Adaptive User Interface Patterns for Mobile Applications

Kornilova Olga

ITÄ-SUOMEN YLIOPISTO

Faculty of Science and Forestry

School of Computer Science

UNIVERSITY OF EASTERN FINLAND, Faculty of Science and Forestry, Kuopio
School of Computing
School of Computer Science

Student, Kornilova Olga: Adaptive User Interface Patterns for Mobile Applications
Mater's Thesis, 124p., 1 appendix (105 p.)
Supervisor of the Mater's Thesis: Prof. Pekka Toivanen, Prof. Petri Vuorimaa, Jaakko Parviainen
Advisor of the Mater's Thesis: Evgenia Litvinova
June 25, 2014

Abstract:
Over the past few years, there has been a widespread adoption of handheld devices. Despite the efforts of mobile software producers to improve user interfaces there are still many challenging usability problems in mobile applications. These issues are due to the limitations of mobile devices, new input methods, devices mobility, etc. Constraints and usability issues of mobile devices can potentially be addressed by *Adaptive User Interface* (AUI) techniques, which have been identified as one of the promising solutions for usability issues in modern informational systems. In addition to this, such properties of handheld devices as mobility and context-awareness make mobile devices a perfect platform for using context-based user interface adaptation techniques.
Nevertheless, there are no clear, accepted and widely used guidelines or set of patterns for *AUI* creation for mobile applications and most of the solutions are left for application developers' discretion.
The Thesis focuses on the topic of *AUI* for mobile applications. Particularly, a possibility to create a standardized approach to *AUI* design and prototyping of mobile applications is considered.
The main objective of the Thesis is to design guideline rules and associated patterns as the examples of a possible standardized approach. The designed guidelines are validated by applying them to two different mobile applications. The result of designed guidelines might possibly be used to influence further development of *AUI* methods into guidelines and specifications for mobile application development.

Keywords:
Adaptive User Interface, Patterns, Guidelines, Sensors, Context-Awareness, Mobile Device
CR Categories (ACM Computing Classification System, 1998 version): A.m, K.3.2

Foreword

This paper is a result of my work experience, academical research and personal interests in this area. It has been a long journey to this final stage and I would like to give credits and thanks to everyone who supported me during this time.

I would like to thank my supervisors Professor Pekka Toivanen and Jaakko Parviainen from University of Eastern Finland, Professor Petri Vuorimaa for the provided opportunity to do this research, special thanks to for my academical kick-ass Evgenia Litvinova. Finally, I would like to express my gratitude to Professor Martti Penttonen for his great and valuable support and help throughout my whole master's education.

Espoo, June 25, 2014

Kornilova Olga

List of Abbreviations

API	Application Programming Interface
AUI	Adaptive User Interface
GPS	**Global Positioning System** A space-based satellite navigation system
GUI	Graphical User Interfaces
HCI	Human-Computer Interaction
IEEE	**Institute of Electrical and Electronics Engineers** The world's largest professional association dedicated to advancing technological innovation and excellence for the benefit of humanity [IEE, 2014]
OS	Operation System
PC	Personal Computer
SMS	Short Message Service
UI	User Interface

W3C	**World Wide Web Consortium** The World Wide Web Consortium is an international community where Member organizations, a full-time staff, and the public work together to develop Web standards [W3, 2014].
Web	World Wide Web
WHATWG	The **Web Hypertext Application Technology Working Group** The Independent Web Hypertext Application Technology Working Group [WHA, 2014].
WiFi	**WiFi** Wireless local area network (WLAN) products that are based on the Institute of Electrical and Electronics Engineers' (IEEE) 802.11 standards [WIF, 2014].
WLAN	A Wireless Local Area Network
WWW	World Wide Web

"It is not the strongest of the species that survives, nor the most intelligent that survives. It is the one that is the most adaptable to change."

Charles Darwin

Contents

List of Tables

List of Figures

Chapter 1

Introduction

With the explosive growth of mobile devices adoption and decrease of conventional PCs usage (1.86 billion smart phones/tablets vs 341 million PC/laptops in 2012 [Meulen, 2013]) as well as increasing Internet penetration. The ways of human-computer interaction are also shifting towards more seamless and ubiquitous, integrating computers to everyday life. Even though mobile operating systems have made a giant evolutionary leap in improving user interfaces in the last few years, there are still many challenging usability problems. Due to them, many common tasks are still easier and faster to perform with traditional keyboard and mouse on a *Personal Computer* (PC). The touch screen as the input device is less precise than keyboard and mouse, and has less room to display *User Interface* (UI) elements. These tasks should be performed differently on a mobile device, i.e. more adapted to the context in which they are carried out. Nevertheless, mobile operating systems still carry many paradigms inherited from the desktop software, that are not always suitable for mobile, and mostly they are improving only by borrowing successful new concepts from each other.

The modern mobile devices are equipped with highly sophisticated sensors and this particular feature may assist the mobile device to be aware about the surrounding, system's state and other contextual data. This information can be used to make the behavior of the mobile system more flexible and

more tailored. With future hardware development the variety of sensors and detecting algorithms might be extended rapidly and as a result bring more options for gathering context data. Better knowledge of the surrounding environment and analysis of the context usage could bring more possibilities for system adaptation to serve users in a personalized way.
Adaptation of the user interfaces has been identified as one of possible solutions for solving usability issues in modern information systems [Alvarez-Cortes et al., 2007]. A combination of such properties as mobility and context-awareness makes mobile devices a perfect platform for adaptation techniques. Moreover, mobile devices constraints and usability issues can potentially benefit by addressing them to adaptive user interfaces [Al-bar and Wakeman, 2001].

1.1 Overview of the Problem

The context-aware mobile systems have gained popularity in past years [Bobek et al., 2013] the amount of research in the field of context-awareness is growing year by year [Hong et al., 2009] as well. Many researchers have been concerned about the context-awareness as a property of mobile devices and potentials of this property for adaptivity of user interfaces and systems. There are numerous researches on the topic of adaptive and context-aware user interfaces, which present statistical [Gajos et al., 2006] [Holzinger et al., 2012], formal [Hanumansetty, 2004] and other approaches. But most of these works concentrate on some generic, high-level solutions which may be suitable only for some specific purposes and may, or may not work for other types of applications.
Despite the interest to this topic in the scientific world, it did not find a spread use in practice [Hong et al., 2009]. Not many context-aware systems can be found in use today, as well as there is not so many commercially released applications. One of the exception are the navigation or location-aware

applications (such as tour guides, navigators), which are widely presented on mobile application markets as well as on the web.

