Interaction Design with Regards to Mobile Rich Internet Applications

Hani Sean Movahed
This thesis was submitted to the Department of Computer Science at Luleå University of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

Interaction design with regards to mobile Rich Internet Applications

Hani, Sean, Movahed

30 May, 2008

Examiner:  Kåre Synnes  
CSEE, Luleå University

Supervisor:  Karl-Johan Lundkvist  
Ericsson Research
Abstract

Indisputably, the area concerning mobile computing and networking is reaching its stride; convincing the developers that, in the near future, the vast majority of devices will be portable and wireless. This evolution will inevitably render current standards obsolete since only powerful servers will remain on the wired side of the Internet.

Although, much progress has been made in terms of technological innovations; many of the present software solutions are difficult to use and lack accessibility, flexibility, and robustness. If you prefer, they offer a limited, and in some cases even poor, user experience. Theoretical analysis of usability joint with new innovative interaction designs can help achieve significant presentation improvements in terms of attractive and immersive interfaces; contrary to the rigid organizational structures imposed by current applications. Also by introducing Rich Internet Applications, referred to as widgets, based on established web technologies provides the ability to develop more efficient and effective applications. This relatively new kind of application has seen significant proliferation onto the desktop and more recently the web enabled portable device market.

To introduce a suitable solution current platforms and user requirements are analyzed. The gathered intelligence represents the design foundation which is transformed into a conceptual model. Consequently, the shared base, design and ideology, logically implies similarities in presentation; however, the subtle differences between the suggested prototypes become more evident while discussing their emphasis and individual bearing on the final product. The results from this technique are deployed in the final solution, which is literally a direct derivative of several user interface designs. In a few words, adapting such a multidisciplinary concept, effective borrowing and user needs, suggests ways of meeting the ongoing stride by capitalizing on the target audience knowledge and available solutions.

Author keywords: Widget, runtime engine, platform dependent, user experience, design, interaction design
Acknowledgments

First and foremost, I wish to express my sincere thanks to colleagues and friends for their invaluable support and contributions during the project. Thanks to Kåre Synnes for the guidance and recommendation that eventually led to the opportunity of working amongst creative people. To my supervisor Karl-Johan Lundkvist and Johan Kristiansson at Ericsson Research for constructive criticism and suggestions throughout the entire process.

I would also like to thank the people who read at least parts of this master’s thesis and provided many comments, making this thesis better and in essence more complete than it would have otherwise been. Last but not least, a hearty thanks to my family who always encourages and believes in my work.
Preface

This thesis draws on the experience of established technologies and principles to provide a thorough grounding in interaction design and platform associated dependencies for portable devices. It will not cover every aspect of development as there are plenty of good books and articles on the subjects. However, it will provide the necessary foundation for constructing two prototypes under native and browser platforms.

The thesis is roughly divided into four distinct parts:

Part one, Chapters 1 and 2, introduces the topic at a very general level and finishes off with narrowing down the subject to the specific area covered by this thesis.

Part two, Chapters 3 and 4, explains the principles and design decisions involved while constructing interactive prototypes. These concepts are then adapted in various case studies that focus on some of the more practical dilemmas and problems that arise when working on an actual design project.

Part three, Chapters 5 and 6, provides the development techniques and paradigms necessary to implement the previously covered design proposals. The different models are then applied to one of the presented design solutions to measure their acceptability and usability in terms of the criteria discussed in the previous part.

Part four, Chapter 7 and 8, concludes the conducted studies and explores future research opportunities.
Audience

The reader is expected to have a basic ground in software development and design, but not necessarily for portable devices. This study provides a general review of both of these necessities, however, for more details regarding each subject please refer to the referenced material. As mentioned the thesis maintain that the reader has some understanding of usability testing, user interface design, and is interested in learning how to apply those principles in the real world.

Beside developers this study may be of interest to other audiences such as:

- Readers who may be looking for techniques to exploit the potential of portable solutions, using Symbian or browser technology.

- Usability specialists, consultants, or designers who want to explore new presentation opportunities, in which case this thesis will act as the foundation to reach such ambitions.

- Anyone with the slightest interest in the subject, since it covers essential problems in current generation of portable applications and interaction design.
Chapter 1

Introduction

Since the introduction of the Internet the sheer volume of user generated content has grown. This has arguably affected the quality of the material as well as distorted the creation/consumption balance. As publications accept more content from the public the distinct line between editorially and user produced content is now slowly fading away. What once was regarded as a one-way street of information flow has now changed, allowing the former consumer to contribute and be a part of that content creation process.

The flood of user generated content has its benefits; it provides the user with alternative sources and views on the same topic. Needless to say, the oversupply of content renders the significance of overview as an even more important consumer requirement. As a user in front of a computer there are certain websites that are generally monitored on a day-to-day basis and chances are that the vast collection of information results in an overwhelming and confusing experience. This is a well known problem and was partially addressed through the introduction of Web 2.0. Stephen Fry of the British Guardian newspaper describes the concept as: “...an idea in people’s heads rather than a reality. It’s actually an idea that the reciprocity between the user and the provider is what’s emphasized. In other words, genuine interactivity, if you like, simply because people can upload as well as download” [5].

As many essential theories Web 2.0 is a soft concept and lacks a standard or a definition. The absence of a hard boundary is, however, compensated with a gravitational core. The core is then associated with a set of principles and practices that tie together a solar system where the varying distance from the center constitutes its effect and importance. So the strict answer to the question “what is Web 2.0?” a service that satisfies and follows the outline set by the solar system and the related principles; for further reading and a more comprehensive coverage of the web principles refer to [10] and [11].

The term Web 2.0 is much more than a new facelift onto an exciting application. It is designed to act as an outline that provides a new perspective on the entire business model, from software concept through deployment of the service. The concept thrives on network interaction; services that get richer in terms of content
the more people network and share, applications that adapt to their user base, and broader computing services through application interaction [12].

The concept of widgets is a descendent of the Web 2.0 model and what is referred to as “harnessing the collective intelligence” [29]. That is a platform that provides inventories of small footprint applications with access to commonly utilized services regardless of the provider. For developers, widgets are created using the same open technologies used to create web pages, which extend the ability to contribute to a very wide audience. This means that, in the majority of instances, the distinct boundary between consumer and developer is eliminated. In addition, the desired material from various sources are presented in rather summarized and concise form; eradicating the multiple steps that is otherwise required in order to track down the wanted information [1].

Thus far, only the editors have had the opportunity to, through a selective process, handpick the best from user and editorially produced content. However, the notion of widgets presents the same opportunity to the user, by providing the necessary platform and the level of personalization requested by the consumers. This is achieved through the adaptation of an appealing and user friendly content discovery and delivery system that conceals the complexity of the core [23].

1.1 Problem

While the worldwide cell phone market continues its substantial year-over-year growth so does the demands with regards to design, interface, multimedia capability, technical specification, and software. Limited by very scarce resources in terms of memory, CPU, display area, etc., portable device development is generally more difficult than developing desktop applications; with performance being one of the top critical requirements.

As object-oriented technology is commonly the prevalent programming paradigm when developing embedded software, the majority of the available mobile applications are developed with this concept. Intuitively, an object-oriented approach is not regarded to be the most suitable paradigm since the programming pattern tends to have a negative effect on the overall system performance, as covered in the research conducted by Zhang and his associates in [36]. Given that performance is considered as top priority, the idea is to provide the developer with an abstract layer that conceals the pitfalls presented by low-level programming languages and object-oriented structures. Thus, the issue is eradicated, masked away, by the initial designers and the ability to develop widgets is extended to a very wide audience.

The software demands, usability and user interaction, are subject matters that are becoming increasingly important in regards to interface design. While in the desktop world the developer can assume a consistent set of capabilities, applications targeted at the portable device market are obliged to take into account the vast variety of user interface and interaction styles that come with the native operating system. For example, whereas tabs are considered as a useful mechanism on a stylus based device, they are much less practical in soft-key based applications.
Thus, a specific solution for one platform might not mobilize and port as intended due to navigation and screen layout patterns.

Designing a high quality interface is often considered to be the most important aspect when creating applications. Regardless of how well the underlying functionality is programmed; if the graphical user interface is difficult to use, or prohibits the users from reaching their goal the program is bound to fail. In these circumstances the consumer is left with no choice but to turn to alternative services in search of more intuitive and efficient solutions.

In regards to the statements above, the main research objectives in this paper are as follows:

- Investigate different realizations of mobile widgets from a user experience point of view, specifically widgets hosted by dedicated runtime engines, or as Rich Internet Applications for web browsers.
- Study the effects of adopting a user-centered approach to widget user interface and interaction design.
- Analyze how two similar projects, conducted in different environments, can face dissimilar consequences in terms of development related problems and restrictions.
- Investigate the significance of non-functional project requirements, i.e. time constraints and the necessary experience, and their corresponding influence on the development.
- Briefly investigate possible techniques to account for the differences presented by the underlying platform.

1.2 Purpose

Given that developing a good graphical user interface is vital to the success of a product the purpose of this master’s thesis is to extend the second generation of web services, mainly widgets, onto portable devices; with focus on the interface. The application interface is to be designed and developed for different environments with usability in focus. As a result this thesis is set out to provide guidance, answers, or information on questions regarding fundamental user interface development. However, the previously stated problems, Section 1.1, constitute the core of the design patterns, which are accordingly refined and developed to meet the user demands. To conclude, simple proof-of-concept widget interfaces and several mockups, Section 4.1.5 are designed and developed for evaluation and demo purposes.

1.3 Delimitation

Developing a complete widget system for deployment on a mobile phone would be a complete success. However, it is out of the scope of this thesis as this may
require a significant amount of time. The main focus is the client-side, ergo the portable device, and the graphical user interface. As users hold the “don’t make me think” mentality very dear the intelligent borrowing concept was adopted to meet the demands. That is, the design was build on existing ideas to maintain consistency and reach higher levels of acceptance among the target group.

Some evident limitations in this thesis include:

- The gathered data were limited to internal observations though no real user testing was involved during any stage of the development.

- Based on the nature of the department the given feedback resulted in very similar response since the majority of the employees share the same academic background.

- The study did not control or include any core development which may have affected the interface quality.

- The investigation and development is restricted to native and browser environments, exclusive of alternative technologies such as Java Virtual Machine, Adobe Flash, etc. As the prototypes are developed for the Sony Ericsson P1 Smartphone the platforms consist of Symbian OS and the Opera web browser.
Chapter 2

Background

Widgets are generic, reusable, directly manipulated, self contained visual idioms [19]. The idea is that widgets are small pieces of software often well designed, visually appealing, and specifically engineered tools developed to do one thing and one thing only. Developed by adopting well established web technologies, such as HTML, JavaScript, and CSS, these encapsulated tools have a zero learning curve due to the simplistic nature of their design. Since widgets only require small amount of storage they can also be distributed with great ease and are accessible by anyone with a portable device and network access [21].

Enclosed in a browser runtime environment, these single purpose web-based applications are typically stripped clear of any browser related control. Given that a combination of browser chrome and the applications interface elements may generate confusion among the inexperienced users. The absence of browser chrome also presents additional advantages compared to browser-based web applications. From a design and development perspective the exclusion simplifies the preservation of consistency throughout the design pattern, and limits the user interaction to predefined and restricted regions. The widget can further extend and implement its own specific navigation scheme with custom menu items and, if required, control mapping.

Available widget engines loosen the strict outline of web applications running in the browser. They present the developer with functionalities such as client-side persistent storage, direct interfacing to the operating system or local applications, and execution of local or distributed JavaScript code from one or various sources. Cross-site scripting and mash-up of client functionality and data with web-based services are additional techniques that provide access to infinite new innovative service opportunities [7]. The results presented at the XTech conference [8], a mobile phonebook prototype conjoint with Yahoo People Search and Google Maps, demonstrate one of what otherwise is a broad spectrum of approaches to new extended services.

