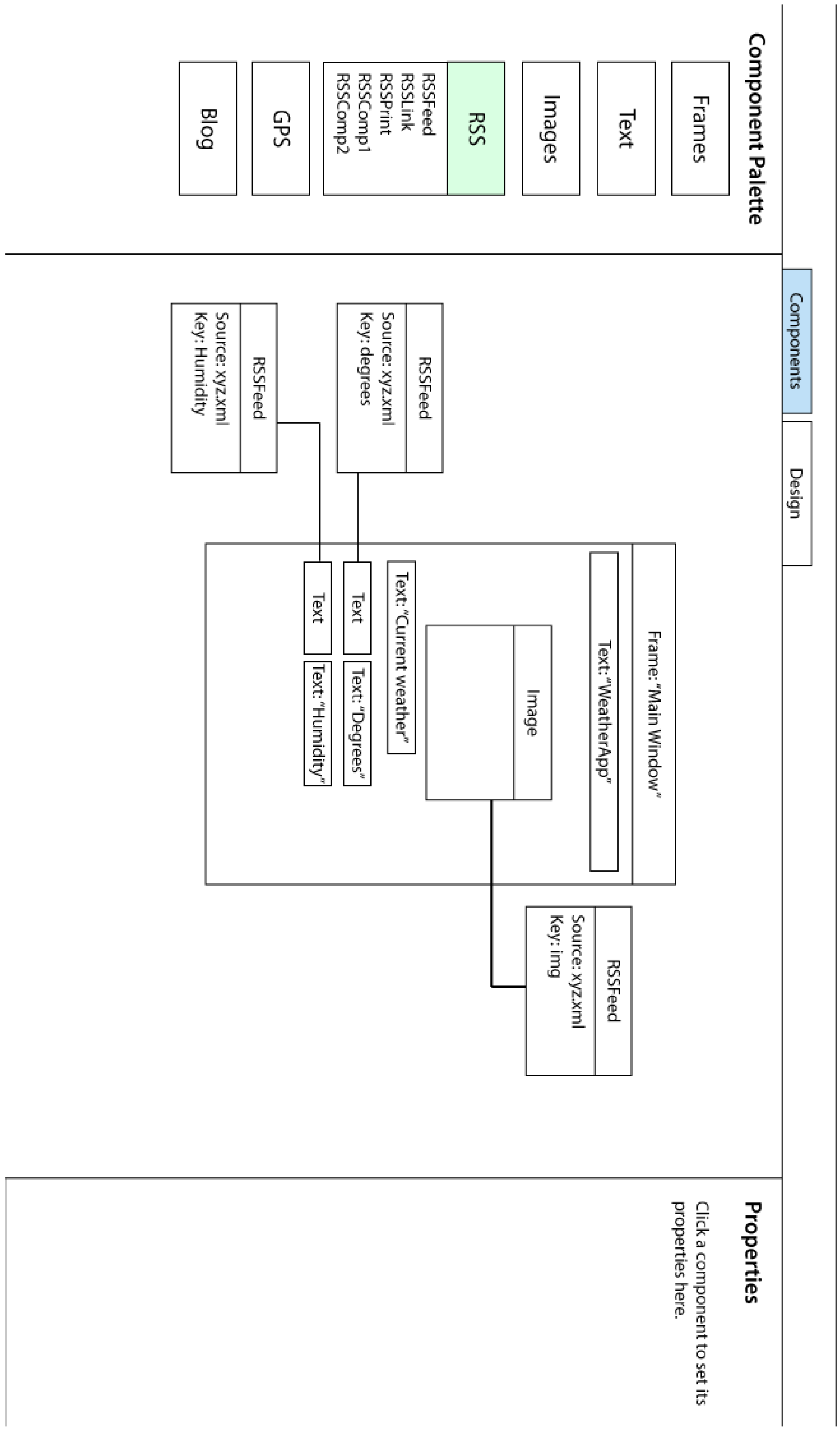
Info

Here will be information about the latest clicked object, for example what it is, how it works, and how to use it. Also link to a video tutorial on how to use it, or simply an option to show guidelines and hints.

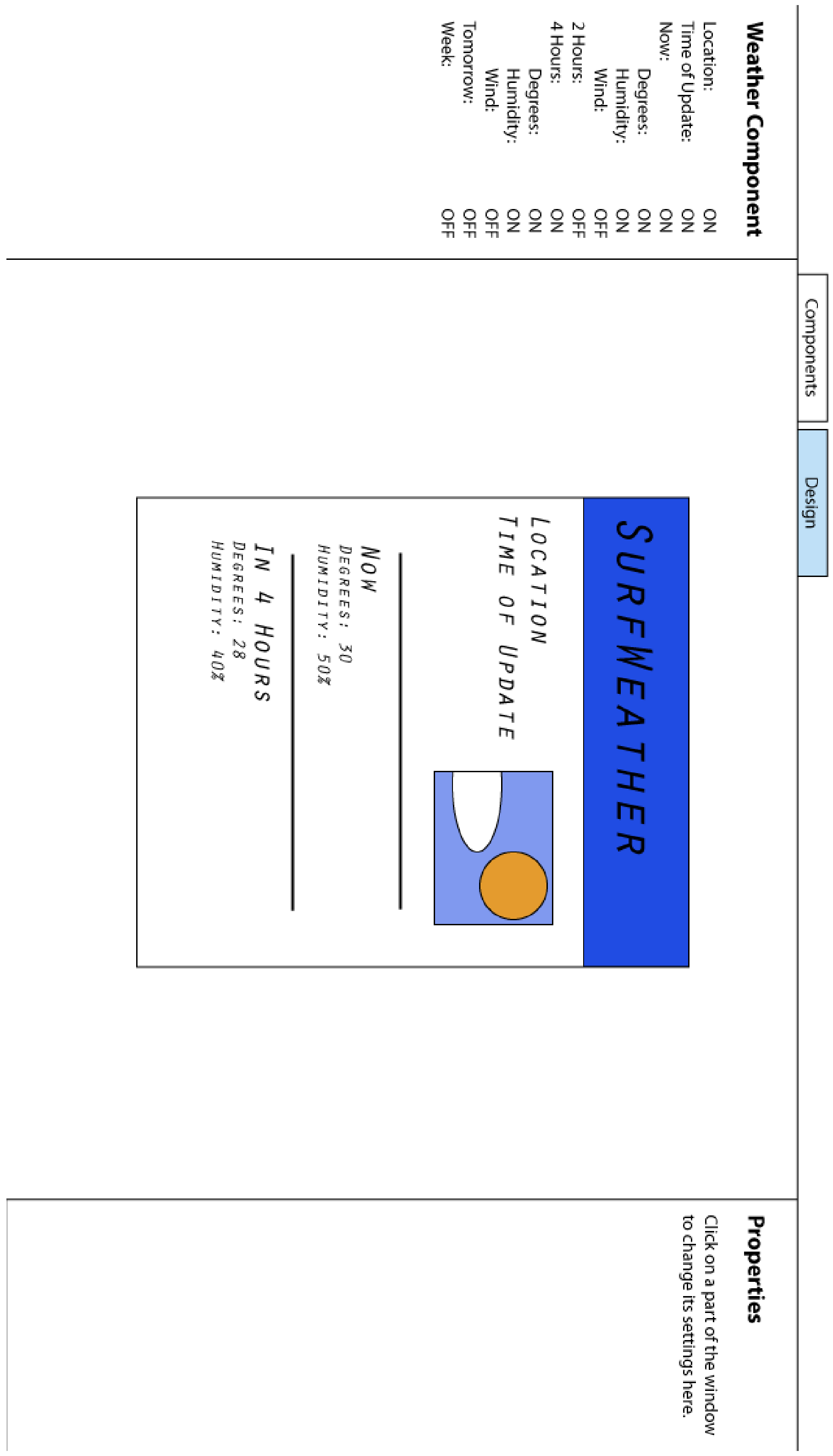
Attributes

Here will be shown the settings and attributes for the component that was clicked the latest. For example if we click the background component then we will here be able to change the settings of it, such as source (currently Logan's Flickr-account), what image, if random, if it should change etc.

Appendix 14 – Design Proposition Xiang



Appendix 15 – Design Proposition Min & Xiang Visual



Weather Component

Location:	ON
Time of Update:	ON
Now:	ON
Degrees:	ON
Humidity:	ON
Wind:	OFF
2 Hours:	OFF
4 Hours:	ON
Degrees:	ON
Humidity:	ON
Wind:	OFF
Tomorrow:	OFF
Week:	OFF