Such tendency could be justified by the lack of the standardized approaches and guidelines. Baldauf in his survey of context-aware systems highlighted the necessity of creating sophisticated approaches with standardized formats and protocols for building reusable and interoperable context-aware services. As an outcome the developed approaches will bring "new contextual knowledge or patterns" [Baldauf et al., 2007].

The majority of mobile and web applications are designed by independent developers or companies. The most of companies do not have dedicated personnel, who is responsible for user interface or user experience design, or research centers for new technologies investigation. Such tasks are mostly carried out, e.g., by graphical designers or software developers. Thus, this fact makes the guidelines, design tools, frameworks and patterns a theoretical base of the software development. For mobile application development the developer's guidelines or specifications, published by the *Operating System* (OS) vendors are the source of this knowledge and background (Android Developer Guide [AND, 2014], Android Design [ADG, 2014], iOS Human Interface Guidelines [IOS, 2014], User Experience Design Guidelines for Windows Phone [WIN, 2014], etc.) or web communities (World Wide Web Consortium [W3, 2014], WHATWG [WHA, 2014]). None of those documents contain recommendations about the utilization of context-awareness or system adaptation techniques. The only direction of interface adaptation is mentioned in the listed above documents is *UI* adaptation for multi-resolution mobile devices. Little to nothing has been done on designing adaptive context based user interfaces for mobile applications based on certain complex requirements. Few common and well-established mobile activities are already adaptive and context aware[1], which are mostly even dictated by the *OS*. There are also numerous frameworks, tools, patterns and components libraries, which are supposed to aid with adaptive *UI* design. However, these frameworks have

[1]Example: the algorithm of the soft keyboard appearance

not found a wide acceptance in mobile development, since none of them has been standardized by the OS vendors or by the web consortium. And only a few of components and approaches have been approved and added to the guidelines by the *OS* vendors[2]. These little steps into the adoption of adaptive techniques are mostly adaptation of interface elements according to the current state of the application. There is no significant progress in the direction of context-awareness either. For this reason, context-aware adaptation and yet many other possible interactions are left for application developers' discretion.

In addition to the forming lack of guidelines in terms of context-awareness and adaptation techniques, there is also the problem of imperfect or too complicated operating system *APIs*. That makes it quite hard to implement proper context-aware *UI* adaptation. This fact makes capabilities of the mobile phones underused. In addition, the *OSes* do not serve users intuitively. This gap in software development is attempted to fill by some commercial applications, such as *"Atooma"* [ATO] and *"Situations"* [SIT]. These software solutions assist to adjust the mobile phone functionality depending on the user's location or activity or period of time (e.g. context). For instance, the following actions can be set to execute automatically by the applications: switching to the silent mode during meetings, have Music player opened when headphones are connected, extend phone's battery life when mobile phone is not in use, notification about someone's activity via *SMS* (Short Message Service)[3].

Such lack of the guidelines has to be filled. In case of native applications this task should be solved by the *OS* vendors by extending guidelines and creating frameworks for acquiring context data from available sensors. These frameworks have to be aimed to transform raw sensors data into a structured form. For web applications, the specifications for adaptive user interface techniques usage have to be published by the web consortium.

[2]Example: Navigation Drawer component was included into latest Android support library.

[3]Examples were taken from the applications presentations

This Thesis focuses on the topic of adaptive user interfaces for mobile devices, particularly, on the research of possible benefits for *AUI* from the mobility and context-awareness properties of handheld devices. This Thesis attempts to design guidelines for *AUI* creation for mobile application. The result of this work possibly might be used to influence further development adaptive user interface techniques into guidelines and specification.

1.2 Research Objectives, Scope and Limitations

The main objective of the Thesis is to design guideline rules and associated patterns as an attempt to create a standardized approach to context-aware application design. The result documentation might be used as extension to existing guidelines as a reference for application developers and possibly *UI* designers for adaptive interfaces creation on mobile applications. In order to achieve the main objective, the following goals should be accomplished:

- **to design** an approach to adaptive user interface design;
  The first objective includes creation of the guidelines or rules for prototyping mobile applications, and associated patterns for applying adaptive techniques for mobile applications in practice;
- **to model** an application with respect to the formulated adaptive user interface guidelines and patterns;
  The second objective of the Thesis is to exploit the proposed approach to designing a new application. In addition, to illustrate a usage of the guidelines, these rules will be applied to published application with an existing user base;
- **to validate** the results of the created and improved applications;
  The third objective is to illustrate the benefits or drawbacks of the

designed approach. In order to determine the efficiency of the proposed approach, the user retention of the modified version will be measured.

- **to demonstrate** that guidelines for adaptive techniques based on the device context-awareness can be applied in practice for prototyping mobile applications;

Due to limited resources of the project some restrictions will be applied on the scope of this work. The complete and comprehensive validation of the guidelines application results can not be fitted into the scope of this work. This task can be covered by the future work.
Nevertheless, due to limited resources, another validation will be used in this work: an evaluation of the modernized published application. The user retention is used as the factor of successfulness of patterns usage and possibility to determine the efficiency of the proposed approach. We will have to keep in mind that such approach can not guarantee the accurate results of the validation, due to various factors which may affect on the user value variation.

1.3 Research Questions

The following research questions were raised to understand the advantages of the adaptive user interfaces techniques for mobile application development:

Question 1: How can adaptive user interface techniques be used for interface design and prototyping of mobile applications?

This is the key research question and it forms a basis for the rest of the research. In order to fully disclose the main question of the research the following sub-questions have to be considered first:

1. What questions to ask before making decision to use adaptive user interface for mobile application design?
2. What kind of limitation problems can be addressed by adaptivity techniques?
3. How can adaptivity processes benefit from specifics of mobile devices?
4. What kind of mobile device characteristics have to be taken into consideration for a successful adaptation?
5. How can mobile devices specifics increase applications usability?