Widget platforms can be coarsely segmented into mobile, which are designed for devices where scarce resources dictate a dedicated approach and a concise delivery of information, and desktop widgets that tend to unify the remaining
Background

forms. The distinct categories share a common denominator in terms of providing identical services to their various relatives; that is single purpose applications that can be combined and are in fact fairly easy to develop, in contrast to a full featured application. Consequently, the common foundation and ideology logically implies similarities in presentation; however, the settle differences between these categories become more evident while discussing the way they are distributed and engaged.

Desktop widgets can be segmented further into application, operating system, and embedded widgets. The first mentioned is a standalone widget engine for both Mac and PC that can be downloaded, installed, and accessed like any other application. The best example of application widgets are Yahoo Widgets and Google Desktop. Operating system widgets are undoubtedly the evolution of application widgets and can be regarded as an integrated widget engine. On Apple Mac OS and Microsoft Windows Vista these are referred to as the Dashboard and the Windows Side bar, respectively. Embedded widgets, occasionally referred to as buttons, are small pieces of code that can be integrated into a pre-existing web page to provide the user with additional functionality.

2.1 Related work

Usability research on interface design, especially in regards to portable devices, is an emerging area of interest. However, research regarding design and usability testing is a process often mistaken as a straightforward technique that can be followed in a step-by-step manner. As discussed in [18], the process of interaction design is far from straightforward. Their study concludes that the challenges and the multitude of difficult decisions only become apparent when actually involved in a design project. It was also found that iterative design frequently involves carrying out different parts of a project in parallel and under tremendous pressure; seeing as the constant need to deal with various sets of demands and trade-offs, like testing versus time and resource limitation, acts as a major influence on the way design projects are carried out.

The aim of the research conducted in [7] and [8] is to convey what interaction design is like in the real world by describing how user interfaces for portable devices can be developed more easily with web technologies than with existing paradigms. Further, it was postulated how to deal with the challenges of an actual design project in regards to the development time, and the impact of introducing established technologies in a unfamiliar environment.

As discussed, there is a substantial history of research on the interaction approach and effectiveness of small screens. Designed and developed with regards to these aspects, research on Rich Internet Applications for portable devices has yielded a diverse variety of technical approaches. This relatively new kind of application has seen significant proliferation onto the desktop and more recently the web enabled portable device market. Although there is consistency across these different platforms in terms of available functionality, the way that the user interface is designed is inconsistent.

Available solutions such as Widsets, ZenZui, iPhone, etc., adopt vastly dif-
2.1 Related work

Segmented into two different interface categories, desktop and mobile, these interactive software products provide the consumer with identical services. However, indistinguishable to the user, these applications adapt vastly different approaches to interface and interaction design; note, as the exact technical detail regarding each widget engine is not disclosed, the associations made are strictly from a visual perspective.

In short, while there have been successful adaptations of widgets to portable platforms this thesis will integrate and significantly expand upon them from a usability perspective. The conducted work investigates un-explored areas of development platforms and interaction design by the means of visibility, feedback, constraints, mapping, consistency, and affordances, Section 3.3.
Chapter 3

Theory

To elevate the application to a usable product, the underlying complexities of the system rely on the user interface design to eradicate the barrier that is generally introduced by technological innovations. This concept highlights how the user experiences the final product and not the technological advancements as the key to user acceptance and subsequent marketability. Consequently, the most important motivation behind incorporating user interface design throughout the development cycle is the constant thrive for achieving greater acceptance among the target group.

Even if, optimizing the user interface requires a systematic approach to the design process, to guarantee optimum performance a thorough usability testing is essential. The participation of the target group provides vital data regarding what does and does not work as anticipated. Only after gathering the feedback and adjusting the design can the product be deemed to have reached a user optimized state.

Needless to say the importance of a well researched and tested user interface design can be the difference between product acceptance and rejection in the marketplace. If the user reactions are negative in terms of usability an otherwise excellent product could fail. By introducing the working relationship between usability specialists, focus on the interface, and product engineers, focus on the technology, can help make the product easy to understand and use; avoiding the otherwise common pitfalls [25].

This chapter covers the essentials behind user interface by examining the notion of interaction design. The differences between good and poor design proposals are examined by highlighting how visibility and consistency can tilt the balance towards more usable and intuitive interface patterns. In the last part of this chapter a general outline with core aspects of graphical user interfaces and how these are employed to assess interactive products is discussed. In Chapter 4 the various usability criteria are put into practice by evaluating mobile widget interfaces and constructing several low-fidelity models.
3.1 Design quality

As previously stated a central concern regarding interface design is to develop interactive products that are usable and not cumbersome. This logically implies that the final solution is easy to learn, effective to use, and provides a memorable user experience. The first step towards the goal of designing a usable interactive system is to include examples of well and poorly executed solutions in the equation. By thoroughly understanding and perpetuating useful patterns the specific strengths and weaknesses of different systems are effectively identified, and in either case preserved or avoided.

A system is regarded as problematic when a simple task is: infuriating, confusing, inefficient in terms of requiring the user to perform a number of unnecessary steps for basic tasks, difficult to use, and has no means of letting the user know what to do at a glance. As expected the required user-demands are directly proportional to the complexity of the system. Hence, regulating the offered functionality is crucial to maintain the accessibility. Also by capitalizing on the target audience everyday knowledge and how they interact with similar systems results in approachable and aesthetically pleasing products designed with the users in mind.

The focus is on providing the user with an enjoyable experience whilst maintaining the demands as low as possible through the adoption of ubiquitous everyday routines and experiences. However, it is important to note that one interface design that is very elegant and usable in one situation might not transition well to new yet similar settings. For example, while text input through a desktop keyboard is regarded as a fairly easy task the same cannot be said about portable devices since the keypad size limits its usability. Therefore, it is necessary to reflect on where it is going to be used, and who is going to use it when considering the quality and usability of a design [18].

3.2 Design process

Designing usable and enjoyable interactive user interfaces, thus, entails considering who is going to be using the final product and where they are going to be used. This eventually leads to another central concern; understanding the user activities when interacting with the system. The appropriateness of various user interface designs and placement of input and output interaction mechanisms are directly dependent on what kinds of activities that needs to be supported. As the range of activities that can be supported is diverse, a user interface design that allows readily access to frequently used functionalities is most appropriate.

To optimize the user interactions with the system by matching functionality and the user activities, design decisions are obliged to follow a set of principles, Section 3.3 based on the understanding of the target group; as an alternative to using intuition and hope for the best. This concept involves identifying user needs through understanding the subsequent aspects [18].

- Taking into account what people are good and bad at.
3.2 Design process

- Considering what might help users with the way they currently interact with similar solutions.
- Thinking through what might increase the overall user experience.
- Involving the user throughout the development process.
- Utilizing the “tried and tested” concept during the design phase.

This technique contrasts with software engineers’ primary focus, production of suitable software solutions for the application. However, the integration of these aspects draws the line to finding ways that supports the user in achieving their goal.

3.2.1 Interaction activities

At its core, the process of interaction design covers four basic activities [18].

- Understanding and conceptualizing the problem space.
- Identifying needs and establishing requirements.
- Developing alternative designs that meet those needs and requirements.
- Building and evaluating interactive versions of the design for assessment and testing.

These activities are intended to function as a process model and are deliberately designed to inform one another and to be repeated. During this process several design iterations are fashioned, all of which have undergone specific changes based on the provided feedback from each interface proposal. Measuring the usability of an interface design, in terms of how intuitive and easy the product is to use, provides the necessary data regarding the changes that must be made, or the requirements that have not yet been met. This concept, often referred to as user-centered approach to design, seeks to involve the target group throughout the design process. The focus is on ensuring that the final product is usable and provides the user with a memorable experience.

Since evaluation is the heart of interaction design, there are several different methods of achieving this; through user observation, feedback, evaluating the performance against the identified needs and requirements, questionnaires, etc. The results from these findings are equally important and must be interpreted with respect to the users’ knowledge of the underlying system and the design activities [18].

3.2.2 The design goal

Essentially, by capturing and understanding the users’ needs the interface design will lead to a very efficient environment that allows the consumer to be highly productive in their work. To attain this level of usability the products primary
objective must be clear. This top level concern allows the designers to discover how the goal can be met and through what means, i.e. how they are operationalized.

The usability goals are well suited for introducing or updating applications running on mobile, desktop or networked systems; with the intention of increasing overall productivity through enhancing and improving how different tasks are done. Since the usability goals cover a decisive portion of interaction design the requisites are couched, in general terms, and specific questions are formulated by turning the goal into usability criteria. The primary usability criteria that prevail in the widget interface context are, efficiency, referring to how forgiving and supportive the interface is in terms of providing user-assistance when carrying out specific tasks; learnability, which refers to how easy and intuitive the product is to use, and finally memorability, denoting how easy it is to remember and use the system once the peak of the learning curve is reached. This form of objectives enables the usability of a system to be assessed and provides the means of measuring the user performance [18].

3.3 Design principles

In previous sections usability was conceptualized through the notion of usability goals and user experience goals. Another approach is the adaptation of generalizable abstractions; intended to inspire and orient the initial designers towards reflecting about various aspects of their design. The latter technique is referred to as design principles and is in essence a direct derivative from a mixture of theory based knowledge, common sense, and experiences. These principles are regarded as an outline equipping designers with the means and rules that suggest the dos and don’ts of interaction design; if you prefer, they are intended to help designers explain and improve the graphical presentation [21]. A common misconception, however, is the different interpretations of the principles; they are to act as a set of reminders ensuring the success of imperative elements, and not as a specification to the actual user interface design. Hence, the model is to be regarded as a checklist rather than a step-by-step tutorial on how to design particular menus, icons, etc.

Depending on the field of interest, design principles are exercised and adopted in various yet vaguely different conventions. The most relative to the subject concerns the method of how to determine what users should see and do when carrying out the desired task while using the system. The subsequent sections will briefly cover this concept, since the subject is covered exhaustively in [15] and [18], by the means of visibility, feedback, constraints, mapping, consistency, and affordances.

3.3.1 Visibility

The importance of visibility is undeniable; the concepts effect, consequently, renders as the most crucial aspect of graphical user interface design. Poor design and misplacement of frequently used functionalities conceals their presence and disrupts the conceptual image perceived by the user. Visibility assists the user and
3.3 Design principles

provides enough data to create an accurate mental model of the underlying system. This helps the user in predicting the effect and the corresponding causality of their actions.

The more visible functions are, the more likely users will be able to know what to do next [18]. Moreover when functions, such as those that aid navigation, are hidden or have been positioned incorrectly, in regards to its operation, spawns finding the appropriate control as a very perplex task; if you prefer, essential elements that are “out of sight” are more complicated to find and know how to use. Hence, the controls for different operations should be clearly accessible at a glance, indicating what can and cannot be done.

3.3.2 Feedback

Strictly related to the previously covered concept of visibility, the notion of feedback is best illustrated by means of everyday analogies and how they are perceived. Imagine trying to write a note using a pencil if none of your strokes produced any effect for several seconds. This would create an unbearable delay and register tremendous confusion while attempting to continue with the next stroke; in reality a simple task such as writing is almost impossible and perplexing for any person if the feedback is not instantaneous. Thus, feedback is classified as user confirmation regarding what action was performed and what has been accomplished [18]. This allows the user to continue with the intended activity and eventually reach the initial goal.

Actual feedback is presented in various forms in interaction design: visual, aural, audio, verbal, tactile stimulation or as a combination of these. Deciding which type or combinations are appropriate for specific activities and interactions is vital. Further, matching the system and the real world, “speaking the users’ language” [18], is essential to avoid confusion or misinterpretation. Adopting feedback is, however, not obvious and must be used in the right way to accommodate and provide the user with the necessary visibility for interaction.