SURFWEATHER

LOCATION
TIME OF UPDATE

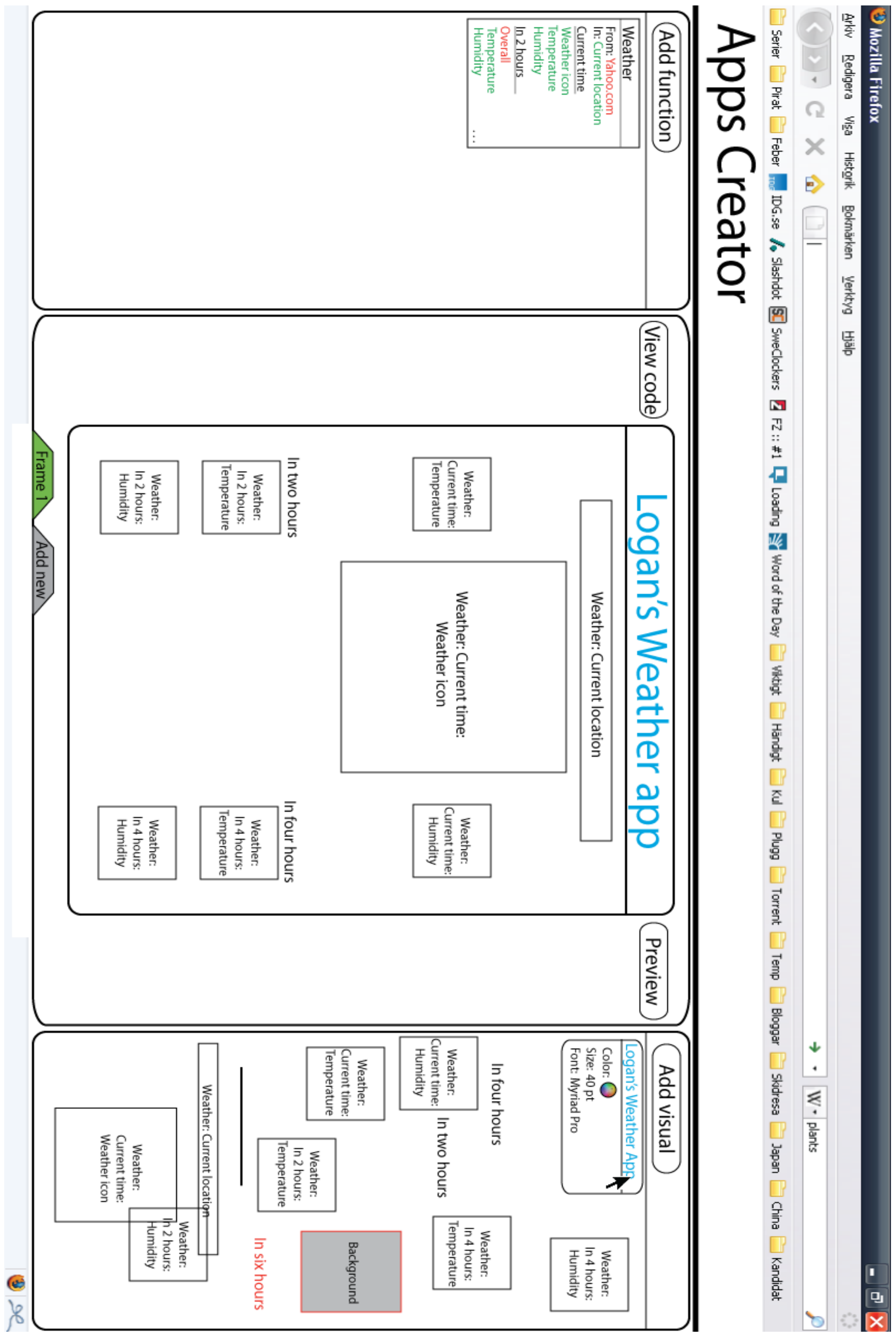
NOW
DEGREES: 30
HUMIDITY: 50%

IN 4 HOURS
DEGREES: 28
HUMIDITY: 40%

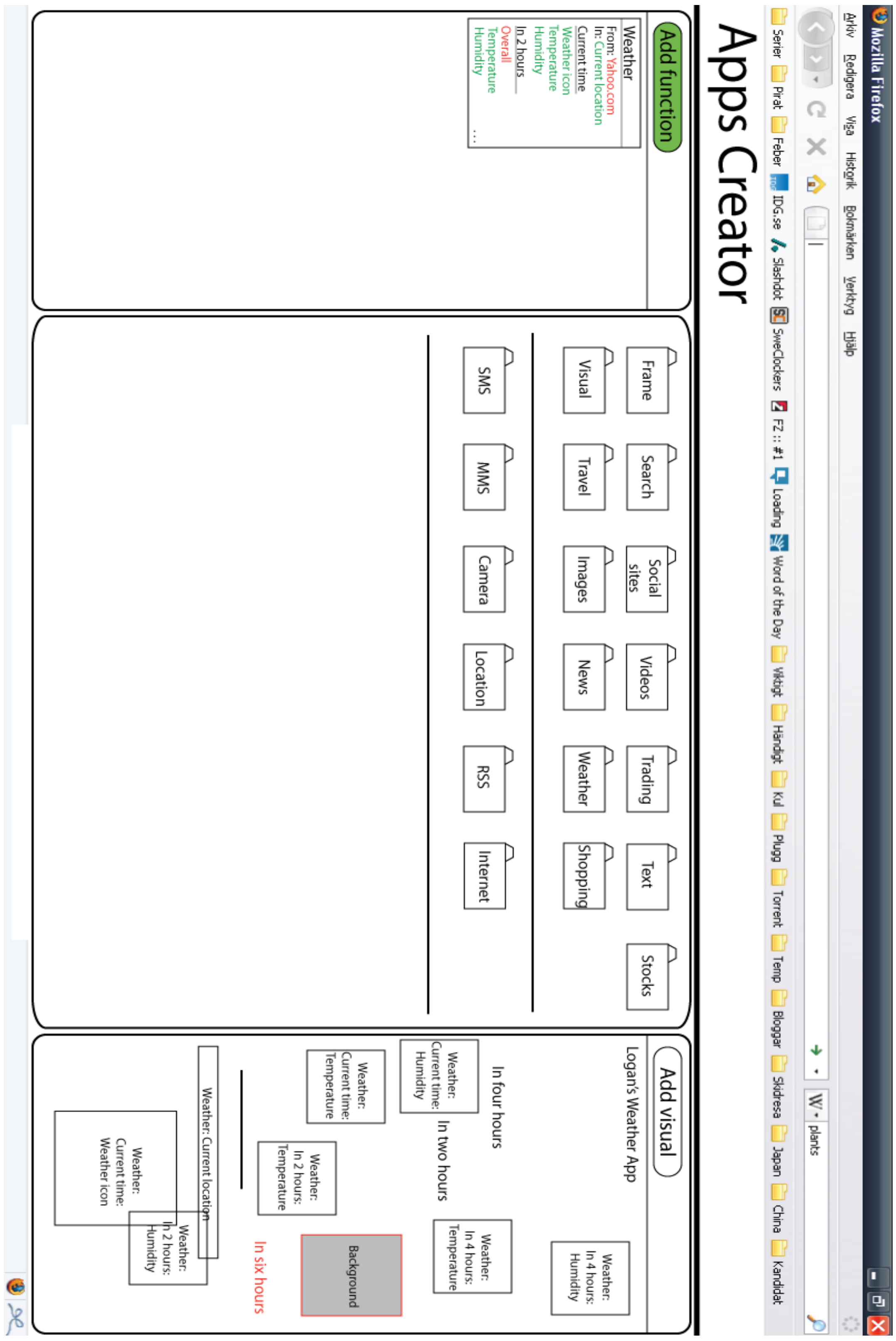
Properties

Click on a part of the window to change its settings here.

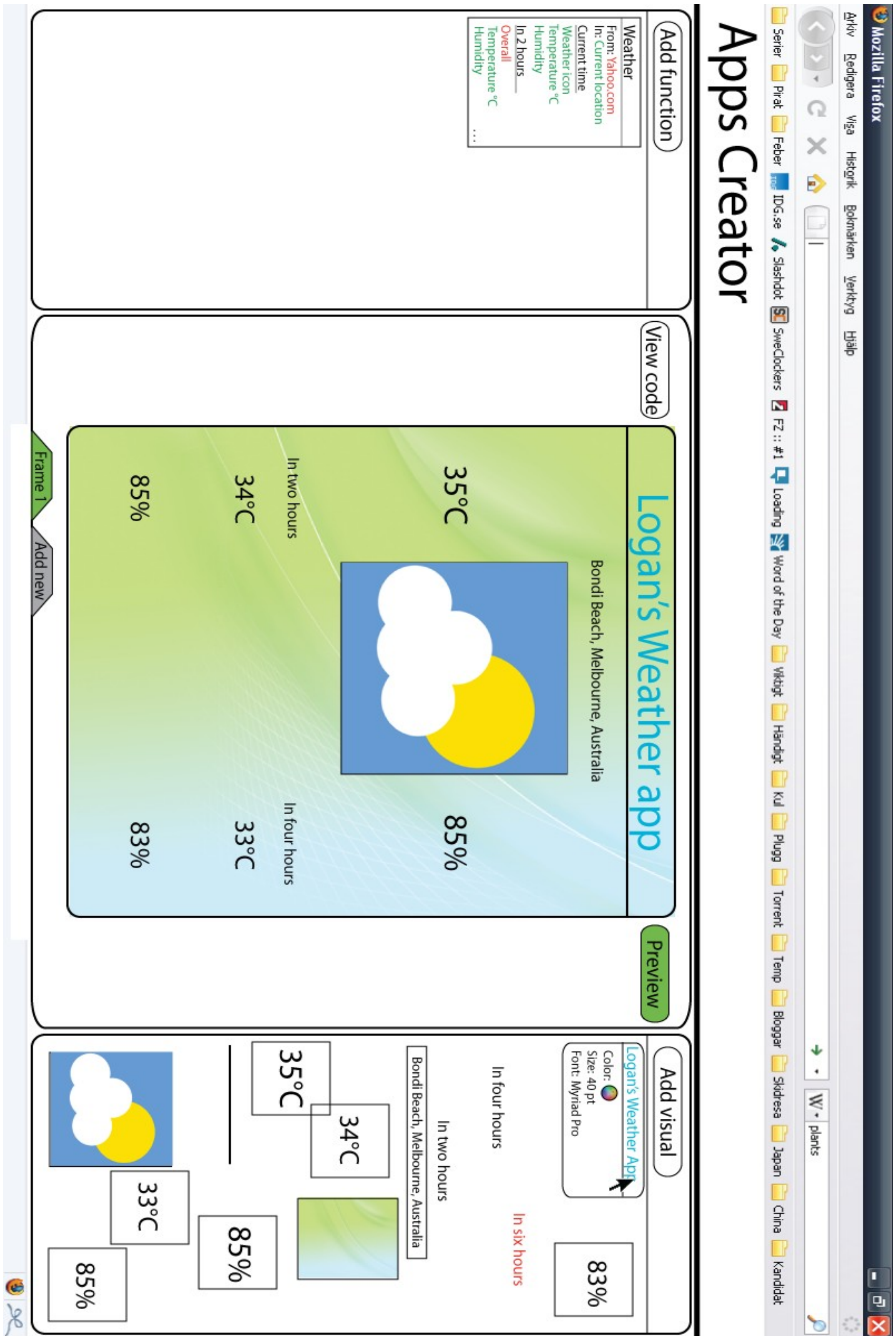
Appendix 16 – Design Proposition Wu Screen 1