Question 2: What are the existing approaches to adaptive user interface development for mobile devices?

This question can be divided into following sub-questions:

1. What kind of approaches for adaptive user interface design exist: guidelines, patterns, component collection?
2. What are the pros and cons of these solutions?
3. Can any of them be used as standardized solution?

Question 3: Can a standardized approach to *AUI* design for mobile applications be developed?

Can such approach be used and successfully implemented for designing AUI for mobile applications?

The research of this work reviews different types of adaptation, different approaches to acquiring contextual data. As a practical part of the Thesis is the creation of patterns for development of context-aware application for mobile devices.

1.4 Structure of the Thesis

This is organized as follows. First part of the work consists of the theoretical base of the Thesis. Chapter 2 presents the definitions of the key elements of the *AUI* design. It is extended by the review of existing implementations of the *AUI* in mobile application development with the examples of *AUI* usage. Following Chapter 3 contains the definition of context-awareness as one of the mobile devices property and reasons for *UI* adaptation. Later it defines the key elements of context-awareness.
Chapter 4 presents the research of the previous attempts of standardized approach to *AUI* design creation. Moreover, this chapter includes proposal for *AUI* design guidelines for mobile applications. As the proof-of-idea an associated with each guideline rule pattern is presented.
The application of the designed guidelines in the practice is presented in Chapter 6 as proof-of-workability and effectiveness of the proposed solution to *AUI* design of mobile applications. Finally, Chapter 7, Chapter 8 and Chapter 9 present evaluation of the designed guidelines, conclusions and future work.

Chapter 2

AUI in Mobile Application Design

This chapter introduces a theoretical basis for the key elements of the *Adaptive User Interface* (AUI) development. It lists the most common reasons for *AUI* usage and particularly focuses on implementing AUI in mobile applications with examples of *AUI* use.

2.1 Adaptive User Interface

Adaptive User Interface is interface which adapts its elements to the needs of individual user. The examples of adaptive interfaces on mobile devices are: navigator which indicates a user current position and calculates the path to destination, expandable search bar, soft keyboard, etc. *Adaptive User Interface* has taken a big role in the *Human-Computer Interaction* research. Recently this research is gaining a new wave of popularity. This interest is mostly stimulated by technological evolution: the growth of the application complexity and computer systems capacity. Moreover, computers, handheld and mobile devices are more accessible to people and becoming a part of everyday life. This brings a new challenge to *User Interface* (UI) designers: the

UI has to be able to accommodate a wide range of potential users. Often, applications are built for a variety of use cases and the UI has to be flexible to satisfy different users and various tasks.
In addition, interactions between the user and the device interface have to be simple and clear for users independently of their technical background, education or computer system knowledge. Application designers also have to cope with the fact that all users are different and each of them has unique personal preferences. User's preferences, mood, characteristics, etc. have also to be taken into consideration while designing the application user interface. Besides, the increased complexity of computer systems makes this task even more challenging. At the same time such progress creates potential opportunities for a big amount of improvements. The wide variety of sensors and the big capacity of computer hardware is a good base for creating rich and flexible interfaces, which will serve user in a better and more organic for individuals' way.
Modern applications have to operate on different platforms, run on different devices with different capabilities and used in different contexts, to serve different users [López-Jaquero and Montero, 2003]. The idea of *Adaptive User Interface* is aimed to automatically handle distinctions of such kind. A flexible or adaptive application can interact with the user in a personalized manner and in addition solve the differentiation problems described above.
In this section a definition of the *Adaptive User Interface* term is given, examples and potential domains of use of such type of *UI* provided, outlining the reasons for adapting user interface.

2.1.1 AUI definition

AUIs are systems that change their structure, contents and elements according to the need and context of the user [Schneider-Hufschmidt et al., 1993]. Schneider-Hufschmidt gave a complete definition of a new type of user interface, complemented and summed up the one given by Browne, Totterdel, and

Norman in 1990 [Browne et al., 1990]. Jameson described user-adaptive system as "an interactive system that adapts its behaviour to individual users on the basis of processes of user model acquisition and application that involve some form of learning, inference, or decision making" [Jameson, 2009]. Consequently, *Adaptive user interface* is a type of interactive system, which adapts its interface elements and behavior. As the basis of such transformation if the information acquired about its users, the context of use and its environment is used. Such systems are able to change their characteristics, settings, elements automatically according to user's needs, goals and interaction with the system. In other words, *AUI* allows user to create a flexible interface not only on the stage of its design, but also during usage. It makes the user more independent of the designer, and does not force the designer to decide about user-specific optima before the user works with the system [Oppermann, 1994]. Thus, adaptive systems not only can adapt to meet user's needs at particular moment, but also may anticipate the future steps and requirements.

Interface adaptation has been implemented in a wide range of interfaces and systems including web, mobile, desktop, and many other platforms. The next subsection provides a set of use cases of *AUI* as an illustration of the popularity and prevalence of this type of interface design.

2.1.2 Adaptive User Interface Applications

This subsection presents the possible domains of implementation of UI adaptation, with some commercially developed use cases as well as with research prototypes. The categorization of use cases presented below is nominal and is done by diving into usage domains. Concurrently, it is not strict and can be expanded by other examples. Moreover, many use cases may intersect and can be presented in the same user interface. The concepts, definitions and ideas in this section are mostly based on the Jameson's paper "Adaptive Interfaces and Agents" [Jameson, 2009] with some extension. The author

covers an extensive range of adaptive systems and classifies them by the main functional assistance they aim to provide. In the presented set of applications the most obvious and outdated cases are omitted and it is extended by the most relevant and currently actual cases as of the time of this work.

Personal Assistant System

One of the most promising and rapidly developing domain at the moment, is the **personal assistance system**. Such an adaptive system analyses user's related data, her behavior and environment conditions. Based on the results of such analysis, it predicts the next or current user's activity and suggests relevant information or functionality.
A good example of such system is *Google Now* [GNO, 2014], which has been recently released. This application runs in the background of the user mobile phone. In case relevant information can be retrieved related to the user's activity, e.g. location and other type of personal data at a time, it offers the right information and functionality in the form of a card. An example of such card can be found on Figure 2.1 below. In the figure there is a card which might be shown in case if the user has booked a flight. The application reminds user about the amount of minutes remained for leaving in order to be in time for the flight. In addition, it offers the functionality and information related to the current task. This data is a result of the interface adaptation based on user's goals or tasks. In the case described above, the interface will present elements for such actions like route calculation to airport, flight date and time, terminal and gate information. By reaching the destination the system will offer current weather forecast and local rates, news [GNO, 2014].