3.3.3 Constraints

The design concept, constraints, refers to restricting user interaction mechanisms to only actions permissible at that stage of the activity. There are numerous ways this method can be achieved. Most commonly in graphical user interface designs the initial developer deactivates irrelevant options by temporarily shading or disabling the menu; as a result confining and preventing the user from selecting incorrect options. The notion, constraining the user to activity specific options, also renders the system as less error-prone. Introducing several layers of different graphical representations and information space can too constrain the users’ interpretation. For example, consider a flow chart diagram which displays the relations between different objects and thereby restricting the manner of which information is perceived [18].

Constraints can be classified into three kinds of behavioral categories: physical, logical, and cultural [14]. Physical constraints are closely related to real af-
fordances and refer to how physical objects restrict the movement of things based on correlation. For example, restricting the cursor visibility to predefined regions of the screen is a physical constraint.

Logical constraints depend on people’s conception of the way reality works, reasoning, and common sense to determine the alternatives. It is how the user knows to scroll down to see the rest of the page content. Making actions, the fundamental design model, and their corresponding effects visible enables users to readily deduce what actions are required.

Cultural constraints are learned conventions shared by a cultural group, like red traffic lights denote stop and a smiley represent the human emotion of happiness; are both universally accepted principles. These constraints are considered as arbitrary in the sense that there is nothing inherent in the device or design that requires the system to function in that particular way; if you prefer, since the relationship with what is being represented is strictly abstract the green traffic light instead of red could have evolved to symbolize stop.

### 3.3.4 Mapping

Mapping refers to the established relationship between controls and their corresponding effects in the system. A large portion of all the interaction mechanisms need some kind of reference that denotes the association between controls and their corresponding effects; independent of the underlying system. The up and down arrows on a computer keyboard is one example where the concept of mapping has been utilized. The mapping between the arrows control and effect are used to represents the up and down movement, respectively, of the cursor.

Aside from control and effect associations through mapping, the relative position of controls in regards to each other is also important. Consider the controls play, rewind, and fast-forward on any musical playing device, like CD, MP3 player, etc. They generally follow a common design pattern in terms of conventions and placement; the button layout provides a universally accepted sequence with the play button placed in the middle, the fast-forward on the right, and the rewind on the left. Consequently, the button configuration maps directly onto the directionality of the available actions with the intention of preventing any design pattern related problems [18].

### 3.3.5 Consistency

The principle, consistency, refers to preserving similar operations and the use of similar input or output mechanisms for achieving related tasks; if you prefer, the user interface follows a predefined set of rules identifying similar operations by using the same action throughout the system. For example, the user interface design is regarded as consistent when the same input mechanism is used to highlight any object. Inconsistent user interfaces, on the other hand, allow exceptions to that rule; making user interaction difficult and more prone to mistakes since the inconsistencies are relatively arbitrary [18].
3.3 Design principles

The main benefit of consistent interfaces is that they are easier to learn and use since one single mode of operation is applicable throughout the system. The concept adapts well in reasonably simple interfaces with limited functionality where all operations are mapped onto separate buttons that are constantly visible. However, consistent design is more problematic as the complexity of the interface grows, in particular when the number of functionalities is remarkably increased. As the available display area is limited, there is literally not enough space for hundred or even thousand buttons mapped to individual operations; even if there were, the lack of structure renders finding the desired operation extremely difficult and time consuming. This requires the introduction of categories of commands that are mapped into subsets of operations; like all the file concerned operations, e.g. new, save, open, close, etc., are placed together under the same file menu. Still, this solution effectively breaks the one-to-one mapping between command and operation as operations are not immediately visible forcing the user to learn finding operations based on category and relation.

Determining what aspects of the user interface to make consistent in regards to what else is another problem. Generally the choices are widely spread and some can be regarded as inconsistent relative to other aspects of the interface design or the way tasks are performed. As a designer there are two approaches, external consistency referring to the way users address the task in the real world and internal consistency by adopting actions introduced through existing solutions. The problem facing designers is, knowing which method to adopt and be consistent with [18].

3.3.6 Affordance

The term, affordance, is used to refer to the actionable properties between the world and a user; if you prefer, an attribute of an object that provides critical clues required for it to function properly. At a very primitive and basic level: to afford means “to give a clue” [15]. The sole purpose is to allow users to know how to use the object even if this is their first encounter. Consequently, when the affordance of an object is perceptually obvious the required user task is regarded as more intuitive and logical. For example, the keyboard buttons affords pushing, a door handle affords pulling, etc.

The term has since first coined by the perceptual psychologist J. J. Gibson [6], regrettably, become a general container for all phrases alike; thus, losing its original potency as a design principle due to the way it has come to be exploited in everyday parlance. To clarify the notion and its relevance in graphical user interfaces the concept has to be categorized into two different entities perceived and real affordance. The latter refers to interaction with physical objects that are considered obvious and does not have to be previously learned. Graphical user interfaces are, on the other hand, conceptualized as perceived affordance which is in essence learned conventions. In graphical terms the icons, buttons, and scroll bars are designed with respect to how to make the usability more obvious at a glance; scroll bars should be designed to afford moving up and down, icons to afford clicking, etc. [18].
3.4 Design construction

Once the user requirements have been established the next advancement in product development is designing activities; in general terms, these are referred to as conceptual and physical design types. The latter is concerned with actual graphical elements and details of design such as menu structures, graphics, icons, etc., while the former is concerned with developing a conceptual model that captures the products behavior patterns; if you prefer, what the product should do and how it can be achieved [18]. As covered in previous sections, the final design proposal is not achieved through a simple draw and construct technique, but rather as an iterative routine through repeated design-evaluation-redesign cycles involving the usability specialist and the target group.

Since the user is deemed as the only valuable source to effective design evaluation of interactive products, designers are obliged to produce an interactive version of their proposals. Early in the development phase these interactive models are made of mockup drawings on paper and cardboards; however, as the design iterations and ideas become more detailed simple software or hardware models, which function and resemble the final product, are constructed. The activity concerned with building interactive versions for evaluation is referred to as prototyping [18] and is covered in greater detail in Section 3.5.

“It is often said that users can’t tell you what they want, but when they see something and get to use it, they soon know what they don’t want” [18].

The main purpose of this concept is to collect information regarding work practices and different views on what the system should and should not do, in terms of functionality. The next logical step is to deploy this knowledge by building prototypes and iterating through several versions; with the intention of designing an enjoyable user environment and reach greater acceptance among the target group.

3.5 Design prototyping

Interface prototyping is a widely employed and proven technique by software designers for evaluating design alternatives and for obtaining general feedback in a cost effective way. The process of creating a model before development is regarded as common discipline among engineers. This allows the users to be involved at a very early stage in development where modifications are still inexpensive and feasible; as changes are time consuming, impractical, and extremely expensive after the implementation [4]. Prototyping permits the usability designers to meet the needs and desires of the target group at the beginning of the development phase.

Regardless of knowledge and experience the usability specialists are often considered as the worst judges of the user interface they design; hence the expression “no author can view his or her own work with dispassion”. The target audience is regarded as the only viable source who can adequately judge the accessibility and effectiveness of the interface. User interface prototypes allow the software development team to investigate and analyze the organization of the design pattern, layout, its wording, overall appearance, appropriateness, menus, buttons, color
3.5 Design prototyping

choices, and fonts. Including concrete scaled down prototypes, more on that in a moment, in the curriculum spawns the user as the focal point when interface design decisions are made. The benefit from this process is prominent products that are aesthetically appealing, convenient, and allow the user to accomplish their goals. Further, usable software is generally much more apt to succeed in the targeted market place \[4\].

As previously stated prototyping allows for iterative modifications to the interface design prior to the final approval. This technique also serves as a mean to receive the users “sign off” on the recommended design proposal. To achieve this, two variations of discovery prototypes are developed, these are covered in greater detail in Sections 3.5.1 and 3.5.2. The first model is a paper mockup, in traditional methodologies commonly referred to as layout chart; an effective concept that provides early feedback by refining the design and workflow through various user scenarios. The second model of discovery prototype is an electronic mockup developed with no underlying action attribute. Thus, the prototypes lack the necessary working functionality and algorithms, ergo non-operational. However, the interface output and graphical elements are in the second phase prototype populated with pretend data, hardcoded in the program to provide a more realistic experience \[35\]. At each stage, the prototypes are minimally specified, deliberately leaving some aspects vague so as to stimulate further ideas and discussion.

3.5.1 Low-fidelity prototyping

Low-fidelity model refers to prototypes that do not necessarily look like the final product in terms of casing. For example, it may use materials that are very different from the intended final version, like cardboard and paper instead of electronic equipment and metal. These types of prototypes are primarily useful because they tend to be very simplistic in nature and therefore cheap and quick to produce. This also implies that they are very modifiable, to encourage and support the exploration of alternative interface designs and ideas. The latter is a very important attribute of low-fidelity models since prototypes in early stages of conceptual design should be flexible and encourage rather than discourage discovery and modification. Consequently, the proposals are never intended for integration in the final product but rather as a means of exploration \[18\].

When designing a graphical user interface, usability specialists greatly rely on one of the three methods of conventional low-fidelity prototyping, referred to as storyboarding, sketching and the wizard of oz technique \[18\]. Much of the discussion about the model to use depends on the task and the purpose of the prototype. Storyboarding is a method often adopted in conjunction with scenarios; that consists of several sketches illustrating how a user might progress through different tasks using the available tools in the system. This technique can accommodate graphical user interface based software development as well as provide fairly detailed feedback regarding the necessary steps required to achieve a specific goal.

Regrettably one of the most useful but least adapted techniques is sketching; as many consider this activity difficult to engage and are discouraged because they are inhibited about the quality of their work. This is a common misconception, given
that the method does not emphasize the quality of the drawings but rather the intended message. The required graphical elements can be very straightforward and do not have to be anything more than simple boxes, circles, and stick figures.

Wizard of oz refers to simple software-based prototypes that emulate the underlying system through the involvement of a human operator. The user interacts with the software through a computer as though interacting with the product. However, the user interactions are registered and forwarded to another machine where a human sits and simulates the appropriate response to the user actions. Logically, this method is fairly intricate and is not regarded as the most appropriate method for harvesting feedback [20].

3.5.2 High-fidelity prototyping

High-fidelity model refers to prototypes that do look very close to the final product and have complete functionality; providing a full interactive experience. For example, a software-based prototype developed with available graphical components in Adobe Flash is higher fidelity of a paper-based mockup; a molded piece of plastic with a “dummy” keyboard is a higher fidelity prototype than a cardboard-based box with non-operational input mechanisms [13]. Accordingly, whereas low-fidelity scenarios address the layout and terminologies of the application, also called surface presentation, high-fidelity models address the issues of navigation and workflow; matching the design and user models of the system.

These types of prototypes are useful because they are interactive and the user can operate the model as if it were the final product. The user can navigate through the system selecting icons on the screen and expect functions to be launched or windows to open; according to their actions. Data can be displayed in real-time fashion or updated periodically and feedback can be delivered to the user at appropriate times. In general terms, the user can get a feel for how the product will operate and, therefore, can make informed recommendations about how to improve its graphical user interface [20].

Representative prototypes can be available for public testing months before the product code, allowing test case construction, help panel design, usability testing, and documentation to be initiated much earlier in the development cycle. High-fidelity prototypes also provide great marketing and sales tool. This presents the usability specialists with the opportunity of harvesting solicited customer input via community sites and trade shows. The generated feedback is then used to refine the prototype according to the user experiences.