Mozilla Firefox
Arkiv Redigera Visa Historik Bokmärken Verktyg Hjälp
Serier Pirat Feber IDG.se Slashdot SweClockers FZ :: #1 Loading Word of the Day Viktigt Händigt Kul Plugg Torrent Temp Bloggar Skidresa Japan China Kandidat
plants
Apps Creator
Add function
Weather
From: Yahoo.com
In: Current location
Current time
Weather icon
Temperature
Humidity
In 2 hours
Overall
Temperature
Humidity
View code
Logan's Weather app
Weather: Current location
Weather: Current time:
Weather icon
Weather: Current time: Temperature
Weather: Current time: Humidity
In two hours
Weather: In 2 hours: Temperature
Weather: In 2 hours: Humidity
In four hours
Weather: In 4 hours: Temperature
Weather: In 4 hours: Humidity
Preview
Frame 1
Add new
Add visual
Logan's Weather App
Color:
Size: 40 pt
Font: Myriad Pro
In four hours
In two hours
Background
In six hours

Appendix 16 – Design Proposition Wu Screen 2


Mozilla Firefox
Arkiv Redigera Visa Historik Bokmärken Verktyg Hjälp
plants
Serier Pirat Feber IDG.se Slashdot SweClockers FZ :: #1 Loading Word of the Day Viktigt Händigt Kul Plugg Torrent Temp Bloggar Skidresa Japan China Kandidat
Apps Creator
Add function
Weather
From: Yahoo.com
In: Current location
Current time
Weather icon
Temperature
Humidity
In 2 hours
Overall
Temperature
Humidity
...
Frame
Search
Social sites
Videos
Trading
Text
Stocks
Visual
Travel
Images
News
Weather
Shopping
SMS
MMS
Camera
Location
RSS
Internet
Add visual
Logan's Weather App
Weather: In 4 hours: Humidity
In four hours
Weather: Current time: Humidity
In two hours
Weather: In 4 hours: Temperature
Weather: Current time: Temperature
Weather: In 2 hours: Temperature
Background
In six hours
Weather: Current location
Weather: In 2 hours: Humidity
Weather: Current time: Weather icon

Appendix 16 – Design Proposition Wu Screen 3



Appendix 17 – Consent Form

Purpose	We are asking you to participate in a user test of a possible future computer system. By participating in this study, you will help us understand how parts of our designed system allow the user to work with the system. You will also help us understand how the concept of application customization for mobile phones is relevant to end users.
Study Environment	The study will take place in an office at Sony Ericsson. First you will fill in a pre-questionnaire about your experience and knowledge concerning relevant aspects of the system. Then you will be observed as you use a prototype during which we want you to speak out aloud about what you're doing and thinking. Lastly, you will be expected to fill in a questionnaire about your experience.
Information Collected	The pre-questionnaire will let us know which target group you belong to depending on your experience and knowledge of the relevant area. We may write down how you react to the system when using the prototype and we will ask you to fill out a questionnaire. The information collected will be used to analyze the system's strong and weak aspects. Your contribution will be anonymous and the information gathered will only be used for this research, not sold or shared to third-parties.
Non-disclosure Agreement	We will in this study ask you to work with concepts and ideas which are still incomplete and unannounced. Any information you acquire relating to this system is confidential, and is disclosed to you only so that you can participate in this study. By signing this form, you agree that you will not tell any of this information about the system or concept to anyone else.
Comfort	You may take a break any time you want during the study. If you have any questions, feel free to ask one of the administrators at any time.
Scenario	You will during the study act as a person who has a Blogger account and has a desire to create an application on the mobile phone enabling you to post new blog posts on said Blogger account.

If you agree with these terms, please indicate your acceptance by signing below.

Signature:___________________________________

Printed name:________________________________

Date:_______________________________________

Appendix 18 – Background Questionnaire

Please fill in the answers below with as much detail as you feel necessary.

Do you use any social networking site (please give examples)?

Answer:____

Do you read any blogs?

Answer:____

Do you have a blog yourself?

Answer:____

Have you ever created a website yourself?

Answer:____

Do you have any programming experience?

Answer:____

Do you use your cell phone for anything else than calling and texting (please specify)?

Answer:____

Have you customized any content in your cell phone (please specify)?

Answer:____

What cell phone model do you own today?

Answer:____

Appendix 19 – User Test Questionnaire

Please circle your answer, or write in the marked areas where available.

General

1. What did you think of the program?

Very bad	Bad	Good	Very good

2. How easy or difficult was it to understand what you were supposed to do in the program?

Very difficult	Difficult	Easy	Very easy

3. How easy or difficult did you find the program to use?

Very difficult	Difficult	Easy	Very easy

4. What did you think of the number of steps required to make an application?

Too few	Somewhat too few	Just about right	A few too many	Too many

5. Did the buttons do what you expected them to do?

Yes No

5a. If no, which buttons were imprecise?

6. Did you ever feel that there was not enough help documentation in the program?

Yes No

6a. If yes, where in the program?

7. Did you ever feel uncertain of what you were supposed to do?

Yes No

7a. If yes, where in the program?

____

____

____

____

8. Did you notice the process bar (which showed which step of the process you currently were in) that was displayed in every step?

Yes No

8a. If yes, did you find it helpful?

Yes No

Welcome menu

9. What did you think of the number of alternatives available in the main menu?

Too few	Somewhat too few	Just about right	A few too many	Too many

10. How clear did you find the names of the applications to be?

Very unclear	Unclear	Clear	Very Clear

11. Did you miss any kind of application?

Yes No

11a. If yes, what kind of applications did you miss?

____

____

____

____

Blog service

12. What did you think of the number of choices in the blog service menu?

Too few	Somewhat too few	Just about right	A few too many	Too many

Username/Password

13. How did you feel about entering the username and password of your blog?

Very unsafe	Unsafe	Safe	Very safe

14. Would you prefer to enter the username and password of your blog in your cell phone each time you wrote a post?

Yes No

Visual settings

15. How interesting did you think it was to be able to change the visual appearance of your application?

Completely uninteresting	Uninteresting	Interesting	Very interesting

16. Did you understand how the visual tool worked?

Yes No

General functionality

17. Are you satisfied with the functionality that is offered in the program as a whole?

Yes No

17a. If no, what kind of functionality do you miss?

Miscellaneous

18. Would you use such a program if it existed for real?

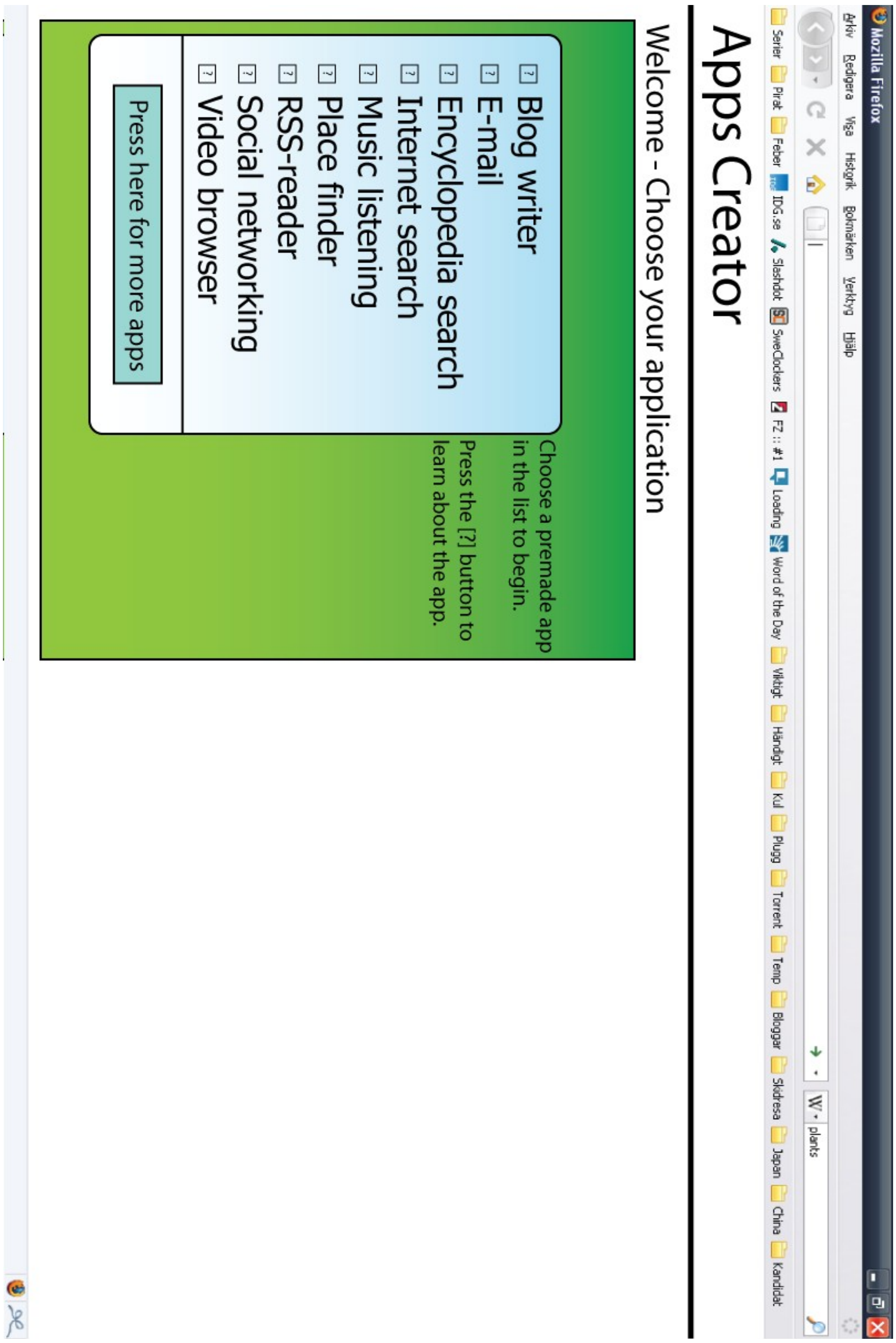
Yes No

18a. Why?