Personalization of Content

Content personalization is an acquisition of information with respect to the user's individual interests. This technique is widely used in news pub-

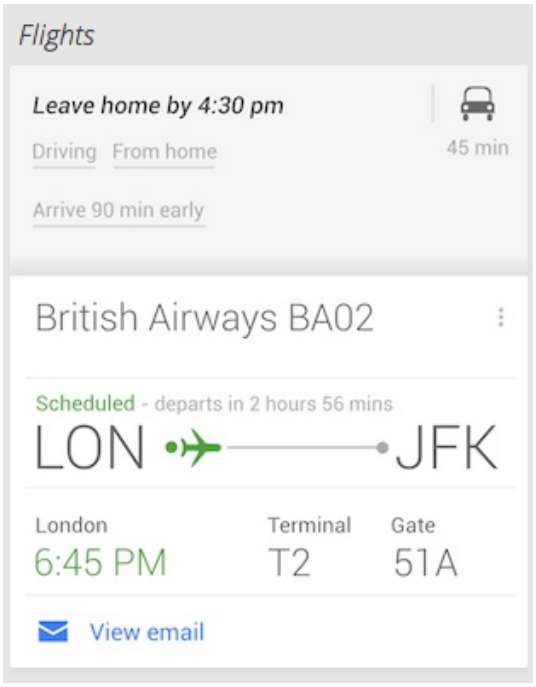

Figure 2.1: Example of Flights Google Now Card [GNO, 2014]

lishing systems, commercial web sites and applications. The general idea of this *AUI* use case is to make content filtering based on the user's behavior or preferences. This process of learning and/or inference is relying on the basic information about the user: (*User Model*): user-personalized knowledge base. Content filtering is applied according to the *user model*. The system analyses the user's activity, preferences, location, connections, browsing/listening/watching history, etc. Based on the results of such analysis it filters the content of the application. Examples of such content may be featured products, hypermedia, cinema, articles, music tracks or video clips recommendations or suggestions. Such logic personalizes the content for particular user depending on his user model, and the application performance is therefore different for different users.

The described use case can be illustrated with a list of the following examples: movie database *IMDB* [IMD], music website *last.fm* [LAS], video

sharing website *Youtube* [YOU], music streaming service *Spotify* [SPO], native application and website of *Helsinki Sanomat* [HS], adaptive news access [Billsus and Pazzani, 2007] and many others.
Content most relevant to a user is presented as "Recommended", e.g. video clips on Youtube, performer suggestions on Spotify and many other examples of content adaptation.
Personalized content adaptation can be based not only on the model of a single user, but also on using collaborative modeling. Such approach to adaptation is called *collaborative filtering*. The most well-known example is the purchase suggestions feature ("Customers Who Viewed This Item Also Viewed") in the *Amazon* [AMA] online shopping service. In addition to the user's activity on the site, Amazon system gains knowledge for adaptation from the comparison of the current user activity with activity of other site customers. The Amazon suggestion system takes into account numerous factors for content adaptation such as purchase history, user's rates and *likes* [AMA, 2014]. Another illustration of *collaborative filtering* is the "most" lists (such as "most viewed", "most popular", "most interesting" and others). That is, the system collects a history of all activity of the application users and this data is used for *UI* adaptation.

Adaptation of Interface Elements

In this type of *AUI* application user interface elements are conforming to fit better with the way a user exploits the system. The interface is adapting to serve to the user's needs, goals and current situation more efficiently. One of the most familiar example is the personalized menu, a feature of Microsoft Word 2000 [Gui, 2000]. The adaptive mechanism features only the menu items which are used most often and hide infrequently used menu options from the menu list. This logic makes the menu personalized. The algorithm determines the order of options in the menu according to their usage frequency: the more an option is used, the more accessible it is in the menu.

An option appears in the main part of the menu after it has been selected for the first time, and the order in the menu is updating permanently. In case an option was not in use for a period of time it will be moved to the hidden part of the menu (see Figure 2.2).

There are several other interface elements which can additionally illustrate

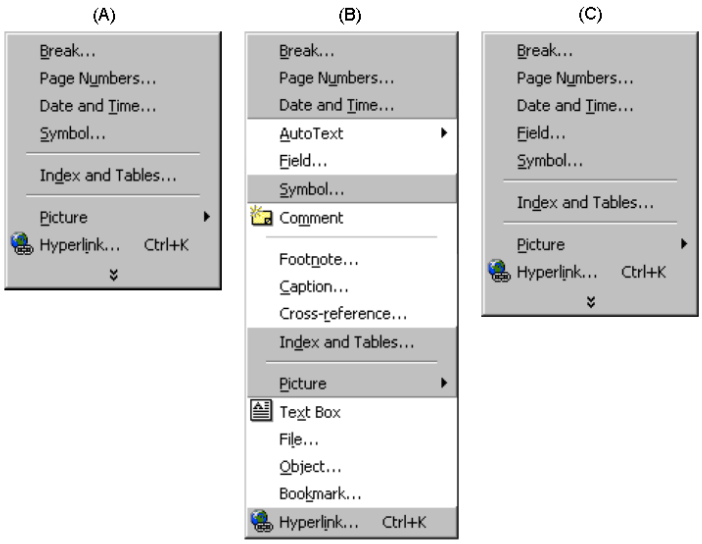

Figure 2.2: Example of adaptation in personalized menu. (The user accesses the "Insert" menu (A). Not finding the desired option, the user clicks on the extension arrows and selects the "Field" option (B). When the user later accesses the same menu, "Field" now appears in the main section (C).) [Jameson, 2009]

this case of interface adaptation: input fields' auto-completion, forms' auto-filling, context-sensitive menus, breadcrumbs and others.