High-fidelity prototypes can, also, be used very effectively to encourage customer buy-in; the prototype is more or less fully functional in comparison to a low-fidelity prototype and thus provides a better foundation for thorough evaluation by the target group. This rapid turnaround fosters greater acceptance among the target audience, as users can immediately see their design recommendations put into place. The target group becomes constructive, contributing members, rather than “evaluators” of the design team when they see immediate results based on their input [20].
3.5 Design prototyping

3.5.3 Compromises in prototyping
As expected the very nature of prototypes introduce several compromises; since the purpose is to develop something fairly quickly to test a specific aspect of the product. The design must be constructed with the key issues in mind to provide the kind of questions or choices that the designers can answer. Hence, the type of feedback that any prototype generates is regarded as very limited and specific to the task in mind. For example, with a paper mockup the obvious compromise is that the device is non-operational; if you prefer, it does not actually work [18]. For electronic mockup or software-based prototypes the restrictions may still be relatively clear like the response time may be slow, or the graphical elements may be sketchy, or only a limited amount of functionality is supported.

Divided into two common compromises, the breath of the functionality is often traded against the depth. These are regarded as two different types of prototyping techniques and are referred to as horizontal prototyping and vertical prototyping, respectively. The former provides a wide range of functionality but limited detail while the latter provides complete detail for only a few functions.

Not all compromises are obvious to the users of the system. For example, the underlying and internal structure of the system may not have been designed as carefully and the prototype may contain redundant and badly partitioned code. These compromises introduce the danger of users misinterpreting the prototype and believing that the mockup is the actual system, thus creating a false mental image. However, the less obvious deficiencies generated by the adaption of prototypes must be ignored, since the intention is to harvest knowledge and feedback that may improve the usability [18].
Chapter 4

Design

Portable devices are subject to several predictions about the booming future of internet related technologies and the potential of promising third generation of mobile applications, such as location based services, mobile commerce, and multimedia entertainment. Still, if the human-computer interaction aspect of mobile applications are not appropriately addressed these predictions are not so likely to come true. The consumer will not adopt portable technologies if the complexity of limited interaction capabilities is not masked away through a well designed interface [2]. Services targeted at the portable device market will not be successful if the context is not designed and tailored for the mobile audience which is different from the traditional desktop users.

Chapter 3 covered the notion of interaction design and how it has evolved by coarsely examining the various processes that constitute the core of conceptual design ideologies. The concept was divided according to usability and its profound affect was explained through how it is operationalized to assess the appropriateness, effectiveness, and quality of highly interactive interfaces. Different design principles were introduced to provide guidance on how to achieve greater acceptance among the target group and how to assign a user-centered approach to the actual design.

In this chapter, a variety of widget dashboard interface designs will be presented and evaluated. The low-fidelity design prototypes included in this thesis are the result of the process and constraints discussed in the subsequent sections. They provide a clear picture of several major issues and how these are addressed. In particular, the impact and the central role users have in interaction design. Concerns regarding input techniques, severely limited peripherals, and limited resolutions are issues which the design prototypes are specifically devoted to address.

Each design should undergo rigorous testing and evaluation to investigate its appropriateness, as well as usability analysis to examine the interaction design [3]; however given that the emphasis of this thesis is the development platform, each prototype is only coarsely covered and discussed. The goal is to design an interactive interface that reduces the mental and physical stress, the learning curve, and improves user device operability.
4.1 The design process

Conceptual design, introduced in Chapter 3, is concerned with harvesting information through user requirements and needs. The gathered intelligence represents the design foundation which is transformed into a conceptual model. Conceptual model covered in Section 3.2 was only introduced theoretically; in contrast, the following sections provide the necessary details and discuss how this technique is deployed. To recapture, the model was defined as: “a description of the proposed system in terms of a set of integrated ideas and concepts about what it should do, behave, and look like, that will be understandable by the users in the manner intended” [18].

As stated, the basis for this model is the various users tasks or functionalities the final product will support. Regrettably, there exists no straightforward transformation technique to apply to a set of requirements data that will produce “the best” or even a “respectable” conceptual model. The only available method is stepping yourself in the data while reflecting and trying to empathize with the target group; as an abstract picture of how the users experience is intended to be when using the product will gradually emerge.

4.1.1 Identifying constraints

The user interface is the part of software that is used to convey information to the user, or take instructions from the user. Hence it consists of two distinct components, frequently referred to as an input and output language. The user utilizes the input language to communicate with the application by manipulating interaction mechanisms like keyboard, stylus, etc. The portable device uses the output language, graphical user interface, to display its current state and occasionally give the user various notifications.

Since the graphical user interface is targeted at a diverse target group that often lack the experience, training, or reference to a manual to utilize the application; a highly intuitive interaction design is critical. Broadening the target group with users who have no previous computer experience renders the effectiveness of the interface design as even more crucial. Despite that a number of conferences and workshops [17] have examined the problems of portable device interaction there still remains concerns in terms of lack of transformation techniques. Key problems in interface design for portable devices are the development of appropriate visual presentation and the implementation of suitable interaction mechanisms; since these differ dramatically from the desktop domain [17].

Interaction design for portable devices, in particular mobile phones, is subjected to various constraints compared to relative desktop applications in terms of scarce resources. Evident constraints [17] that have a profound impact on the interface design include:

Limited resolution ; constitutes the key problem that must be appropriately addressed by the interface designer through smart placement and optimal use of the available display resolution; ranging from 240x320 feature-phones to 800x480 smart-phones and personal digital assistants.
4.1 The design process

Small display size: yet another central constraint which introduces concerns for users with navigation and/or possible vision problems though applications are aimed at a wide target group.

Scarce hardware resources: limited processing power severely affects the use of complex graphical solutions, distortion based layouts, and eventually the response time.

Limited interaction mechanisms: while in the desktop world the developer can assume a consistent set of capabilities, applications targeted at the portable device market are obliged to take into account the vast variety of interaction styles, often device specific, that come with the product.

Device specific interaction techniques: the lack of standardization creates confusion and the user cannot rely on the presence of input mechanisms making skill transfer between different interfaces difficult.

Level of attention: during certain timeframes the attention that a user can devote might be limited or interaction may be interrupted by external events.

To conclude, the design of usable interfaces for portable devices is critical for the application success; however, current interaction experience is limited and activities are complicated. The lack of interaction standardization for portable devices and the absence of hardware resources produce special requirements that should be addressed by usability specialists and product engineers [17]. As a first step in this direction the following sections represent a repository of graphical user interface design solutions that enable effective presentation of the content under the above mentioned constraints of portable devices.

4.1.2 Identifying needs and establishing requirements

When designing a product to support people, the target group must be identified and the form of interactive support needs to be investigated. These needs are essential and form the foundation of the product requirements; this, in due course, underpins subsequent design and development decisions. If you prefer, the activity is fundamental to the product development and to a user-centered approach; making it a very important factor in interaction design.

The design project may aim to update or replace an existing and established system, or it may aim to develop a new and innovative product experience with no distinct precedent. Depending on the initial situation there may or may not be a predetermined set of requirements. For example, in the case of new and unexplored products the requirements need to be produced from scratch whereas the requirements can be borrowed, refined and reapplied in situations where similar services are available. Regardless of the situation and primary aim of the project the users’ needs, requirements, expectations, and aspirations have to be discussed, clarified, and probably re-scope to some extent. This naturally requires a thorough understanding of the target group and their capabilities, the user task and
goals, the conditions under which the product will be used, and the constraints presented by the product’s hardware [18].

As covered in Section 4.1.1, the constraints of portable devices are quite a few and are mainly hardware related. Identifying user needs, however, is not always as straightforward and frequently depends on more than one factor. Establishing requirements is also a very time-consuming and difficult task; not to be confused with writing a simple wish list of desired features. Isolating requirement activities from design activities and from evaluation activities can be regarded as a little artificial; given the iterative nature of interaction design. In practice these activities are very well intertwined meaning some design decisions will take place while requirements are still being established; and the design will literally evolve through a series of design-evaluation-redesign cycles [18]. However, each of these activities can be distinguished by its own emphasis and their individual bearing, of each corresponding technique, on the final product.

The issue of content overflow and the significance of overview were discussed and issued as the most significant requirement in the introduction of this thesis. In software engineering terminology, there are two different types of requirements. These are traditionally captured as functional requirements which refer to what the system should do, and non-functional requirements that states the constraints there are on the actual system and its development. The issue of overview falls under the former category and can be further decomposed into more specific requirements detailing the underlying necessities vital in achieving this goal. The focal point of widgets is to provide the user with instant visibility and easy access to the desired operations; these requirements act as a subcategory to the overview problem presented by current generation of applications. Another important functional requirement is the option of personalizing content to the level of desire; this implies that the user should control the content and have the choice of selecting only the preferred information.

Non-functional requirements are hardware restrictions presented by portable devices, like battery life, limited resolutions, etc. Moreover, interaction mechanisms touch or soft-key based, and the time constraints introduced by the thesis deadline are other attributes that falls under this category. System migration is a subcategory to the former mentioned constraint which specifically denotes the portability of the graphical user interface from one environment to another. For example, how well the interaction design transitions from stylus to soft-key based devices without sacrificing accessibility. These represent a constraint on the development activity itself rather than on the product being developed [18].

4.1.3 Developing alternative designs

This consecutive development step is considered as the core activity of designing interactive products. The activity involves suggesting concrete ideas for meeting the prior established functional and non-functional requirements, Section 4.1.1. As with requirements this activity is also categorized into two sub-activities referred to as conceptual and physical design, Section 3.4. The former concept involves constructing a conceptual model for the product; if you prefer, designing a model
that describes what the product should do, behave, and look like. The latter considers the actual detail of the final product including the sounds, colors, images, menu, and icon design [18].

Ideas for various conceptual models may surface gradually during data gathering; however, there must be a clear separation between the real requirements and the solutions ideas. Considering alternative interpretations and repeatedly designing with regards to different perspectives helps to expand the solution space. Covered in Chapter 3, prototyping and scenarios are two important techniques to help explore ideas, prompt insights, and make design decisions. Hence, before explaining how these can help the design process the notion of how to go about envisioning the intended product is necessary.

Developing a conceptual model, as stated above, involves envisioning the final product in conjunction with the users’ needs and other identified attributes such as requirements. To ensure product success and reach the desired acceptance among the target group the design must be improved through iterative testing as it is developed. A key aspect of this concept is initially to decide what the users will be doing when carrying out their tasks [18]. For example, will the users be primarily creating documents, communicating with other users, searching for information, or maybe some other activity. Reverting to the widget interface, the user activity is strictly limited to searching for information. Thus, the design is focused on eradicating the need to retrieve data manually; introducing the user to an environment where they can easily gain access to information from different services.

At this stage in development the interaction mode most suited to achieve and support this activity needs to be considered. For example what would be considered more appropriate: allowing the users to ask questions directly to the system in their native tongue, or would allowing the user to navigate and browse be regarded as more effective. Regardless of interaction mode the decisions about which type of interaction style, e.g. menu-based, speech input commands, etc., to adopt must be made in relation to each other. Consequently, decisions regarding the mode of interaction differ, quite dramatically, from what interaction styles that needs to be supported; the former being more abstract and at a higher level while the latter is concerned with attributes more specific to the kind of graphical user interface.

Seeing as the focal point of the widget interface by definition is to provide “glanceability” in terms of general overview, and “accessibility” through consistency and simplistic design, a harmonious model must be established through integration of several styles; if the design is to be understandable in the manner intended. Already established systems, covered in Chapter 2, rely on symmetric and asymmetric presentation constellations; arranged according to a two-dimensional grid pattern layout. A first thing to note is that the various applications are mutually exclusive, as they are designed to accomplish their individual tasks asynchronously. However, the manageability and distinct segregation is achieved on the expense of usability since accessing menus is not as intuitive as deemed necessary. Frequently accessed functionalities are concealed due to inaccurate prioritization of activities, literally eliminating any usability gained by adopting one-to-one mapping techniques.
Once the patterns of interacting with the system have been clearly identified, the design of the conceptual model then needs to be carefully thought through in terms of actual concrete solutions. This implies working out the behavior attributes of the user interface, the particular interaction styles that are to be adapted, and finally the “look and feel” of the graphical output [18]. At this stage the simplest way to analyze the ideas is to construct several possible designs and assess the merits and problems of each solution in comparison to each other, covered in the subsequent sections.