19. How big of an implication would the availability of such a tool have on your purchase of a new cell phone?

None	Small	Big	Decisive

Appendix 20 – Interactive Prototype Screen 1


Mozilla Firefox
Arkiv Redigera Visa Historik Bokmärken Verktyg Hjälp
Serier Pirat Feber IDG.se Slashdot SweClockers FZ :: #1 Loading Word of the Day Viktigt Händigt Kul Plugg Torrent Temp Bloggar Skidresa Japan China Kandidat
plants

Apps Creator

Welcome - Choose your application

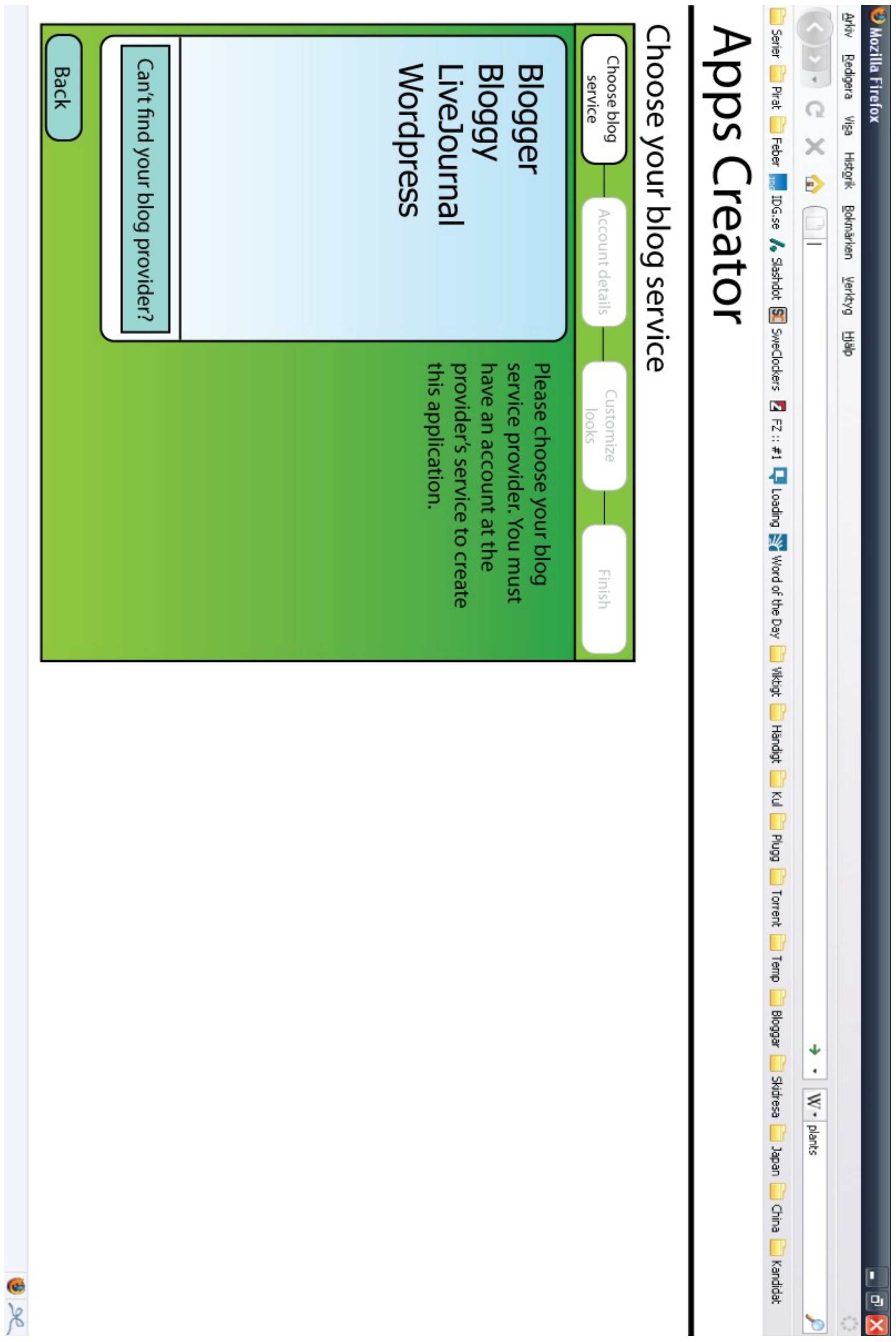
[?] Blog writer
[?] E-mail
[?] Encyclopedia search
[?] Internet search
[?] Music listening
[?] Place finder
[?] RSS-reader
[?] Social networking
[?] Video browser

Press here for more apps

Choose a premade app in the list to begin.

Press the [?] button to learn about the app.

Appendix 20 – Interactive Prototype Screen 2

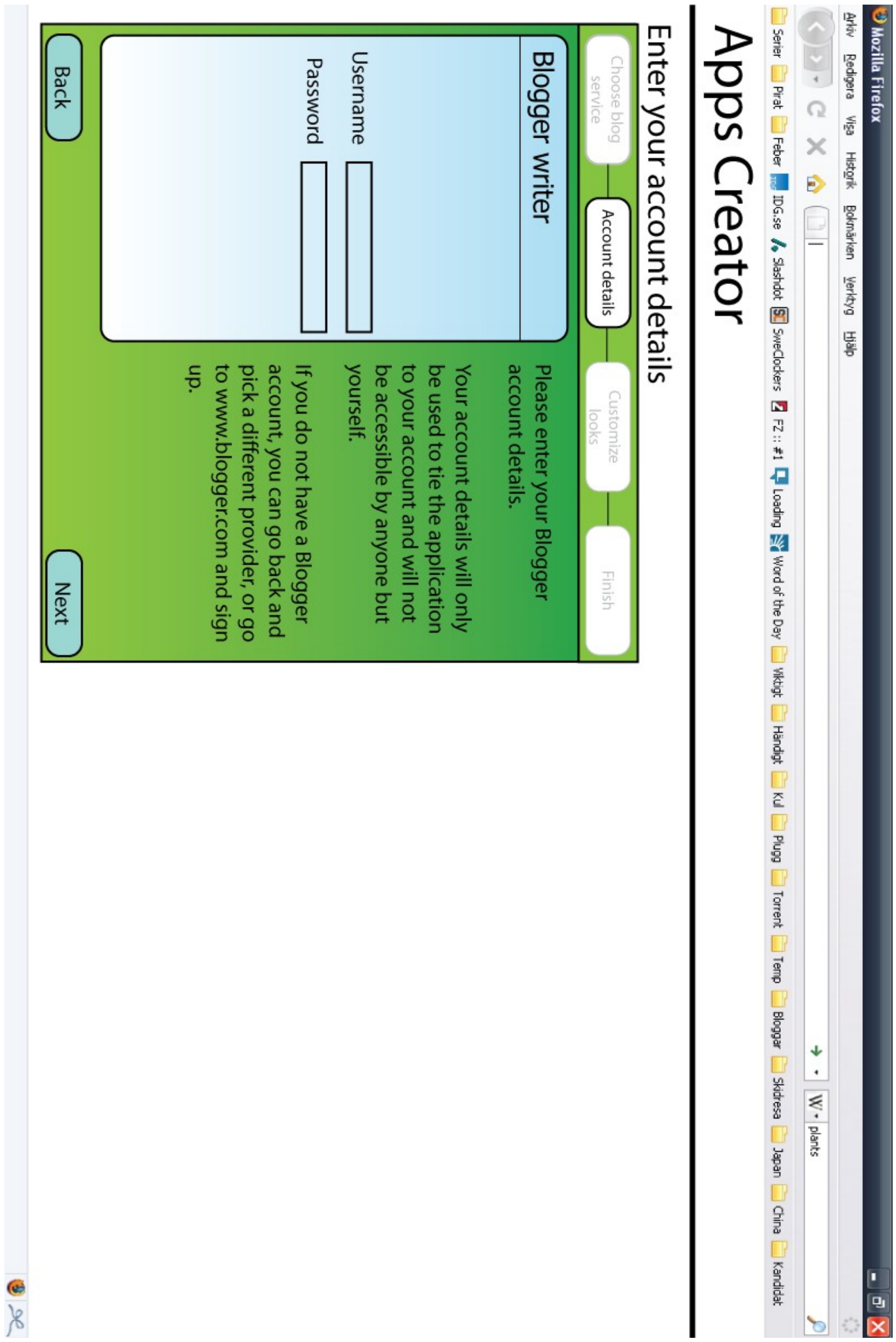

Mozilla Firefox
Arkiv Redigera Visa Historik Bokmärken Verktyg Hjälp
plants
Serier Pirat Feber IDG.se Slashdot SweClockers F2 :: #1 Loading Word of the Day Viktigt Händigt Kul Plugg Torrent Temp Bloggar Skidresa Japan China Kandidat
Choose blog service
Account details
Customize looks
Finish
Blogger
Bloggy
LiveJournal
Wordpress
Can't find your blog provider?
Back

Apps Creator

Choose your blog service

Please choose your blog service provider. You must have an account at the provider's service to create this application.