Another direction of user interface adaptation is so called *Responsive Design*, one of the main concepts of which is called Adaptive Layout. This is an interface design approach that became extremely popular due to the variety of computer devices appearing on the market. Responsive design techniques provide an optimal content representation across the wide range of devices and screen sizes. In other words, in this approach the *UI* adapts depending

on the device screen dimensions in order to make it convenient to use on any kind of screen. Mainly this approach is used for web interface design, however, it is also applicable to any user interface. For instance, the adaptation of an application interface for landscape and portrait modes in mobile devices can be done with the same approach.

Adaptive Dialog System

These systems are adapting their content by involving some sort of artificial intelligence or heuristics in the process. An information/answer or a help article is offered based on the user's input in a separate text-based dialog inside the main application. Hereby, the system provides assistance to the user by displaying the data according to the user's questions. One of the first examples of such content adaptation was presented by Microsoft's Office Assistant. Nowadays, most of the help systems are built based on the same principles. The most recent and well-known example of a help assistant is the *Ikea web site* [IKE]: "Ask Anna" (see Figure 2.3). For the request "I need a bed", the assistant offers bed varieties available at the moment in the selected store, sorted by name, type, color etc. For more precise requests (like "I need black bed") the system will update the current page with product suggestions. There are several other examples of adaptive dialog systems, which were developed as proof-of-concept cases by researches: [Komatani et al., 2005], [Thompson and Goker, 2000], [Müller et al., 2003].

Multi-User Interface

In multi-user interface systems information presentation is tailored according to the user's role or their characteristics in the system. It is achieved by adapting content and functional controls according to the knowledge and domain knowledge of the user's group: admin, user, guests, etc. Software for healthcare centres and hospitals is an example one where the differentiation

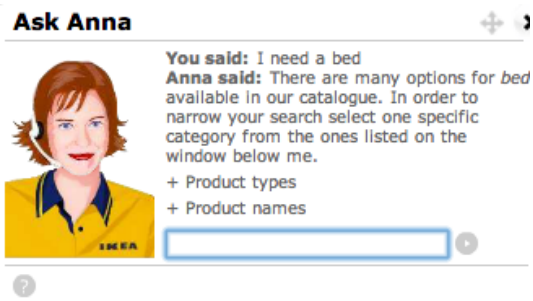

Figure 2.3: Help assistant on Ikea web site [IKE].

by role is highly important due to various reasons, e.g. taking into account patient privacy. Such collaborative system for handheld devices has been designed and presented by Muñoz and his colleagues[Muñoz et al., 2003]. The authors claimed that in health centres with different shifts the only certainty is about the role that person will have when attending patients. Thus, the system must be able to recognize roles as well as particular individuals. [Muñoz et al., 2003]

Systems Interacting with the Environment

The last use case of user interface adaptivity is the systems which change user interface depending on the surroundings and user/application interaction with it. The data collected about the environment is used as the *context* (see Section 3.2 for more details) for adaptation: user's location, temperature, noise level etc. Particularly location-sensitive systems will be reviewed, where the content and the user interface is altered depending on the user's current position and its related data (for instance, user's geographical history). The adaptation process might involve location-related suggestions as it was described in the *Personalization of content* case (see above in this subsection), or navigation support, etc. This adaptation is benefiting from the mobility of computer more than other types of adaptation described in this subsection.

A tour guide application is one example of this use case of adaptive user systems. A number of tourist guides were built and presented: "Cyberguide" by Abowd et al. [Abowd et al., 1997], "GUIDE" [Cheverst et al., 2000]. Moreover, it can be illustrated by numerous newer commercial implementations such as *Foursquare* [4SQ], *TRIPADVISOR* [TRI], *TouristEye* [TOU].
Moreover, most of the commercial applications and websites are aiming to provide location-based adaptation or integration of their services via *API*s. The following online services: *Facebook places* [FAC], *Twitter* [TWI] and others demonstrate this influence.
Another example of implementation of this use case was described by Schilit in his works about context-aware applications [Schilit, 1995]. The author presented an example, in which the location data is used in combination with other data as the context for determining the current user activity in order to switch a device to a proper mode: "meeting", "running", "at school" etc. Google has advanced even more in this direction, and currently the *Google Maps mobile application* [GMA] can detect not only user location in terms of address, but also their indoor positioning including floor detection, and can guide users inside a building.

2.1.3 Inefficient Adaptation Cases

Despite the popularity and wide implementation of adaptive systems it is not a panacea to solving all the user interface challenges and goals. In this section we demonstrate an example of adaptivity, which were unsuccessful or where adaptivity worsens the final version of the product.
McGrenere in his research compared the personalized menu of MS Word 2000 with traditional static menus and MSWord PERSONAL [McGrenere et al., 2002]. The authors presented another approach of filtering options in a menu. MSWord PERSONAL is the result of such approach implementation and an example of an *adaptive* system. In this variant of the menu the user can control the importance, jointly the visibility and accessibility, of an option

in a menu.
In the menu solution presented by Microsoft there is no control over the adaptation algorithm. Users can not view or change the underlying user model maintained by the system; their only control is to turn the adaptive menus on and off, and to reset the data collected in the user model [McGrenere et al., 2002]. This limitation induced a high level of user's discontent by the new feature. Moreover, continuous dislocation of options in the menu, can easily confuse users and make them lose their focus while using the application. A test group preferred the solution proposed by McGrenere over the MSWord PERSONAL. The author indicated as the result of their research the necessity to keep balance between adaptivity and adaptability: potential user control of the adaptation process.

2.1.4 Stages and Agents in the Adaptation process

Before selecting adaptation as the main technique for building a user interface, a set of fundamental questions be answered [Dieterich et al., 1993]:

- **who** should adapt?
  what side (user or system) of the user computer interaction model has to take the main role in the adaptation process,
- **what level of interaction** should be adapted?
  what part of the system has to be adapted: the presentation or the functionality.
- **what data** (information) has to be taken into consideration for applying adaptation?
- **what goals** should be promoted by the selected level of adaptation and based on what context?
  what is the aim of the application, can it be achieved with adaptation techniques and with available resources (context, technical characteristics, etc.).

- **when** should be the changes made?
  what the triggers are for adaptation.