4.1.4 Building interactive versions of the designs

Emphasized throughout this thesis is the notion of iterative interaction design. As covered in Chapter 3 the concept involves cycling through various design processes, each effectively contemplated at different levels of detail. The process cycle involves a series of steps that are deliberately designed to inform one another and to be repeated. As proposed by [18] this primarily involves: “thinking thorough a design problem, understanding the users’ needs, coming up with possible conceptual models, prototyping them, evaluating them with respect to usability and user experience goals, thinking about the design implications of the evaluation studies, making change to the prototypes with respect to these, evaluating the changed prototypes, thinking through whether the change have improved the interface and interaction, and so on”. Simply put, each process iteration involves processing through the design pattern in more depth.

As introduced in Section 3.5 several user-centered techniques can be utilized to construct the prototypes of the potential candidates. Since interaction design involves designing interactive products; logically, the most sensible way to evaluate interface designs is to actively interact with them. To allow this process, interactive versions of the design needs to be constructed; however, this does not necessarily imply that the underlying software is required. There are numerous techniques for achieving non-operational interaction models, covered in more detail in Sections 3.5.1 and 3.5.2. Based on the presented notions in this and the previous chapter, a number of design solutions are presented in the subsequent sections each addressing the discussed limitations, requirements, and needs in different ways.

These interface prototypes are designed not to conflict with the users cognitive process involved in achieving the desired task. That is the graphical user interfaces are designed with human characteristics, such as attention, perception, and memory in mind; to help circumvent and decrease the demands introduced by the system. To help ensure that the prototypes are representative in the manner intended Shneiderman’s eight golden rules, covered coarsely below, of interaction design are adapted to inform of any abnormalities or misconceptions that needs refinement [22]. All of these guidelines focus on making the communication between the system and the user as clear as possible [18].

1. Strive for consistency. For example, in every screen have a back button positioned at a fixed position throughout the system, or actions that results in the loss of data should ask for user confirmation to provide the users a chance to change their minds.
2. Enable frequent users to use shortcuts. Provide the user with alternative short commands to frequently used functionalities, or present these as simple one-to-one mapped elements that are easily accessible.

3. Offer informative feedback. Avoid incomprehensible feedback and make it clear what the message actually means. For example, instead of displaying “Error 404”, provide the user with meaningful feedback to aid or avoid similar mistakes.

4. Design dialogs to yield closure. Present the user with visual confirmation upon successful completion of the performed action.

5. Offer error prevention and simple error handling. Design the interface with the intention of assisting users and preventing them from making any mistakes. However, as mistakes are inevitable provide support and develop the underlying system to be forgiving.

6. Permit easy reversal of actions. Provide the user with the option to undo where possible.

7. Support internal locus of control. Users feel more comfortable if they feel in control of the interaction rather than the device being in control.

8. Reduce short-term memory load. Wherever possible offer users options rather than asking them to remember specific information.

### 4.1.5 Constructed prototypes

As important as thoroughly understanding the different cognitive process the users engage in when interacting with the graphical user interface, it is also very useful to understand the way users cope with similar solutions and their corresponding demands. Initially covered in Section 4.1.1, the design project aims at updating existing solutions in the same field by adding more value in terms of usability and visibility. A well established method for borrowing, refining, and reapplying knowledge from one service to another is to **emulate**, in the digital sense of the word, the strategies and techniques that delivered a memorable and enjoyable user experience [18].

Prototype one and two, illustrated in Figure 4.1a and 4.1b respectively, draw inspiration from existing solutions where widgets are arranged symmetrically according to a two-dimensional grid pattern. The former, accents more detailed view over a few widgets and provides a three step zoom functionality that allows the user to have a maximum of six applications visible at one time. The latter, however, emphasizes the importance of overview and introduces the notion of “groups”, demonstrated with the shaded boxes in Figure 4.1b. Groups are to be regarded as containers encapsulating up to four separate widgets under one category; this provides the user with the option of personalization and allows the user to expand a single group to full screen mode. Prototype two is in fact the final iteration of prototype one as it capitalizes on the group and one-to-one mapping
phenomenon to increase usability and maintain broad-spectrum visibility; thus, addressing some of the problems introduced by the first design proposal. These prototypes are categorized as soft-key based and are both intended to be designed for portable devices that lack the touch input mechanism.

Prototype three and four, illustrated in Figure 4.2a and 4.2b respectively, exploit the “sheet model” introduced by Apple in their iPhone and iTouch products; as an intuitive interaction technique to browse through music by album art. This implies that these designs are mainly suited for portable devices adopting touch input mechanisms; seeing as in most situations soft-key based interactions can turn out to be counter-productive, forcing users to do things in inefficient, bizarre, or inappropriate ways. The first iteration of this prototype neglects the overview aspect that is otherwise regarded as crucial; however, this was partially addressed in the second installation which provides an independent strip at the bottom of the screen to present a fairly limited overview of available widgets.

The previously discussed designs applied the knowledge of users, coping strategies while interacting with established services; effectively borrowing, refining, and reapplying concepts that are well received in the market. Another approach is to apply theories and conceptual frameworks to interaction design that adds a different perspective on cognition. This approach conceptualizes how the mind works and introduces real life metaphors and analogies. For example, the human mind can be considered as an information processor, a prevalent metaphor from cognitive psychology, because information is thought to enter and exit the mind through a series of ordered processing sequences or stages. Each stage consists of various processes that are presumably connected and act upon the perceived mental representations, like comparing, matching, and are assumed to comprise rules, images, and other experiences. If you prefer, mental models that have their origins in similarities are developed to create a passage to aspects of a physical entity; however, the entities are still entitled to their own set of properties and behavior patterns.
4.1 The design process

Models based on this concept can be constructed with regards to an activity or an object or both.

The final two designs, illustrated in Figure 4.3a and 4.3b, are categorized as conceptual models based on objects and interface metaphors. Despite the commonly used metaphor, current virtual desktops bear little resemblance to the look and feel of real world desktops. Prototype five has been designed to encourage and invite rational comparison with everyday analogies and how they are perceived. Note that the similarities alluded to by the adaptation of the term “desktop” are at a very general and high conceptual level. Consequently, the actions introduced by the desktop metaphor are performed in a slightly different way from how a human being might interact with their real desk. Widgets are represented as paper sheets that can either be placed on the desk or attached to the walls and their size and color are two aspects that convey information regarding relation and importance. Motion is constrained to the desktop by the walls enclosing the surrounding environment. In addition, the wall corners provide a place for widgets to pile up on top of each other and act as a landscape feature that could aid in cognitive grouping of applications or documents. These everyday analogies are meant to conjure up the essence of the process of working at your desk, enabling the user to leverage off this anchor to gain further understanding of other aspects of the interface.

Prototype six, Figure 4.3c, explores the notion of analogies further by introducing an additional level of graphical abstraction. This introduces a design that draws inspiration from some of the valuable characteristics of the universe. The interface metaphor is based on conceptual models that combine general knowledge with new concepts to produce alternative designs to the ubiquitous galaxy paradigm. Instead of the rigid organizational structures imposed by current graphical user interfaces, prototype six introduces a more casual and intuitive planetary constellation premises. This model provides the user with a complete system...
overview as each widget is represented as a star where their size and colors denote significance and relation. As mentioned this prototype is based on a conceptual model of the familiar knowledge of the universe and solar systems; hence the galaxy representation is designed to possess some of the properties of their physical counterparts.

A general and less evident misconception, however, is the false sense of user satisfaction that may arise during the development of the analogy and metaphor derived designs. That is an incorrect conceptual image of the designs is crafted by the developers, which clouds their actual judgment and perception of the interface. This is a vital problem since it leads to a common predicament where innovation is, literally, regarded as the solution; even if, revolutionary designs, in terms of new scheme, most often creates confusion among the target group as they cannot rely on prior experiences. In addition, prototypes that depend heavily on graphics also require specific hardware and therefore might not transition well to portable devices, where scarce resources dictate a dedicated approach.

The presented design models all share a common denominator in terms of extending the notion of cognition. That is by modifying representations, to reflect settle changes that are actively taking place, enables the user to mark the information they wish to remember. For example, people very often choose to write to-do lists to remember certain events and as they are completed the event is crossed off the note. This kind of modified cognition is called annotating which simply involves modifying the external representations such as crossing off an event or underlining items for easier access [18]. This notion is often used when shopping because people are well aware of the fact that they will not always remember what needs stocking up; and so externalizing it as a written shopping list circumvents the problem. Annotating is a very powerful tool for assisting the user in achieving their task, it can also be included as a method for the system to remind or notify the user of various modifications. As stated, common for all the design prototypes
4.2 Evaluating designs

above is the notification system which serves as a feedback mechanism; informing
the user of a specific applications status. For example, the notification icon is intu-
itively placed at the top right corner of the individual widgets, which instructs the
user of specific content changes. This adaptation allows the consumer to decide
whether they want to download the new content or not; potentially avoiding any
unnecessary charges.

4.1.6 Summary

The aim of this chapter was to introduce several graphical user interfaces based on
some of the issues surrounding design construction. One of the main advantages
of creating low-fidelity prototypes is the great ease of development and the output
of the study. Prototyping and scenarios are used throughout the design process
to test out ideas for user acceptance and feasibility. As a result the subsequent
section provides a concise summary of the previously presented designs by listing
their respective prospects in relation to the design principles, discussed in Section
3.3. In Table 4.1 the prototypes are graded according to a 1-5 scale, where low
values reflects poor performance, to provide a short and concise summary of the
various proposals.

<table>
<thead>
<tr>
<th>Design principles</th>
<th>One</th>
<th>Two</th>
<th>Three</th>
<th>Four</th>
<th>Five</th>
<th>Six</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visibility</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Feedback</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Constraints</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Mapping</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Consistency</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Affordance</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Total</td>
<td>27</td>
<td>23</td>
<td>23</td>
<td>22</td>
<td>20</td>
<td>14</td>
</tr>
</tbody>
</table>

Table 4.1. Design summary

4.2 Evaluating designs

Design evaluation is the final process of determining the acceptability and usability
of the user interface and is concretely measured in terms of the criteria discussed
in Sections 3.3 and 4.1.4. Coarsely, the variety of criteria covers: the number of
errors users make while interacting with the interface, how esthetically appealing
it is, how well it matches the requirements, etc. Interaction design demands a
high level of involvement from usability specialists since the final design derives
from several prototype iterations. In most scenarios there are numerous of design
activities concerned with quality assurance and testing to enhance the chances
of acceptance among the target group. A general misconception, however, is to
replace these activities with evaluation instead of adapting the notion to complement and enhance them. The main purpose of evaluation is to feed back the gathered data into further design solutions; once again referring to iteration, the key characteristic of the interaction design process [15].

Due to the importance of the evaluation process two separate paradigms called “quick and dirty” and “usability testing” was adapted to conduct the necessary usability studies [15]. The former is regarded as common practice and can be done at any stage of development. The emphasis is on fast input, with focus on finding any design errors that need refinement based on Shneiderman’s guidelines covered in Section [4.1.4] rather than carefully documented findings. For example, at an early stage in development the usability designers informally meet with potential users, or in this case colleagues, to collect general feedback regarding the ideas and the important factors concerning the interface design. However, seeing as the prototype development was conducted behind closed doors the user involvement stretched only to company employees.

The first prototype iterations discussed in the previous section received various internal feedbacks regarding their accessibility and appropriateness, which in due course resulted in the second installation of prototypes one and three. At the final stages of prototype implementation the concept of quick and dirty was revisited to try out the different color combinations, icons, and graphical elements. This technique proved to be very effective since it is done in a short space of time allowing the prototypes to evolve and receive the essential ingredients that produce a successful design.