Appendix 20 – Interactive Prototype Screen 3


Mozilla Firefox
Arkiv Redigera Visa Historik Bokmärken Verktyg Hjälp
Serier Pirat Feber IDG.se Slashdot SweClockers FZ :: #1 Loading Word of the Day Viktigt Händigt Kul Plugg Torrent Temp Bloggar Skidresa Japan China Kandidat
plants
Choose blog service
Account details
Customize looks
Finish
Back
Next

Apps Creator

Enter your account details

Blogger writer

Username

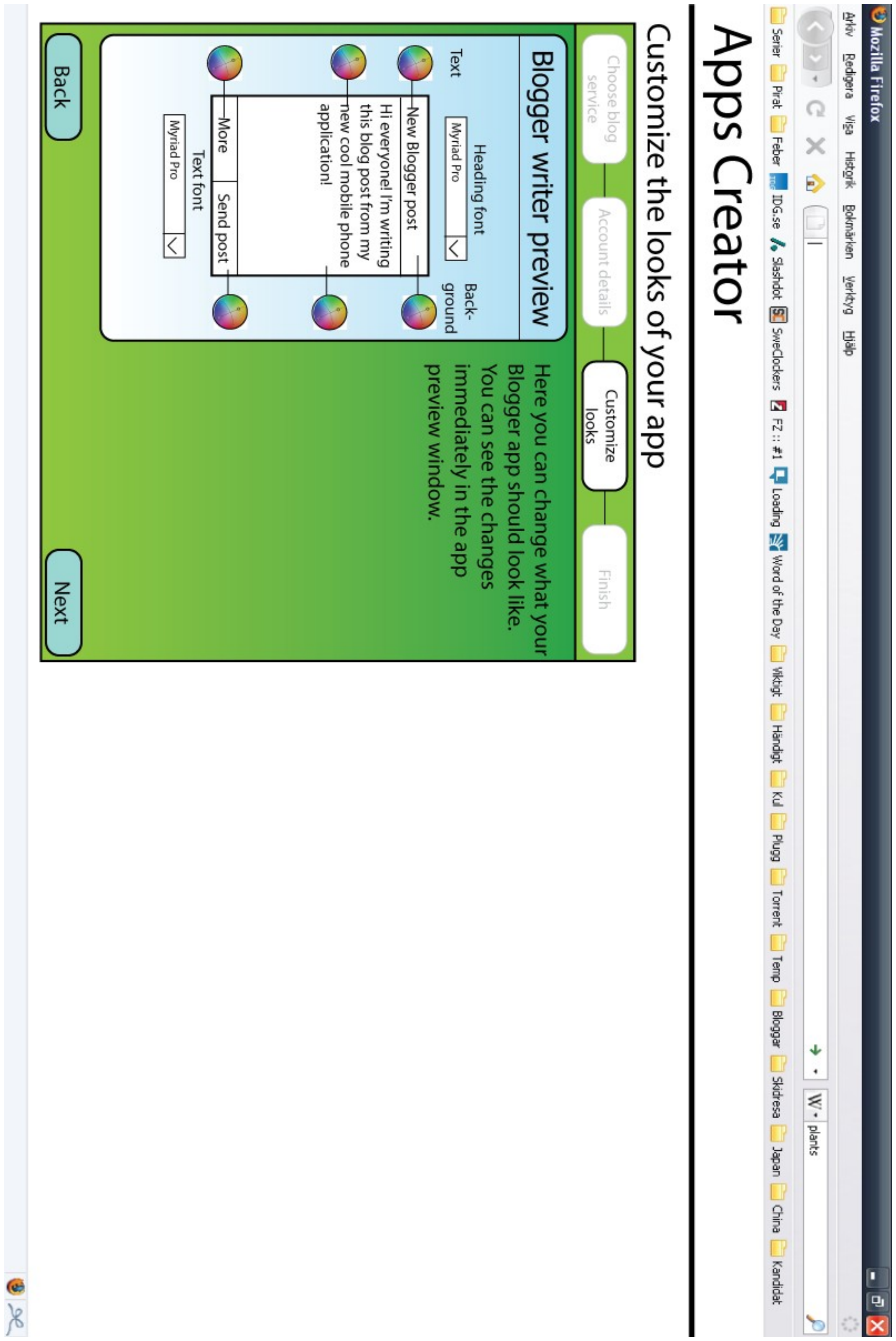
Password

Please enter your Blogger account details.

Your account details will only be used to tie the application to your account and will not be accessible by anyone but yourself.

If you do not have a Blogger account, you can go back and pick a different provider, or go to www.blogger.com and sign up.

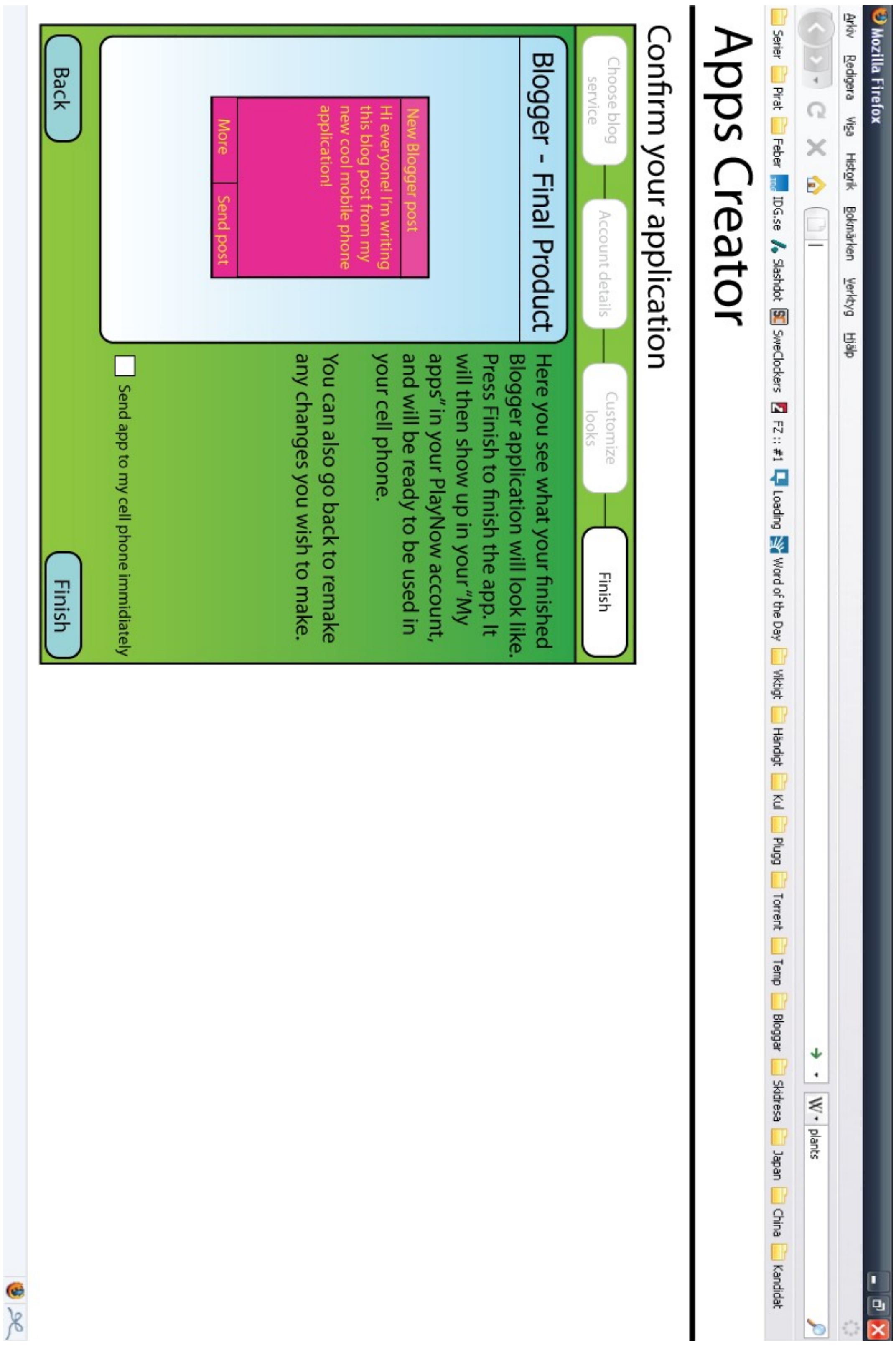
Appendix 20 – Interactive Prototype Screen 4


Mozilla Firefox
Arkiv Redigera Visa Historik Bokmärken Verktyg Hjälp
plants
Serier Pirat Feber IDG.se Slashdot SweClockers FZ :: #1 Loading Word of the Day Viktigt Händigt Kul Plugg Torrent Temp Bloggar Skidresa Japan China Kandidat
Apps Creator
Customize the looks of your app
Choose blog service
Account details
Customize looks
Finish
Blogger writer preview
Heading font
Myriad Pro
Text
Back-ground
New Blogger post
More
Send post
Text font
Myriad Pro
Back
Next

Hi everyone! I'm writing this blog post from my new cool mobile phone application!

Here you can change what your Blogger app should look like. You can see the changes immediately in the app preview window.

Appendix 20 – Interactive Prototype Screen 5


Mozilla Firefox
Arkiv Redigera Visa Historik Bokmärken Verktyg Hjälp
plants
Serier Pirat Feber IDG.se Slashdot SweClockers FZ :: #1 Loading Word of the Day Viktigt Handigt Kul Plugg Torrent Temp Bloggar Skidresa Japan China Kandidat
Choose blog service
Account details
Customize looks
Finish
More
Send post
Back
Finish

Apps Creator

Confirm your application

Blogger - Final Product

New Blogger post

Hi everyone! I'm writing this blog post from my new cool mobile phone application!

Here you see what your finished Blogger application will look like. Press Finish to finish the app. It will then show up in your "My apps" in your PlayNow account, and will be ready to be used in your cell phone.

You can also go back to remake any changes you wish to make.

☐ Send app to my cell phone immidiately

Appendix 21 – Results of Questionnaires: Comments

Advanced User Group		**Simple User Group**	
Testperson 1		Testperson 5	
5a	The balls – that you could press them.	5a	
6a		6a	
7a	Are others or me supposed to use it?	7a	
11a	Non-network/Internet apps	11a	
17a	[For the blog]: Must see history (mine + other people's posts)	17a	
18a	Lacks good examples, unclear what is required from me, and what the results might be.	18a	I don't blog.
Testperson 2		Testperson 6	
5a	New blogger post and the font.	5a	
6a	How will the interaction design be, what's under “more”?	6a	
7a		7a	
11a	Games, animation.	11a	
17a	More creativity when it comes to graphics.	17a	
18a	I can write write my own java midlet instead.	18a	It's a great tool working with if you have something you want to share with everyone.
Testperson 3		Testperson 7	
5a		5a	
6a	The view where you choose font and background color felt somewhat cluttered.	6a	
7a	See 6a.	7a	
11a	Did not really miss anything, but an image processing app would've been suitable.	11a	
17a		17a	
18a	Yes, at least to explore.	18a	
Testperson 4		Testperson 8	
5a		5a	However, in the “welcome page”, it might be unclear that you're supposed to press the name of the application in order to choose this. To tick boxes might by preferable, especially if the program is developed further so that you can choose

			several applications at once. And I don't know what an RSS-reader is.
6a	Unclear what was instructions and what was the application.	6a	
7a	Thought the program never started – only instructions	7a	When you were supposed to change the looks of the application. It looks a but messy with the color palettes.
11a		11a	
17a		17a	
18a	Good to be able to create your own apps for the telephone.	18a	To listen to music via the internet.
Testperson 10		Testperson 9	
5a		5a	But I would have preferred arrows from the color palettes, as well as explaining text such as “Choose font, color and background”.
6a		6a	See 5a.
7a		7a	Customize looks.
11a		11a	
17a		17a	
18a	I still don't use my cell phone for very much, prefer to do most things on the computer. Might happen that I become more “mobile” in the future, in which case such an application might be of interest.	18a	
		Testperson 11	
		5a	Color buttons – what does the “more” mean? RSS – What is it?
		6a	It was there but could've been clearer
		7a	
		11a	
		17a	(Fonts) Icon/Pictures – links to own pics/videos, enter external links, box for underlining/italic/bold
		18a	I'm busy having a real life instead of a cyber one.