Finally, as the result of the questions listed above, the decision about type of changes for adaptive user interface creation has to be done.
One of the main questions of the adaptation process is "**who** should adapt?". The question of allocating control over the adaptation process has been investigated by numerous researches ([Lavie and Meyer, 2010], [Dieterich et al., 1993], [Kobsa et al., 2001]). According to these research, AUI maintaining is not boolean, it can not be viewed as having or not having adaptive features. Instead, the adaptation principles are employed with different levels of adaptivity. Levels of adaptivity reflect the distribution of control over adaptation between the user and the system: is the system solely controls adaptation or has a co-operative process with the user.
At the same time, the tasks of adaptation process can be grouped by the stages of adaptation. Dieterich and his colleagues suggested the stages of adaptation from the user's point of view. These stages, based on the authors' position, are the ones to be considered for examining the adaptation process and exploring levels of adaptation (see Figure 2.4). The first stage is *initiative*, the decision regarding adaptation suggestion is made by one of the *agents* (user or system, as the most interesting agents are the ones who interact). Following by the *proposal* stage, the alternatives for adaptation have to be proposed by an agent. In the sequent stages one of the alternatives is chosen (*decision*) and finally executed (*execution*).
Hence according to the authors' vision of the topic there are two agents with four possible stages (16 possible cases in total).
Dieterich suggested a matrix (see Figure 2.4) as a visualisation representation of any possible case and classification of an approach or prototype.
Nevertheless, some combinations may be excluded from the review as non-perspective. For instance, the case when adaptation was selected or proposed by the system, but is actually made by the user. Just the same, the case when the user tailors the system to his/her own preferences, called *Simple*

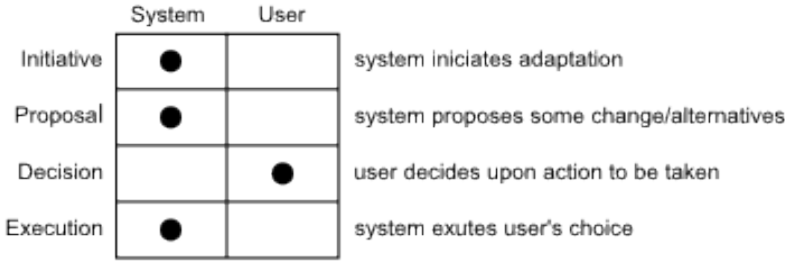

Figure 2.4: Task and Agents: Example of Configuration [Dieterich et al., 1993]

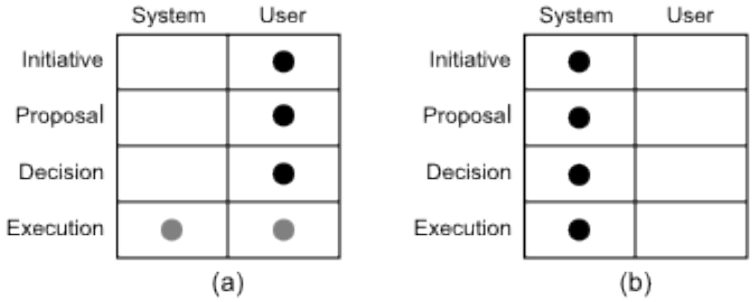

Figure 2.5: Examples of Task and Agents Configuration Scheme: Simple (a) and Self-Adaptation (b) [Dieterich et al., 1993]

Adaptation (see Figure 2.5 (a)). Color schemes and other user-controlled settings are a good illustration of this case. The contrary case is *Self-Adaptation* (see Figure 2.5 (b)). The system executes the adaptation tasks on all stages: observes the situation, makes a decision about possible adaptation, evaluates different variants and finally performs the most appropriate one. According to the analysis made by the authors of the paper, the approaches which give more control over the adaptation are more promising.

Flexibility or adaptation helps people to use systems more efficiently. Adaptation of the user interface may constantly improve interaction between the

user and the system. AUIs are situated and aim to enable the user to become smarter and more empowered [Jameson, 2009].

2.2 Mobile World

In the last few years the usage of mobile devices is growing exponentially. By the end of 2013, the smartphone user base gets at 22% of global population (or about 1.6 billion devices around a world), or closely every fourth human is a user of a smartphone. This is an 18% increase in four years [Heggustuen, 2013]. In developed countries the number of smartphone users reaches up to *50%* [Smith, 2012]. Moreover, the remarkable tablet penetration has exploded from 2% to 6% in last few years. On top of this, the *25%* [Cooper, 2013] of the owners of smartphones use them constantly not only as a phone, but also as tool to perform workday tasks. No doubt, handheld devices are a part of everyday life for a significant number of people and this penetration of mobile devices is raising rapidly.

Such tendency in the mobile world declines the priority of desktop computers. The number of desktops sold is lagging behind in comparison to the handheld devices (by approximately 8% in total [Heggustuen, 2013]). This substitution of computers with mobile devices can simply be explained by the accessibility of the handheld devices, technical specifications and capabilities of these devices, which anticipate the user's requirements and can perform nearly most of the same tasks. Modern smartphones have better hardware characteristics than the first generation of personal computers. The mobile Internet and emerging technologies, such as near-filed communications, real-time location and a variety of embedded sensors transform a mobile device into a powerful tool. The small size and light weight of handheld devices make them more attractive for end-users. Amid of the discussed above facts the usability of the mobile devices becomes a critical factor of their usage.

This section contains an overview of mobile device characteristics and lists challenges of user interface development for mobile devices. As a conclusion the arguments why to use *AUI* as a possible solution to mobile devices limitations are presented.

2.2.1 Mobile Devices

A *Mobile device* or *handheld device* is a small, portable, wireless computing device. Weiss formulated three criteria for computer devices, by passing which a device can be qualified as a handheld device [Weiss, 2002].

1. to be a wireless device - it must operate without cables, except temporarily, while recharging or synchronizing;
2. to be a portable device - it must be easily used while in one's hands, not resting on a table;
3. it must either allow the addition of new applications or support Internet connectivity.