The latter evaluation paradigm, usability testing, was the dominant approach at the final stages of development mainly measuring distinctive user performance. In contrast to the prior technique the testing involves carefully finding the user workflow based on prepared tasks that are typical of those for which the system was designed. During evaluation the user performance was measured in terms of number of errors and the time required completing the desired tasks. The observational data was thoroughly examined to determine why the users did what they did. These interaction issues defined the foundation of further prototype iterations, refining the graphical user interface through repeated design-evaluation-redesign cycles.
Chapter 5

Technology

The conducted design investigation, in terms of mobile capabilities discussed in Chapters 3 and 4, is exposed through two standardized application programming interfaces: Symbian OS and UIQ, generally accessed through C++, and web technologies, i.e. JavaScript, CSS, and XHTML, for the device browser. The subsequent sections introduce the utilized standards as the fundamental building block in achieving the desired results. Expressing the relationship between the platform and the corresponding paradigm, explicitly, is highly useful for describing the differences in approach as well as the inner workings of the presented prototype in Chapter 6.

In this chapter the different paradigms, idioms, and philosophies will be introduced without getting deeply involved in coding. These fundamentals constitute the essential foundation required to realize the proposed graphical user interface designs from Chapter 4. However, considering that the details of mobile application development vary considerably across the available platforms, the components behind each technology are not covered in detail. Instead, the comprehensive details are deferred to each platform’s developer documentation.

Briefly, this section consists of two parts; part one gives an introduction to the web concepts that are adopted, and part two covers the native standards provided by the Symbian platform.

5.1 Browser platform

Practically all portable devices are equipped with a browser; rendering the engine as the most widely deployed platform, given the tremendous proliferation of mobile phones. A rapidly growing subset of these devices comes pre-installed with a powerful web browser that includes support for JavaScript, CSS, and XHTML.

This section introduces these web concepts, putting them in the context of the mobile browser environment. Also, the fundamentals behind these technologies are covered through a brief look at the principles, followed by a very fundamental introduction to the available resources.
5.1.1 XHTML

Extensible Hypertext Markup Language, or XHTML, is a markup language conforming to the XML syntax, and is widely regarded as one of the standard publishing languages of the World Wide Web. It derives from a family of current and future documents modules that reproduce, subset, and extend the HTML standard. XHTML family document types have the same depth of expression as its relatives; however, the document type is designed to work in conjunction with XML-based agents [26].

Contrary to HTML, a very flexible paradigm, XHTML is an application of XML, a more restrictive subset of the markup language as it requires a well-formed structure. This implies that an XHTML compliant browser can theoretically omit error recovery procedures even if it may need slightly more error detection checks. Consequently, the reformulation is well suited for mobile devices where extra resources cannot be devoted to support the additional complexity of the HTML syntax [33].

5.1.2 CSS

Web standards, especially in regards to creative design and presentation, have come a long way since its first glimmer. In the early days, Cascading Style Sheets, or CSS, was mainly the preserve of developers. Since then the concept has successfully established a broad acceptance among the web community; as the complexity of services forced the separation of *document presentation* from *document content*. This distinct separation introduces improvements such as content accessibility, flexibility, and reduces the amount of redundant information in the structural content [30].

The standard is a simple mechanism for adding layout, fonts, colors, spacing, and other characteristics to web documents. Hence, by attaching style sheets to structured web documents the developer can influence the presentation of the content without forfeiting device independency or adding unnecessary markup tags. The concept also allows the same markup document to be presented in various styles for different rendering methods, like in print, on-screen, by voice, etc. [27].

5.1.3 JavaScript

Client-side script is a program that may accompany a markup language document, i.e. XHTML, or be embedded directly as an extension to the document content. The actual program only executes on the client machine when the document loads or as trigger when certain events are activated; for example, when the user requests access to the menu or a hyperlink. Thus, scripts in general, regardless of the actual programming language, offer the developer a means to extend the web documents in highly active and interactive ways [28].

Despite the name, JavaScript is essentially unrelated to the object-oriented programming paradigm Java; on the other hand, these share a common denominator in terms of providing similar syntax and naming conventions. The common
foundation and ideology logically implies relations, but the script was designed to be easier for non-programmers to work with. The latter being one of the main reasons for the wide adaptation of JavaScript in current web services [34].

5.1.4 Asynchronous JavaScript and XML

Asynchronous JavaScript and XML, or Ajax, represents a fairly new web development paradigm based on open standards, covered in prior sections. Ajax is not to be confused as a new programming language but rather an umbrella term for design enhancements to improve traditional web applications. The paradigm signifies an approach in which web applications communicate and exchange data behind the scenes. This approach enables partial page rendering which creates a dynamic and responsive environment.

“The significance of Ajax lie in the fact that it provides a superior user interface using technologies already present in browsers i.e. it requires no supplementation” [10].

The main purpose of this concept is to prevent complete page reloads to achieve higher levels of interactivity and usability. Compared to the previous generation of web applications, web services integrated with asynchronous requests render the entire server-client communication as transparent to the user. The model allows applications to push new data to the browser interface via DOM manipulation and so avoid unnecessary page loading and annoying pauses in the workflow.

5.2 Symbian platform

The Symbian operating system is regarded as the leading platform in the smartphone market; and the platform is used in a variety of mobile phones with a wide range of differing screen sizes and input mechanisms.

This section covers the paradigms and concepts that constitute as the foundation of the Symbian OS architecture and its subsequent C++ frameworks. Since this entails discussing several aspects of the system, the discussion is firmly divided between the operating system basics, covered in 5.2.1 and the framework principles that encompass them, covered in 5.2.2 and 5.2.3.

5.2.1 Symbian OS

Symbian OS is a proprietary operating system, a software with restricted copying and modifying rights, designed for battery powered portable devices i.e. mobile phones. This profoundly affects the general design and the associated user interface frameworks, libraries, and reference implementations of common tools. The kernel is designed for 32-bit ARM processors, which operate at very modest speeds in comparison to their workstation relatives and is a direct descendent of the EPOC family of operating systems. The book [9] provides an excellent grounding in this topic covering the fundamentals behind Symbian OS, the system, and the framework that encompasses it; however a brief outline on the subject is given in following sections.
Symbian OS is developed with regards to three core principles: the integrity and security of user data is paramount, user time must not be wasted, and all resources are scarce \[^{[31]}\]. The hybrid design of the kernel introduces a complementary micro-kernel built as a personality on top of the real-time nano-kernel which is responsible for primitives such as fast synchronization, timers, initial interrupt dispatching, and thread scheduling. The latter, nano-kernel, also referred to as Real Time Operating System provides low interrupt and thread latency and does not deal with any dynamic memory management which is outsourced to the micro-kernel \[^{[9]}\]. As a result this design focuses on user-driven recourses-constrained operations optimized for portable devices.

The Symbian OS kernel is a lightweight pre-emptive multitasking operating system which runs in privileged while the remaining services run in non-privileged mode. All access to memory and hardware is restricted through the integration of a memory management unit. This model protects and only allows for kernel-side code to access the memory belonging to any active application. The concept actively segregates the different memory protection units; the memory protection is called the process while the unit of execution is called the thread. Consequently, the threads are the only units scheduled and not the processes, which are in essence thread and memory protection containers \[^{[9]}\].

In order to provide a strong assurance of server integrity the integrated micro-kernel architecture provides client-server relations. The Symbian OS server is regarded as a programming interface which can be utilized by the client to gain access to various services, thus creating an explicit boundary between the two entities. The inter-process communication known as a session comprises of client requests and server responses \[^{[9]}\]. This concept is designed for event-driven user-initiated interactions between the application, the user, and the corresponding server that handles the requests.

Accompanying the operating system is the Symbian OS C++ framework which refers to the domain specific C++ dialect used to develop the kernel and the related software. The paradigm differs slightly from its desktop relative since the dialect emerged from twenty years of experience in the portable device domain. For instance as thread management and synchronization are difficult to handle by the developers the notion of active object is introduced by the Symbian OS C++ paradigm for implementing concurrency; a concept that is otherwise absent in the C++ language.

5.2.2 Symbian OS Application Framework

Graphical user interface application development in Symbian OS is supported through the UIKON architecture, an application framework subsystem. UIKON can be coarsely segmented into two important components; the control environment, CONE, the framework for graphical interaction and the application architecture, APPARC, the framework for applications and their respective data. This architecture model provides a foundation for various mobile phone platforms, like UIQ and S60, to run on top of the native operating system; however, only the former is discussed further in this chapter.
The framework revolves around the Model-View-Controller design pattern, which encourages encapsulation of the different application logics. Application roles are represented by separate classes: the Model, the View, and the Controller which provide the basic functionality of an application. In addition to these classes the application interface element definitions, i.e. menus, dialogs, shortcut keys, placement, etc., are defined in a resource file [9].

The application class ; application entry point. Contains a factory for the application document, to define properties, and an interface to the resource file.

The document class ; aside from creating the application user interface, responsible for storing and restoring the applications data.

The application view class ; display the application data on the screen and provide handlers for user interactions.

The application UI class ; provides the commands logics and creates the application view.

5.2.3 UIQ Platform Framework

UIQ extends the Symbian OS platform by adding graphical libraries to provide additional device specific controls to the core system. The library is referred to as QIKON and is regarded as abstract interface layer on top of the Symbian OS UI architecture. Essentially, UIQ is a set of graphical user interface components to enable the development of feature rich mobile phones that are open to expanded capabilities [32].

The library is primarily designed for devices with additional user input interfaces like touch screen, stylus, as well as the standard keypad interaction mechanism. User interface applications created for the UIQ platform derive from base classes provided by the libraries which sequentially derive from the Symbian OS framework.
Chapter 6

Implementation

In a usable interface the design and visualization encourages and supports the user in exploring and understanding the presented information. To leverage some of the valuable characteristics, discussed in previous chapters, the final design combines different media elements into a coherent presentation instead of the rigid structures imposed by current solutions. The final user interface consists of design fractions from several different prototypes, introduced in Chapter 4, to create an effective and intuitive environment that supports the interpretation tasks of the user. Visual structure and presentation in this graphical user interface are indentified through a moderate review of existing applications, Section 2.1, that are complemented by the study conducted on the previously covered prototypes, Section 4.1.5.

After a series of chapters describing individual aspects of mobile human-computer interaction and application frameworks, it is time to pull a few threads together by introducing two simple non trivial implementations based on the technologies covered in Chapter 5. With regards to the mobile interface purpose stated in Chapter 1 and the time table associated with this thesis secondary presentation goals are formulated. The final prototypes, derived from the previously presented solutions in Chapter 4, are to only support visual presentation with focus on the main dashboard lacking any necessary working functionality and algorithms, hence considered as non-operational.

In this chapter, the application graphical user interface is divided into two distinct categories by their respective technologies and frameworks. The two categories cover the interface structure, behavior, and design of each prototype under these versatile yet diverse runtime environments. To address the discussed issues, this chapter makes two contributions; the first an user interface implementation in the native phone environment, and the second presents a browser solution covering the same topic. The two implementations are, however, only covered from a design perspective to avoid direct references to the actual code.
6.1 The development process

The primary idea behind prototype construction is to provide the development team with user feedback of emerging design proposals, or feedback confirming that the initial idea is not technically feasible. As stated in Chapter 3, prototypes are produced in order to validate or answer questions regarding specific attributes of the interface. Producing something concrete in conceptual design, however, requires more consideration and planning of the actual design details. This allows the user to evaluate the design more seriously as they are able to interact with the device and perform tasks supported by the product in the manner intended [18].

Prototype implementation involves considering detailed issues of designing the graphical user interface, such as screen or keypad design, how to structure the menus, which icons to use, etc. Note that there is no rigid border between the physical design and the conceptual design; due to the fact that constructing design proposals inevitably implies making more thorough investigation regarding specific graphical elements [18]. The lack of distinct boundaries is the result of the iterative nature of interaction design; some detailed issues might come up during conceptual design while, similarly, other might only arise during physical design. This requires the usability specialists and product engineers to occasionally revisit the decisions made during each development phase to appropriately address the issue at hand. Thus, the exact outline is not relevant but rather the significance of separating these two concepts; allowing development to be done freely without being tied to constraints at an early stage, as this will eventually inhibit creativity.