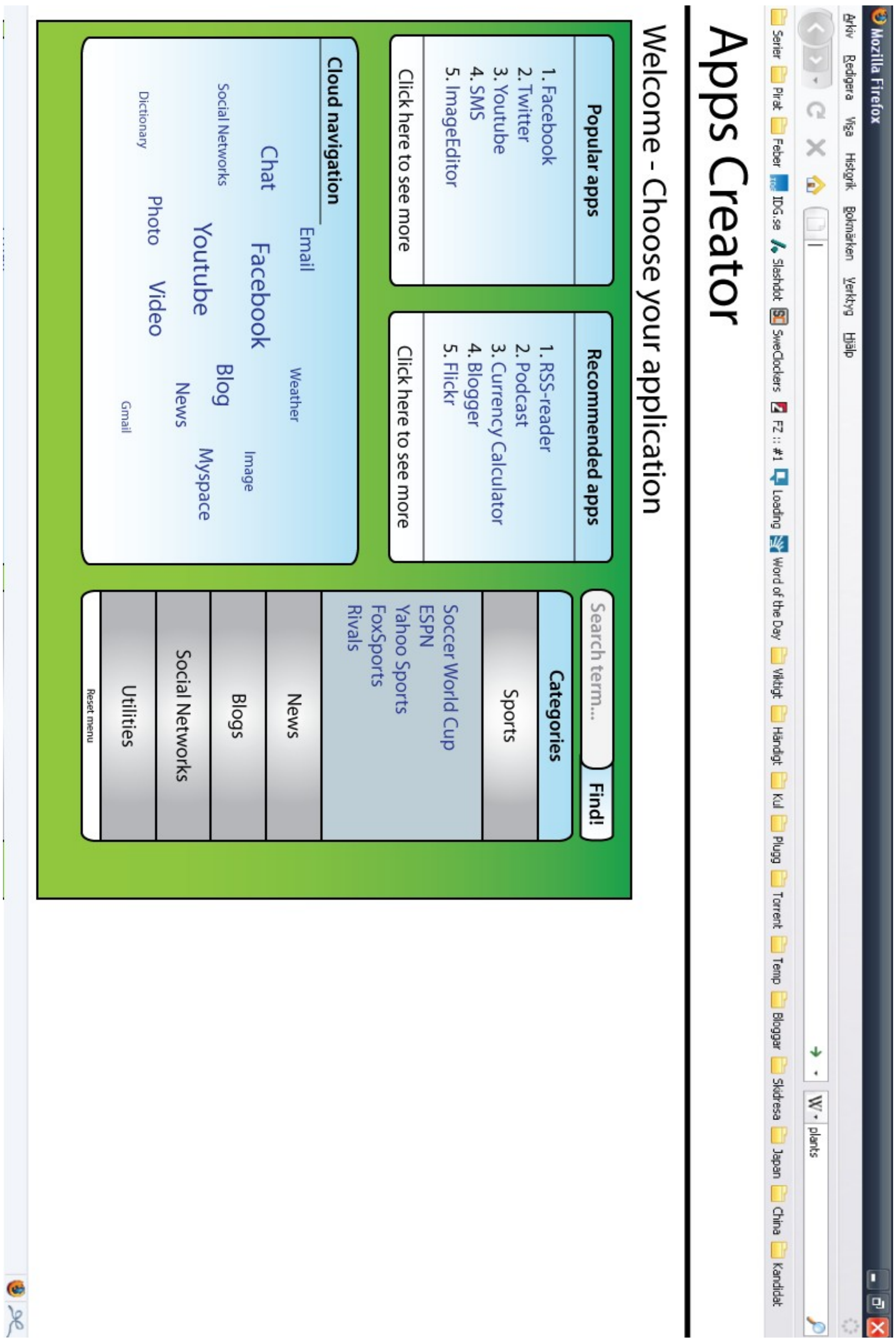
Appendix 22 – List of Observational Notes

- User is uncertain regarding the password.
- User thinks the instructions are written in a rather pretentious way.
- User is uncertain regarding what is information and instructions and what you are expected to actually do. Expected Next → Next → Begin sequence, and not the wizard-like mode.
- User thinks there is very much text with unnecessary information that makes the text uninteresting.
- User wonders what Finish does.
- User does not quite understand what the application is for. Is it personal, or for sharing with other people?
- User does not see the personal value of customization.
- User would like to have a direct link to My Apps.
- User would've liked to have an introductory step with an example.
- User thinks it's easy to get the impression that one can only make internet apps.
- User wonders why Blogspot is not in the Blog service provider list.[1]
- User would have preferred to enter username and password at the end of the creation process.
- User would like to have more functionality in the blog application.
- User would like the tool to be more similar to other online tools when it comes to login.
- User would like to have more advanced functionality, to hook apps, etc.
- User is uncertain about how the application connects with the telephone in the finish-step.
- User is uncertain about the example sentence in the visual editor, as well as how the fonts are managed and the name about the app. Wonders if the font chosen and application title is the font and title the blog post will have.
- User would like to have more information about how the app works.
- User thinks it is difficult to read the process bar.
- User feels locked in and restricted by the premade apps.
- User would like to be able to make games and training apps for example.
- User wants to be able to choose functions such as GPS, Network, etc.
- User wants more functionality in the visual, more creativity.
- User would like to see a tool like this in reality, but thinks it would have to much more flexible.
- User thinks the system should enable people to invest a lot of time in graphics, and as little time as possible in functionality.
- User is uncertain about what the colorful balls do in the visual editor.
- User wants text and background to be more clear and structured in the visual editor.
- User would prefer to choose what the blog post would look like, and not so much the application itself.
- User is uncertain about what Finish does.
- User thinks the functionality of the application is unclear – can it send pictures?
- User thinks the application feels like information before you actually start using something – like an introduction.

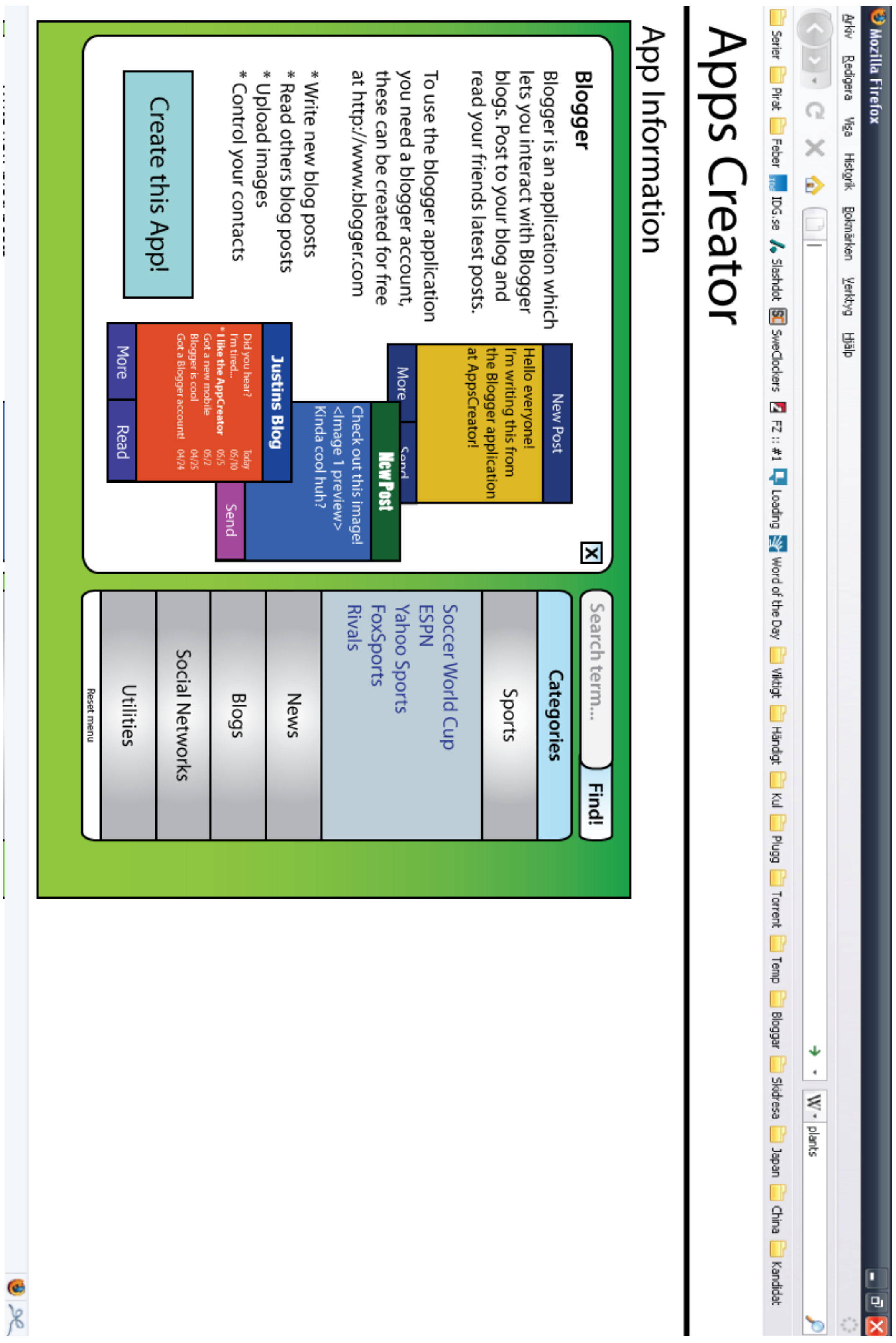
1. Blogspot is a subdomain of Blogger.