One of the most important property of mobile devices is that they can be easily relocated due to the possibility to function without cables and the small size. Therefore, mobile or handheld devices can be classified as fully mobile in the mobile spectrum. Currently, the most of the handheld devices have a display screen with touch input and are operated by finger, stick and virtual or hard keyboard, but this is not an indicator of device being handheld or mobile.
According to the Weiss criteria such devices as tablets (some example, iPad from Apple, Galaxy Tab or Note from Samsung, Kindle Fire from Amazon, etc.) are excluded from the definition of mobile devices "simply because of [their] size" [Weiss, 2002]. However, in the scope of this Thesis tablet devices will be considered as handheld devices. This association is due to their mode of interaction, which is closest to that of handheld devices, as it was suggested by Gorlenko [Gorlenko and Merrick, 2003].
Thus in this work by using the term *mobile* or *handheld* device a portable, small and wireless device will be implied. Examples of such devices are mobile phones, tablets, similar to mobile phone devices, but without a mobile subscription (for instance iPod touch, a media player from Apple). By using the expression *mobile application* the author of this thesis will refer to appli-

cation which is running on a mobile device.

2.2.2 Mobile Devices Limitations

Despite the obvious advantages of mobile devices there are several limitations, which make interface development more challenging. In this subsection a list of mobile devices specifics, which make the interface development more complex, will be provided. The categorization of usability issues of mobile devices and applications, presented by Gorlenko [Gorlenko and Merrick, 2003], was taken as a basis and updated with reference to the Looije paper [Looije et al., 2007] and to correspond the current state.

- Technical - this category of issues refers to network connectivity, screen dimensions, input methods and battery life of a mobile device.
  The screen size limitation causes the need of fitting the same functionality as for desktop on a significantly smaller screen and the same set of actions in order to present the similar software capabilities. Moreover, the same functionality has to be equally performed in spite of the wide variety of mobile device sizes. This problem is further complicated by the necessity to enlarge interface elements for touch devices in order to make them comfortable to use by finger.

- Environmental - this category of issues refers to data about surroundings collected by sensors, constraints for user (for example, cognitive or psychological), user's mobility and goals. Examples of the data gathered by sensors are such physical environment measurements as temperature, light conditions, noise, distraction, location etc. More about information this can be found in Section 3.3.

- Social - this category of issues refers to the usability issues relating to privacy, acceptance, adoption, comfort and personalization.

Summing up, the listed above usability limitations of handheld devices have to be taken into consideration for application interface design. Unquestionably the distinction between mobile devices and desktop computers is influencing design process as technical as the interface wise.

2.2.3 Answering AUI Base Questions for Mobile Application Development

In order to evaluate *AUI* as a technique for mobile application development, the author of this Thesis will answer the main questions of the adaptive interface design (discussed in Section 2.1.4) from the point of view of mobile application development. Nevertheless, some of the questions will be omitted, because they must be considered for each application design individually and can not be generalized.

Who should adapt?

As it was concluded in Section 2.1.4 the only two agents can play the significant role in the adaptation process: the system and the user. In the case of adaptive user interface design for a mobile application the adaptation of the system has to be moved to the system side as much as possible with respect to the transparency of the adaptation process. In other words, the mobile application's tendency is to be *pervasive* or *ubiquitous*. Thereby the system or application has to manage information in the way to reduce complexity and aiming to the invisibility of the computing system itself [Weiser, 1991].

What to adapt?

Three general classes of adaptation which are suitable for mobile applica-

tions can be identified [Cena et al., 2006]:

1. *Information* - includes the adaptation of the application content and its layout and organization;
2. *Visualization* - includes the way information is presented;
3. *User Interface* - includes the adaptation of user interface controls and interaction between the user and the system.

In this work the term *user interface* will be referred to all classes of adaptation described above.

What data to use as the base for adaptation?

On handheld devices all available context has to be taken into consideration for adaptation processes. Brusilovsky pointed that the environment has to be taken into account as well as the user's characteristics [Brusilovsky, 2001]. In case of mobile based systems this statement is valid especially for other systems, since mobile devices have a huge variety of sensors, which can be utilized as environment meters. Moreover, handheld devices are often found in conditions of diverse surrounding due to their mobility.

What goals should be promoted?

The main goal of the user interface adaptation for mobile applications is to create a clear, straightforward and convenient user experience [Wesson et al., 2010]. In addition, the interface has to react to the changing environment and alleviate mobile device limitations [Hinckley et al., 2000].

Additionally the advantages of addressing the mobile application design to adaptive techniques was confirmed by several researches [Holzinger et al., 2012].

Chapter 3

Context-Awareness as Property of Mobile Devices

This chapter presents context-awareness as one of the properties of mobile devices and reasons for user interface adaptation. Later in this chapter the context as a base for adaptation of user interface will be reviewed. Finally, it depicts sensors as the primary source of context in mobile devices.

3.1 Context-Awareness

Mobile and handheld devices are used in a variety of surrounding situations due to their mobility. The meters installed in mobile devices allow system to determine the correlations of the environment in addition to system's state at the moment of time. It is possible to determine the current context in which the device is used. Whereas the *AUI's* intention is to provide an interface adapted according to user's needs and goals with minimizing user's input (as it was discussed in Section 2.1). Thereby the property of mobile devices of being *aware of the context* can be used to build adaptive interfaces having little or no user interference. This idea can be easily taken into use with current technologies, since the determination of knowledge about

environment and surroundings becomes more approachable.

3.1.1 Context-Awareness: Definition and History

Context-aware systems are commonly used in the wide range of domains (Subsection 2.1.2) such as user interface adaptation, tour guides, multi-user systems, targeting advertisement, smart environment. Moreover, such approach is gaining popularity in research and in the field of applied engineering year by year [Hong et al., 2009].
The predecessor of the concept of context-aware systems, the ubiquitous computing systems firstly appeared in the Mark Weiser article in 1991 [Weiser, 1991], more than 20 years ago counting from the time of this Thesis was in the process. The author described it as a computer property, which adapts its "behaviour in significant ways without requiring even a hint of artificial intelligence" based on the surroundings. From this notion the first discussion about context-aware mobile computing has arisen. Afterwards it was firstly mentioned by Schilit and Theimer in 1994 [Schilit and Theimer, 1994]. They summed up the knowledge of the term and gave the first complete definition of the context-aware computing notion:

"The context-aware computing is the ability of mobile user's applications to discover and react to changes in the environment they are situated in."