The interface design act is a process of delicately balancing the environmental, user, data and usability requirements with functional requirements through well investigated and adapted choices and decisions. These are often in conflict seeing as the mobile phone is expected to provide a rich environment with a wide variety of functionality, but is constrained by having scares recourses such as a small screen and limited input mechanisms, covered in more detail in Section 4.1.1. This means that the display of information and the number of unique functions is restricted, resulting in limited views and the need to associate multiple functions with one function key; a process which actively dissolves the highly regarded one-to-one mapping phenomenon [18]. However, to avoid some of these common pitfalls several principles of good design in the context of some universal interface elements were introduced in Section 4.1.4.

6.1.1 Graphical elements

Graphical user interfaces are made up of various elements such as icons, menus, toolbars, dialog boxes, etc. These elements must be chosen from a set of pre-designed graphical components, or designed entirely from scratch by the usability specialist. Sometimes these decisions are made for you through the use of a style guide, more on that in a moment, and occasionally the underlying system might be incomplete and in that case no guidelines are provided. The actual decision, consequently, depends on whether the system provides the appropriate graphical libraries and how well suited the components are for the intended product.
The style guide dictates the look and feel of the entire interface that is, which elements should be used for which purpose and what they look like. For example, each desktop application has the same layout, in terms of design, as the operating system they are running on. This exclusive identity is the result of adapting the system specific style guide that captures the graphical aspects as well as the placement of certain elements. The developed prototype for this thesis revolves around two separate paradigms; the first borrows its identity from UIQ libraries, covered in Chapter 5, whereas the second inherits its characteristics from web-based services.

The subsequent sections will briefly cover the three main aspects of the implemented prototype: menu design, icon design, and screen layout. These are concepts applicable to a wide variety of interactive products; however, they are primarily discussed from a portable device perspective and might not transition well to dissimilar products or settings.

6.1.2 Menu design

Interface menus provide the user with the alternative to: select a set of commands, or a set of options related to a specific command. They are also associated with task structures and the information required to perform the desired action. Hence, menus provide the means by which the user can perform actions related to the task in hand. These are in general designed as drop-down, single dialog, or pop-up menus and must adapt a set of guidelines to produce a coherent design that delivers the desired message to the intended target group.

As covered in Chapter 3, it may seem obvious how to design a menu; however, the process is not always as straightforward and depends on more than one factor. Determining the menu content, position, and style is a very time-consuming and difficult task; regrettably, this process is taken far too moderately in interface design which often is the key reason for why an otherwise excellent product fails. Consequently, to provide an application that is easy to use and offers user satisfaction some important points must be taken into account. For example, for pull-down and pop-up menus, the most commonly used functions should be placed at the top of the list, to avoid frequent long scrolls and scans. The principle of group introduced in prototype two, Section 4.1.5, can also be used to good effect in menu design. For example, as the complexity of the menu grows the functionalities can be divided into collections of items that are related, with each collection being separated from others and, thus, yielding greater overview. Further, opposite operations such as “save” and “close” should be clearly separated to avoid accidentally losing work instead of saving it.

To effectively illustrate how the design of menus should proceed, the subsequent sections will refer to the implemented prototype shown in Figure 6.1. During the initial data gathering, discussed in Section 4.1, a number of essential tasks were identified. These include creating an account, signing in and out, and changing application attributes. Tied to these tasks would also be a number of “housekeeping” and administrative actions such as finding, downloading, sharing, rating, and removing widgets. The first step is to determine what to call the menu entries.
They need to be clear, short, intuitive, and unambiguous as the available space for listing them is restricted. However, the commands must remain distinguishable, that is not easily confused with another, so that the user will find the sought after action and not choose the wrong one by mistake.

The next step is to consider the logical groupings. In this case, the design focuses on the one-to-one mapping concept to provide visibility in terms of general overview, and affordance through facilitating the user with a simplistic and informative design pattern. The commands are also grouped according to user goal that is all administrative actions specific to a widget are conveniently reached through their individual menus. Similarly, the account management, the application settings, and the downloading task are all contained and represented as a separate widget, maintaining interface consistency, illustrated in Figure 6.1. The task of finding new widgets is generalized to a simple search option that lead to a list of applications available for download; however, the user is also presented with alternative options such as finding widgets based on rating or submission date.

6.1.3 Icon design

Designing icons is another process that may seem obvious, but developing a representative and informative icon takes more than a few minutes. Producing mockup icons may be regarded as a fairly easy task, however, such design solutions are unlikely to be widely accepted by the target group; as a result of the behavioral constraints initially discussed in Section 3.3.3. For example, symbols for repre-
senting men and female toilets might only be appropriate in the region which they are established, as tourists might find the signs deceptive. Thus, designing proper icons is a very time-consuming task that is strictly context specific and highly dependent on logical and cultural constraints [18].

As a minimum, designers should always consider revising existing traditions or standards and certainly avoid contradicting them. It is easier to establish a relationship between controls and their corresponding effect if the action to represent is a concrete object, or based on internationally recognized symbols. Mapping concrete objects only requires a picture of the item to represent its corresponding functionality, a topic discussed further in Section 3.3.4 while new innovative tasks need usability testing to ensure that the affiliations are understandable in the manner intended. Action are harder to represent but can occasionally be captured as long as the icon provides the user with sufficient clues to understand what it implies. Associating to peoples conception of physical objects, Section 3.3.6 can help creating the necessary conventions and affordance; allowing the user to recognize the icons effect at a glance and therefore rendering the user interface as less error-prone.

In the widget prototype a set of informative icons needs to be identified for each of the tasks: account management, changing application attributes, and downloading new widgets. As previously discussed some cognitive aspects relevant to icon design must be well thought-out; if you prefer, icons must be designed so that the intended users can readily perceive their meaning, and easily tell one from another. This can be a very difficult task to achieve since their size is very small [18]. However, this issue was addressed through slightly remodeling existing concepts and placing informative captions beneath each action, illustrated in Figure 6.1. The account symbol is represented through an outline of a simple character, application attributes is illustrated with a few cogwheels which is a fictional representation of the systems underlying complexity, and the download functionality is cleverly masked behind an arrow pointing down to symbolize the notion of retrieving information.

### 6.1.4 Screen design

Screen design can be divided into two different aspects: how the task is split across a number of screens, and how the individual screens are designed. The former aspect can be appropriately supported by reference to the requirement and task analysis covered in Section 4.1 which broke down the task into plans of action. The collected data from the analysis are then translated into separate screens, each representing a task or a subtask [18]. The corresponding interactions are then divided into simple steps, involving a decision or a data entry.

This form of interaction design can be regarded as idiotic and the final layout requires numerous adjustments and redesigns; given that presenting too many screens can be as equally frustrating as having too much information crammed into a very limited space. Another issue concerns the more complicated tasks where the pertinent information must be distributed correctly across several screens, to provide easy access to the desired data at relevant times. However, this is one
of the balances that the implementation drew from as a result of adapting the process of screen design [18].

The latter aspect, concerning individual screen design, derives more clearly from some of the visual communication principles covered in Sections 3.3 and 4.1.4. For example, design the screen so that the target group attention is drawn immediately to the salient areas, and adapt pattern techniques such as color, motion, boxing, and grouping to assess understanding and provide clarity [18]. Hence each screen of the implemented prototype, illustrated in Figure 6.2, is designed to divert the user attention towards the functionalities that are appropriate and useful to the task at hand. Also unnecessary animations and distractions not relevant to the task are avoided to maintain constant awareness of the state of the system, and effectively preserving the user focus.

Good integration of organizational and conceptual methods can also help users make sense of and interpret actions within their respective context, as discussed in Section 3.3.5. This is yet another scenario where the principles of good grouping can be applied. The implementation takes this concept in high regards and distinctly separates the widgets by representing each as a box shape neatly and spaciously arranged according to a two-dimensional grid pattern, Figure 6.1a. In addition, the user is presented with an alternative view, illustrated in Figure 6.1b, where the widgets are arranged as a list similar to the sheet model introduced in Section 4.1.5. Again, to maintain consistency throughout the design the individual widgets are presented as box shapes; however, in contrast to the previous grid view, the placeholders containing the applications are now scaled according to the screen resolution. This design decision actively introduces new accessible interaction and presentation mechanisms that leverage from the prospects introduced.
by these very dissimilar yet complementary screen layouts. A pattern that was thoroughly tested; since there is a trade-off between sparsely populated screens with too much open space and overcrowded screens with too many complicated icons. Consequently, this design was adapted to avoid distracting and confusing the user with cluttered screens and disjointed series of interactions.

6.1.5 Prototype structure

In the original design, the focus was on developing an interface where the potential target group was used as models during the evaluation. Furthermore, the final prototypes were more concerned with providing an intuitive interaction design, with regards to the users’ needs, than with developing elegant programs. That is, the usability specialists do not portray themselves as models of the users, but rather adopt personas to convey the user requirements. This approach effectively prevents mismatches between the design and how users try to find the desired information. A procedure that is noteworthy, since it avoids the common predicament that results in badly designed systems. This technique, literally, includes the target group in the development process instead of erasing the only viable source of feedback and information.

In prior sections the focus has mainly been the graphical elements. This section, however, provides glimpses into the actual implementation process for the different types of platforms. The two case studies roughly discuss the construction of the prototypes in terms of, how the essentials behind the graphical user interface were distributed in various classes. As stated in the introduction to this chapter, the two implementations are only covered from a design perspective to prevent unnecessary code references.

To truly appreciate the extent of the differences between the diverse platforms, the graphical structure of the prototypes are exposed and covered from an interaction designer’s perspective. The relevant class structure for each platform is illustrated in Figures 6.3a and 6.3b. The shaded classes are the layout specific files containing the necessary details regarding the actual graphical user interface pattern and placement. Noticeable is the fairly similar arrangement of document presentation and document content, although there are major differences evident in the underlying application programming interfaces. Given that the main focus is user interface design, the application entry points and the exact specifics regarding framework architectures are irrelevant. However, for more details concerning the components behind each technology refer to their corresponding documentation.

The implemented prototypes both share a common denominator as they draw influence from the Model-View-Controller design pattern. That is the application is split into separate logical parts that encapsulates the different aspects of the whole system; where each distinct part in the application has a specific task and role. To support this model the application was roughly segmented into the design pattern, the application engine, and the application user interface, as illustrated in Figure 6.3.

The design pattern represents the style sheet model for the application. It provides a repository of graphical attributes defining individual interface elements,
serves to define certain properties of the application, and offers an interface to the applications appearance. This class is represented by the cascading style sheet in the browser prototype, and by the resource file in the UIQ version. Unlike desktop developer, portable device developers target a wide range of hardware platforms, each of which may require a different design pattern. Hence, keeping the design patterns separate introduces a layer of abstraction that simplifies the development and the efforts required when moving applications between various hardware platforms.

The application engine is entirely invisible to the user. That is, it creates the application view and handles the drawing and eventual screen-based interactions. In addition, it contains the means of processing commands that may be generated by various interaction mechanisms. In general terms, the view can be defined as the class that enables actual rendering of some or all of the applications data. This process is managed by the application view class in UIQ and by the main JavaScript class in the browser version.

The application user interface is a concrete control whose purpose is to display the application data and status on the device screen and allow the user to interact with it and distributes keystrokes to appropriate controls. This class is intentionally maintained clear of any unnecessary data and is very simple in nature. The UIQ prototype implements this concept through the application user interface class, whereas the browser version utilizes a markup language to convey the necessary information to the user.
6.2 Evaluating implementations

Having implemented the two prototypes and identified the goals and main questions, the next step in the development process is to choose the necessary evaluation paradigm. As covered in Chapter 3, the evaluation paradigm helps determine whether the kinds of techniques adopted were appropriate during the process. The goal is to study the diverse implementations with a combination of techniques to obtain different perspectives that combined can reveal a broader picture. This will serve as the foundation for which some judicious triangulation methods are used to only filter out the desired results.