- User thinks you should force users of the system to interact and play around with the system.
- User wants the system to look more like an editor.
- User thought it felt like a wizard.
- User wondered what an RSS-reader is.
- User thought the questionmark boxes in the welcome menu were tick boxes.
- User thought it was too easy for a process bar to be necessary.
- User did not understand how the visual tool worked immediately, but once the user did understand, the user thought it was very easy.
- User missed some form of image application. Asked if the blog application can upload images as well.
- User felt the visual tool was cluttered.
- User wondered what PlayNow is.
- User did not read the instructions.
- User thought the user hadn't actually done anything once finished; thought it was "too simple".
- User wondered what the color palettes did. Wanted arrows instead of lines.
- User was generally confused with the concept of blog service providers, blogs, and the program itself.
- User wanted text saying "Click the color palettes" as instruction, but did try to click the color palettes even if it did not say so. Once it was explained what happened, the user thought it was very simple.
- User though the questionmark boxes were tick boxes.
- User didn't read the "text" or "background" labels in the visual tool and was somewhat confused as a result of this.
- User did not notice the process bar until the user was in the visual tool.
- User wanted arrows in the process bar instead of lines.
- User did not know what RSS is.
- User did not immediately understand how the visual tool worked, but did so rather quickly nonetheless and thought it was simple once understood.
- User thought it was a good idea to have the questionmark boxes once the user was informed that they were not tick boxes.
- User especially liked the concept of the Place Finder App.
- User liked the "Can't find your blog provider"-button.
- User did not know that app was short for application.
- User wondered what PlayNow is.
- User wanted some sort of mascot to increase inspiration, creativity and exploration.
- User went through everything very quickly, without any doubt, without and questions.
- User did not read the instructions when there was a lot of text.
- User went through it all very quickly and without any issues, but thought it was unnecessary to be able to set different colors for the soft keys.
- User did not know what an RSS-reader is.
- User was somewhat uncertain what two categories in the welcome menu contained and included.
- User missed the existence of the text font.
- User wanted to be able to make the text bold, italic, or underlined.
- User thought the system felt very basic and aimed at tweenies.

Appendix 23 – Final Design Proposition Screen 1



Appendix 23 – Final Design Proposition Screen 2



Blogger

Blogger is an application which lets you interact with Blogger blogs. Post to your blog and read your friends latest posts.

To use the blogger application you need a blogger account, these can be created for free at http://www.blogger.com

* Write new blog posts
* Read others blog posts
* Upload images
* Control your contacts

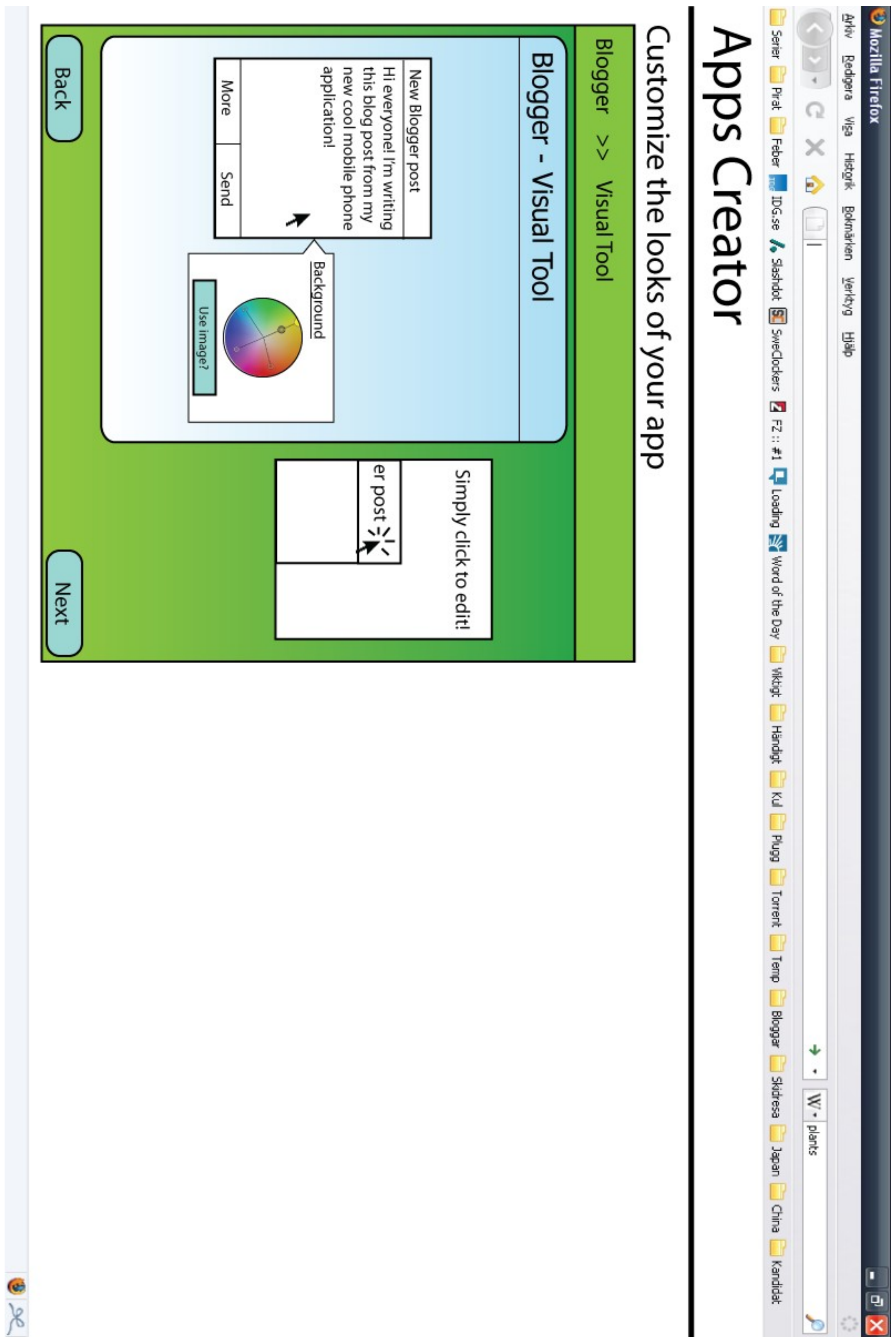
Hello everyone!
I'm writing this from the Blogger application at AppsCreator!

Check out this image!
<Image 1 preview>
Kinda cool huh?

Did you hear?	Today
I'm tired...	05/10
* **I like the AppCreator**	05/5
Got a new mobile	05/2
Blogger is cool	04/25
Got a Blogger account!	04/24

Soccer World Cup
ESPN
Yahoo Sports
FoxSports
Rivals

Appendix 23 – Final Design Proposition Screen 3


Mozilla Firefox
Arkiv Redigera Visa Historik Bokmärken Verktyg Hjälp
plants
Serier Pirat Feber IDG.se Slashdot SweClockers F2 :: #1 Loading Word of the Day Viktigt Händigt Kul Plugg Torrent Temp Bloggar Skidresa Japan China Kandidat

Apps Creator

Customize the looks of your app

Blogger >> Visual Tool

Blogger - Visual Tool

New Blogger post

Hi everyone! I'm writing this blog post from my new cool mobile phone application!

More

Send

Background

Use image?

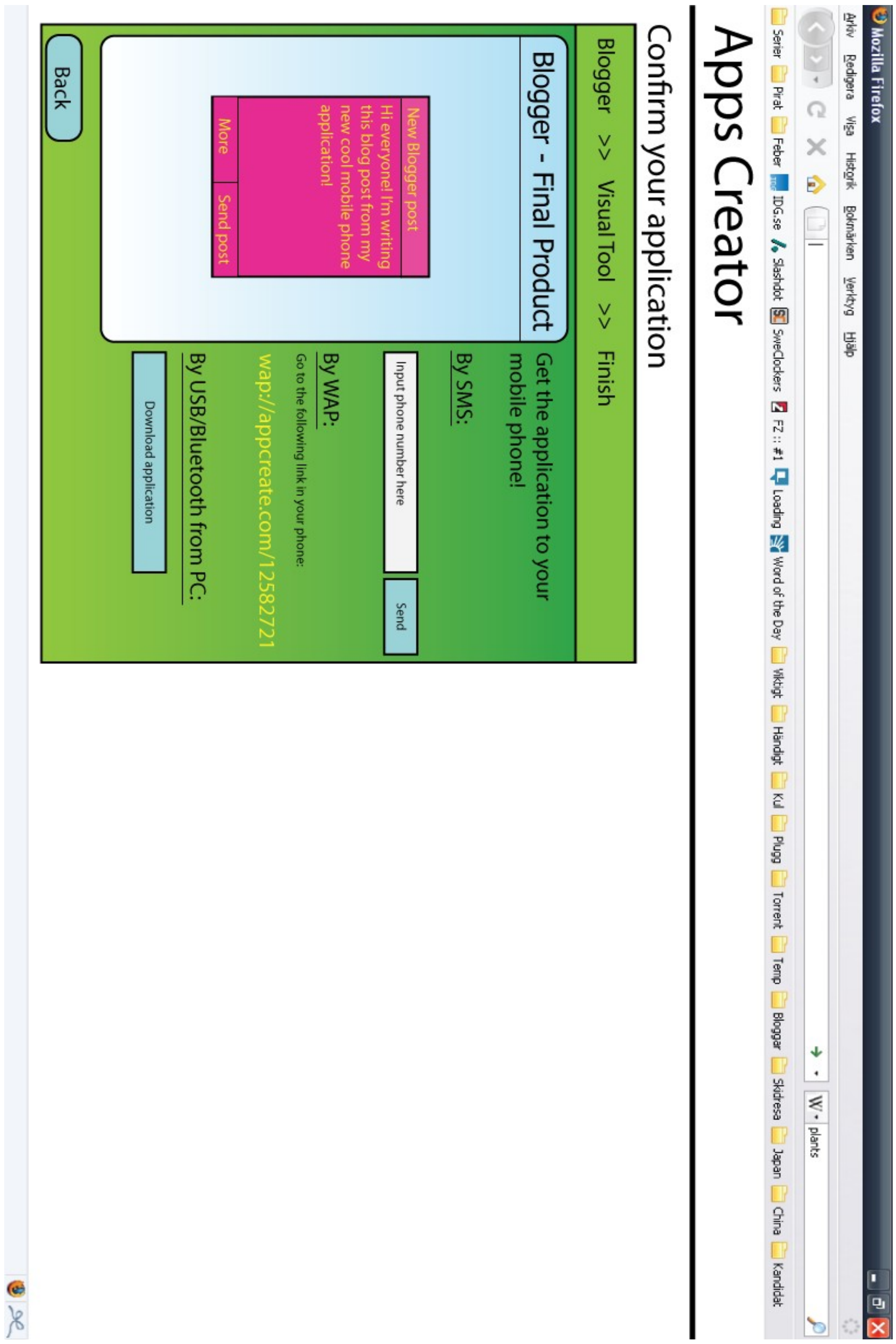
Simply click to edit!

er post

Back

Next

Appendix 23 – Final Design Proposition Screen 4


Mozilla Firefox
Arkiv Redigera Visa Historik Bokmärken Verktyg Hjälp
plants
Serier Pirat Feber IDG.se Slashdot SweClockers FZ :: #1 Loading Word of the Day Viktigt Händigt Kul Plugg Torrent Temp Bloggar Skidresa Japan China Kandidat

Apps Creator

Confirm your application

Blogger >> Visual Tool >> Finish

Blogger - Final Product

New Blogger post

Hi everyone! I'm writing this blog post from my new cool mobile phone application!

More | Send post

Get the application to your mobile phone!

By SMS:

Input phone number here | Send

By WAP:

Go to the following link in your phone:

wap://appcreate.com/12582721

By USB/Bluetooth from PC:

Download application

Back

References

Aouiche, K., Lemire, D. & Godin, R., 2008. *Collaborative OLAP with tag clouds: web 2.0 OLAP formalism and experimental evaluation*. [Online]
Available at: http://arxiv.org/PS_cache/arxiv/pdf/0710/0710.2156v2.pdf [Accessed 6 May 2009].

Baldwin, C. Y. & Clark, K. B., 2000. *Design rules: the power of modularity*. Vol. 1. Cambridge (MA): The MIT Press.

Brandweek, 2009. Study: value trumps price among shoppers. *Brandweek*, [Online]. Available at: http://www.brandweek.com/bw/special-reports/brand-key/2009/index.jsp [Accessed 25 May 2009]

Carroll, J. M., 2000. *Making use: scenario-based design of human-computer interactions*. Cambridge (MA): The MIT Press.

Chang, C.-C. & Chen, H.-Y., 2009. I want my products my own way, but which way? The effects of different product categories and cues on customer responses to web-based customizations. *CyberPsychology & Behavior*, 12(1), pp. 7-14.

Cohn, M., 2004. *User stories applied: for agile software development*. Boston: Addison Wesley.

Cooper, A., 2004. *The inmates are running the asylum: why high-tech products prive us crazy and how to restore the sanity*. Indianapolis: Sams Publishing.

Cummins, R. A. & Gullone, E., 2000. Why we should not use 5-point Likert scales: the case for subjective quality of life measurement. *Proceedings, Second International Conference on Quality of Life in Cities*, pp. 74-93.

Dawes, J., 2008. Do data characteristics change according to the number of scale points used? An experiment using 5-point, 7-point and 10-point scales. *International Journal of Market Research*, [Online]. 50(1), pp. 61-78. Abstract from John Dawes website.
Available at: http://www.johndawes.com.au/publications/publications.html#Link1, John Dawes. [Accessed 28 April 2009].

Dhar, R. & Simonson, I., 2003. The Effect of Forced Choice on Choice. *Journal of Market Research*, 40(2), pp. 146-160.

Dumas, J. S. & Redish, J. C., 1999. *A practical guide to usability testing*. Rev. ed. Exeter: Intellect Books.

Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A.G. & Mehandjiev N., 2004. Meta-design: a manifesto for end-user development. *Communications of the ACM*, 47(9), pp. 33-37.

Google Scholar, 2009. *Nielsen: The usability engineering life cycle*. [Online]
Available at: http://scholar.google.se/scholar?hl=sv&lr=&client=firefox-a&cites=14129673801500960859 [Accessed 6 May 2009].

Grudin, J. & Pruitt, J., 2002. Personas, participatory design and product development: an infrastructure for engagement. *Proceedings of The Participatory Design Conference*, pp. 144-161.

Jacobsen, D. I., 2002. *Vad, hur och varför: om metodval i företagsekonomi och andra samhällsvetenskapliga ämnen*. Lund: Studentlitteratur.

Kahler, H., 2001. More than WORDs: collaborative tailoring of a word processor. *Journal of Universal Computer Science*, 7(9), pp. 826-847.

Kindborg, M. & McGee, K., 2007. Visual programming with analogical representations: inspirations from a semiotic analysis of comics. *Journal of Visual Languages & Computing*, 18(2), pp. 99-125.

Korpipää, P., Malm, E.-J., Rantakokko, T., Kyllönen, V., Kela, J., Mäntyjärvi, J., Häkkilä, J. & Känsälä, I., 2006. Customizing User Interaction in Smart Phones. *Pervasive Computing*, 5(3), pp. 82-90.

Lieberman, H., 2001. *Your wish is my command: programming by example*. San Francisco: Morgan Kaufmann.

Lieberman, H., Paternó, F., Klann, M. & Wulf, V., 2006. End-user development: an emerging paradigm. In H. Lieberman, F. Paternó & V. Wulf, eds. *End user development*. Dordrecht: Springer, pp. 1-8.

Mackay, W.E., 1991. Triggers and barriers to customizing software. *Proceedings of CHI'91 Conference on Human Factors in Computing Systems*, pp. 153-160.

Mahatanankoon, P., Wen, H. J. & Lim, B., 2005. Consumer-based m-commerce: exploring consumer perception of mobile applications. *Computer Standards & Interfaces*, 27(4), pp. 347-357.

McGill, T., 2004. The effect of end user development on end user success. *Journal of Organizational and End User Computing*, 16(1), pp. 41-58.

Mozilla, 2008. *XUL*. [Online]
Available at: https://developer.mozilla.org/en/XUL [Accessed 3 May 2009].

Nielsen, J., 1992. The usability engineering life cycle. *IEEE Computer*, 25(3), pp. 12-22.

Oates, B. J., 2006. Researching information systems and computing. London: SAGE Publications Ltd.

Oppermann, R. & Simm, H., 1994. Adaptability: user-initiated individualization. In R. Oppermann, ed. *Adaptive user support: ergonomic design of manually and automatically adaptable software*. Hillsdale: Lawrence Erlbaum Associates, pp. 14-64.

Ousterhout, J. K., 1998. Scripting: higher level programming for the 21st century. *IEEE Computer*, 31(3), pp. 23-30.

Page, S. R., Johnsgard, T. J., Albert, U. & Allen, C. D, 1996. User customization of a word processor. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Common Ground*, pp. 340-346.

Pane, J. F. & Myers, B. A., 2006. More natural programming languages. In H. Lieberman, F. Paternó & V. Wulf, eds. *End user development*. Dordrecht: Springer, pp. 31-50.

Piller, F. T., 2005. Mass customization: reflections on the state of the concept. *The International Journal of Flexible Manufacturing Systems*. 16(4), pp. 313-334.

Pine II, B. J., 1993. Mass customization : the new frontier in business competition. Boston: Harvard Business School Press.

Preece, J., Rogers, Y. & Sharp, H., 2007. *Interaction design: beyond human-computer interaction*. 2nd ed. Chichester: John Wiley & Sons.

Pruitt, J. & Adlin, T., 2006. *The persona lifecycle: keeping people in mind throughout product design*. San Francisco: Elsevier Academic.

Rivera, D., 2005. The effect of content customization on learnability and perceived workload. *CHI '05 extended abstracts on Human factors in computing systems*, [Online], pp. 1749-1752. Abstract from ACM Portal database. Available at: http://portal.acm.org/citation.cfm?id=1056808.1057013&coll=GUIDE&dl=GUIDE&CFID=36113811&CFTOKEN=98804810, ACM Portal. [Accessed 6 May 2009].

Seidman, I., 2006. *Interviewing as qualitative research : a guide for researchers in education and the social sciences*. 3rd ed. New York: Teachers College Press.

Sony Ericsson, 2008. Profile. [Online] Available at: http://www.sonyericsson.com/cws/corporate/company/aboutus/profile [Accessed 03 May 2009].

Subramanya, S. R. & Yi, B. K., 2005. Mobile Content Customization. *IEEE MultiMedia*, 12(4), pp. 103-104.

Szyperski, C., 2002. *Component software: beyond object-oriented programming*. 2nd ed. London: Addison-Wesley.

Tseng, M. M & Jiao, J., 1998. Design for mass customization by developing product family architecture. *Proceedings of DETC'98*, DETC98/DTM-5717.

Tseng, M. M & Jiao, J., 2001. Mass customization. In G. Salvendy, ed. *Handbook of Industrial Engineering*. 3rd ed. New York: John Wiley & Sons, pp. 684-709.

Wang, A. J. A. & Qian, K., 2005. *Component-oriented programming.* Hoboken: John Wiley & Sons.

Won, M., Stiemerling, O. & Wulf, V., 2006. Component-based approaches to tailorable systems. In H. Lieberman, F. Paternó & V. Wulf, eds. *End user development*. Dordrecht: Springer, pp. 115-141.