The first step towards ensuring a usable interactive graphical user interface is to include the practical issues in the final equation. By thoroughly understanding and perpetuating useful aspects the trade-offs in terms of strengths and weaknesses are effectively identified and in either case documented. For example, what might seem as the most appropriate set of techniques may prove to be far too expensive, or may be too time-consuming, or may require equipment or expertise that is not available, or it may not even be technically feasible [18]. Thus, in all these cases some compromises are needed in order to achieve the initial goal.

As previously stated a central concern regarding the user interface is to develop interactive products that are usable and engaging. This logically implies that the graphical user interface is easy to learn, effective to use, and provides a memorable user experience. However, these requirements are strictly related to the underlying technologies used. For example, the prototype developed for the browser environment depends heavily on its support and is restricted to the functionalities supported by the browser engine. This introduces several concerns regarding the portability and the accessibility of the final product. Given that the application might not transition well to new settings since it is designed for a specific type of browser. It goes without say that a key aspect of any software is to utilize its services in similar products without having to readjust and redesign its entire core system, which consequently renders the browser prototype as a poor solution. Still, the rigorous separation of the web user interface from the underlying application introduced one clear benefit; it provides artistic persons with the tools to accomplish their goal, a task that is otherwise unreachable to developers with no prior programming experience for Symbian devices.

Another concern is the interaction delays introduced by the browser version though it prioritizes its own functionality ahead of the widget service; creating situations where the user might experience a noticeable delay that affects their workflow negatively. In terms of accessibility the Symbian version is far superior since the application is executed as a separate process there is no real concerns regarding feedback delays. On the other hand, the development time is far longer and introduction of new innovative solutions might not always be feasible within the desired time restrictions. However, the Symbian version introduces no real portability issues since once developed the application will undoubtedly function on devices with similar support. In essence, deciding what platform to use depends on the purpose and the goal of the initial project.
Chapter 7

Discussion

The ongoing investigation of this thesis has enriched and expanded the scope of human-computer interaction because it is concerned with a much broader scale of issues, paradigms, topics, and platforms than in the traditional sense. Constructing a graphical user interface based on the interaction design fundamentally entails creating a user experience that enhances and extends the manner in which people communicate. This notion has been extended beyond those intentions by embracing new paradigms and consequently a wider range of issues. These include pervasive and ubiquitous computing that makes use of different platforms and technologies to establish a greater acceptance among the target audience. In either case this only reflects the importance of design, as success is only gained through a mixed set of skills reaching from psychology and interaction design to computer science and marketing.

This thesis introduced several principles of how social, cognitive, and affective issues are addressed and integrated into the design process. It also covers the central theme of graphical interface design and evaluation as an extremely iterative process that relies on good practices and reasoning to produce the desirable results; refer to Chapter 3 for further reading. The theories discussed was then applied in the subsequent chapter with the presentation of several user interface prototypes focusing on different design decisions; covering the evident trade-offs of portable device paradigms and hardware limitations. Chapters 5 and 6 then proceed with describing the necessary technologies and concrete design practices required to develop one prototype under two very different platforms.

This chapter conducts a discussion around the research based on different opinions and interpretations gained throughout the development process. However, these are only to be regarded as the authors’ opinions and not to be confused with actual facts but rather an internal dialog regarding various issues and results.

Simplicity is unquestionably one of the important aspects in product development; a notion applicable both in terms of design principles and actual implementation. A common misconception, however, is the approach where the product functionality is regarded as the central key to success, which as discussed throughout this thesis is rarely the case. This, in due course, underpins designers’ des-
perate attempt to cram too much into a screenful of space, rendering the product unusable and making it unwieldy for anyone to find what they are interested in.

Nielsen and his colleagues [13] propose that usability specialists go through all of their graphical elements to determine what can be discarded without impacting the design negatively; if you prefer, they propose the means to preserve the overall function of the product whilst maintaining a clean and intuitive design that is understandable in the manner intended. However, the key is not a plain vanilla application but rather finding the equilibrium between aesthetic appeal and the appropriate amount of information per screen view.

Too often do developers despair at the seemingly scarce interaction mechanisms a portable device provides; the low resolution, small screen, and tiny keypad. These input and output technologies place the portable device market as an impoverished product in respect to the richly expressive desktop interaction abilities. This deficiency becomes more evident when moving from conventional technologies to limited handhelds that offer little room for new innovative approaches, that the target audience might find fulfilling. Nevertheless, this mentality towards portable device needs to be challenged and initiative has to be taken to introduce new accessible products that leverage from the prospects of mobility. The much reduced hardware resources of the device is a big limitation, but as presented throughout this thesis there are technology developments that will allow designers to effectively address some of the problems from the mobile context.

Another less evident misconception is the false sense of satisfaction that designers may gain during the development process; given that they spend countless hours working on the system trying to find the interaction glitches. This is a common problem in design and is partially affected by the designers psyche. Despite of knowledge and experience the designer is often considered as the worst judge of the graphical user interface they design. They will approach the interaction from a very different perspective than the intended target group; as their knowledge regarding every aspect of the system is thorough. Thus, actions that seem straightforward and appropriate can be interpreted as mystical or bizarre to the user. Also, by investing a great deal of time and intellectual effort developing the product renders the developer as fairly persistent and protective, therefore more likely to avoid criticizing the actual outcome.

In essence, it is very convenient to have a restricted vision of the portable device future, especially the mobile phones market. To recognize portable devices as simply supercharged phones carried in the pocket, a great deal of their potential will be lost. Instead, their affect and importance should be acknowledged, and their potential to be key building blocks in the new digitally amplified world should be graciously welcomed. The technology is pervasive and personal; a well thought-out innovation and design can change the current perception from accessory or sophisticated trinket to a personal assistant that provides highly customized information.
Chapter 8

Conclusion

This chapter ties the conducted studies together by first including short and concise statements of the conclusions, based on the evaluations made at the end of Chapters 4 and 6. These are covered according to the subsequent research objectives, introduced in Section 1.1. To conclude, a short presentation regarding possible work and potential paths that can be investigated in future iterations of the product is given; if you prefer, a short presentation of new problem statements that originate from this research describing future work that may be of interest.

To recap, the main research objectives in this paper are:

- Investigate different realizations of mobile widgets from a user experience point of view, specifically widgets hosted by dedicated runtime engines, or as Rich Internet Applications for web browsers.
- Study the effects of adopting a user-centered approach to widget user interface and interaction design.
- Analyze how two similar projects, conducted in different environments, can face dissimilar consequences in terms of development related problems and restrictions.
- Investigate the significance of non-functional project requirements, i.e. time constraints and the necessary experience, and their corresponding influence on the development.
- Briefly investigate possible techniques to account for the differences presented by the underlying platform.

As frequently stressed throughout this thesis, a user-centered approach to interaction design entails iterative design-evaluation-redesign cycles; as progression is made from the initial proposals through a variety of prototypes to the final product. The number of cycles depends on the concrete constraints of the project, such as deadlines, available equipment or expertise, etc. To ensure some degree of perfection the cycles require a variety of skills, often involving difficult decision making and multitasking. Also, many unforeseen practical issues and events, like
Conclusion

prototypes not working properly, or time to completion being reduced, need to be dealt with. Hence, it is required to have excellent knowledge concerning the range of techniques that can be adopted into the process when needed. However, finding ways to cope with these diverse set of problems, confronting a product, is often regarded as part of the challenge and excitement of interaction design.

Throughout Chapter 6, many tactics have been proposed to follow; to eventually help the target group understand how to use the design. However, it is very important to be clear regarding the distinctions among them, since these have different functions and implications. Needless to say sloppy thinking about tactics and concepts can be the difference between product acceptance and rejection in the marketplace. Sloppiness in design always translates into confusion among the target group and as previously stated; if the user reactions are negative in terms of usability an otherwise excellent product could fail.

Sections 4.2 and 6.2 discussed the questions, decisions, and concerns that come to pass throughout the design and construction of the user interface prototypes. General issues in terms of how to involve users, how to include their needs into the equation, how to understand the problem space, how to design a conceptual model, and how to go about designing and evaluating interfaces, have been extensively covered; by stating that the entire process is a symbolic communication link, one that works only if it follows the purposed conventions. This reflects how two similar projects face a different set of problems in terms of constraints and crises regardless of how related they initially are.

The analyses based on the comparison between the platform and the internally conducted usability testing of these different settings revealed many more types of development related problems than initially anticipated. Those problems discovered tend not to be critical issues concerning usability but rather related to the actual implementation process. Some of these problems are directly associated with the prerequisites introduced by each development platform. With regards to platform specific operations, the personal response was less positive and more negative when developing for Symbian than for the browser. Regrettably most graphical libraries available under Symbian OS are difficult to develop for considering that the paradigm is difficult to engage, which discourage developers because they are inhibited about the time constraints and lack of experience. However, the gain in development time introduced by the browser platform is, on the other hand, irrelevant if the response time of the product is vital. The problem is, in essence, inevitable since the application is a part of the browser process which eliminates the possibility of any low level optimizations that might be required.

This study also investigated reasons to account for the differences summarized above through a judicious separation of the application core and the graphical user interface. In short, the most important design principle is that of coherence and understandability which is more or less applicable during the implementation process as well. By dividing the application functionality and user interface into two separate entities all of the necessary actions will be perceived as an abstraction layer above the underlying system; thus, providing the designers with the option of constructing the design solutions under well established tools, and in so doing avoiding unnecessary constraints. Although there is no explicit or unified way
to explain how this separation will affect everyone, the prospects introduced by this method cannot be ignored, especially when this adaptation technique might provide the advantages of both paradigms.

8.1 Future work

The two different case studies conducted in this thesis illustrate how combinations of design and evaluation concepts provide a well intertwined process that eventually arrive at a design or redesign for a product. In short this process implies that some design decisions will take place while investigating requirements while others are made during actual implementation; in either case the design will literally evolve through a series of design-evaluation-redesign cycles.

This notion is stressed out throughout this report and inevitable in a competitive and innovative market. However, in practice the usability testing may face implications for how it could be evaluated because the product could not be demonstrated to people outside the development team; in fear of losing some advantages for being the first in the market to introduce such features. Also, practical constraints in terms of time restriction and unavailable expertise help create new problem statements for future research. The subsequent propositions cover subjects initiated by improvement suggestions as well as potential projects that can expand and build on top of the presented foundation.

- Involve the target group throughout the process to gain deeper understanding of how well the product assists the user in their tasks.
- Conduct more detailed surveys regarding user requirements and needs.
- Investigate possible solutions for distinctly separating the application core from the graphical user interface, allowing the developer to adopt different technologies for each aspect of the system.
- Implement a complete functional version of the widget engine for more precise evaluation and performance comparison.
- Introduce innovative solutions in terms of new interaction mechanisms that can help coordinate the user in achieving their initial goal.
Bibliography

[1] Brian Braiker. The year of the widget?: User-generated content was a hallmark of 2006. it's a fair bet 2007 will be all about further customizing your online life. Newsweek, 22 December 2006. URL: http://www.newsweek.com/id/44320


Copyright

The publishers will keep this document online on the Internet — or its possible replacement. In addition, the publisher has taken technical and administrative measures to assure authenticity, security, and accessibility.

The online availability of the document implies — permission to make digital or hard copies of all or part of this work for personal or classroom use without fee — provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. Subsequent transfers of copyright cannot revoke this permission. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires the consent, or the specific permission, of the copyright owner and/or a fee.

According to intellectual property law, the author has the right to be credited when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Luleå University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: http://epubl.luth.se/avtal-en.shtml

© Hani, Sean, Movahed