Despite references to other types of context in the Schilit's definition the researchers and developers were mostly focusing on location-aware systems at that period of time. Only location contextual information was used as the source for adaptation and application personalisation. For researchers and developers this limited the domain of possible context applications. Schmidt et al. pointed to the problem of the context definition being to narrow in his work "There is more to context than location" paper [Schmidt et al., 1999].
The unified definition of context-aware computing was offered by Dey, which covers a wider range of usage domains for the concept of context-awareness

in applications [Dey et al., 1999]:

"A system is context aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task."

The author gave a more generalized definition of the term, which does not cover the context based system adaptation. For instance, an application which just shows the context of user's environment (such as a weather forecast at user's location) will fit this definition of context-aware system, although it does not react to the context it has collected about the environment. To sum up, this definition covers the context-aware computing systems, but does not really suit as the definition of context-aware applications.

3.1.2 Context-Aware Applications

A software build with the property of context-awareness is called a context-aware application. Schilit et al. define context-aware applications as software being able to "adapt [itself] according to its location of use, the collection of nearby people and objects, as well as changes to those objects over time" [Schilit et al., 1994]. In other words, context-aware applications can recognize the environment dynamic characteristics and act upon them. Hence mobile devices can collect context data, process and afterwards make necessary adaptation of the system in order to serve the user in a more efficient way. The adaptation of the system can involve not only the user interface flexibility, but also changes in system behavior. Therefore a single application may look and serve the user differently depending on the collected context: hardware characteristics, environment, system state, user interaction with the system, etc.
In addition, the other goal of the context-aware adaptation is to provide the best user experience of the application with minimal user intervention. In brief, a context-aware application is aiming to take the maximum possible

amount of work from user's side. And as a result, to make the use of device, technologies and systems more comfortable and easier. "Context-aware systems are able to adapt their operations to the current context without explicit user intervention and thus aim at increasing usability and effectiveness by taking environmental context into account" [Baldauf et al., 2007].
For instance, if we will take a close look at a *navigation application*. This is a context-aware application, which calculates a route and directs the user to the destination point. The user location as contextual data on the applications running on a mobile device can be readily retrieved with the *global positioning system* (GPS). Hence users are freed from manually pointing their position on a map or entering an address. So the user is exempted from the hand-operated work of defining the context. In other words, the context-aware technology "takes into account the human world and allows the computers themselves to vanish into the background" [Weiser, 1991].
Some illustrations of context-aware applications can be found in Subsection 2.1.2, particularly the *Systems Interacted with Environment* case. The applications described and discussed there have an adaptive interface depending on the changes in the environment they are in and interactions with it. In addition the result of Wai Yip Lum work is a successful example of a context-aware application. Lum et al. designed a content adaptation system for hypermedia [Lum and Lau, 2002]. A part of the system-decision engine is an adaptation based on device capabilities. The context is chosen based on the battery level, dimensions, Internet connection and other device parameters and characteristics. The decision engine provides an optimal version of the hypermedia content, which is renderable for particular client device and suitable for its network characteristics. This decision is derived by executing an algorithm based on a received from the device context data. This approach was widely taken into use as a server-side application for web mobile application generation. Currently, newer techniques, such as responsive design, and increased capabilities of the mobile devices made such idea irrelevant.

3.1.3 AUI for Mobile Devices

As it was discussed in the Section 2.2 mobile devices gained high popularity nowadays. Despite that fact, there are still many unsolved usability problems in mobile application interface development. The mobile device limitations make the interface design for mobile applications even more challenging (see Section 2.2.2 for more details).

At the same time, mobile devices are possessing the property of context-awareness, which is originated from the their very property being mobile. "Context-awareness is especially interesting for mobile devices where the context of the application is highly dynamic allowing the application to deal with the constraints of mobile devices in terms of presentation and interaction abilities and communication restrictions" [Hofer and Schwinger, 2003]. In other words, mobile devices or handheld devices have a wide basis for getting the context due to the variety of sensors or measures, which can be utilized as environment meters. Hence this feature can make a significant impact on the application adaptation on mobile devices and make mobile devices a perfect platform for implementing adaptive techniques. Due to rapid technology development and mobile devices expansion this approach becomes extremely interesting.

The context-awareness as benefit of mobile devices can not only solve their limitation problems, but also can improve the application usability. This statement can be confirmed not only by the increasing popularity of adaptation techniques on mobile devices [Kane, 2010], but also by numerous of researches. Schneider-Hufschmidt in 1993 year identified that changing the system environment or some system features depending on what the user works at, is the reason for application adaptive behavior [Schneider-Hufschmidt et al., 1993]. More specifically, the adaptation techniques were mentioned as a way to enhance the usability of mobile applications in [Al-bar and Wakeman, 2001], [Billsus et al., 2002] and [Wesson et al., 2010]. The adaptive

user interfaces have been promoted as a possible solution for such usability issues as filtering and information overload, automation of task completion and learning to use complex systems [Höök, 1998].

The primary mobile devices' limitations are related to input-output capabilities: in most cases there are no habitual input devices, neither hard keyboard nor mouse, the output area is limited by the small screen size, etc. According to Schimdt, a set of enhancements in this direction can be applied on mobile devices based on situational context [Schmidt, 2000]. The output data presentation can be adapted to the current situation. For instance, such properties as font size, speakers' volume, brightness of the screen, privacy settings, etc can be changed according to the surrounding environment.

Moreover, interruptions can be served in a smarter way: the right time for a notification can be chosen, or it may be skipped completely in case it is not actual or important any more. For example, there is no need to remind someone to go to a meeting if he is already there. On the contrary, it might be appropriate to give a hint about the special meal once the user comes in to a restaurant, at the same time highlighting the possible preferable choices based on time of a day, user taste preferences, etc.

The input interface of a mobile application can also be adapted to the current situation [Schmidt, 2000]. Such features as audio filtering, speech recognition algorithms, etc. can be applied based on the surroundings. The input elements of the user interface can be reduced in order to provide options only appropriate in the current context (e.g. showing a list of available sharing options based on installed applications, or giving a list of restaurants for selection based on the user location, etc.). In some cases, the context can be automatically captured, so there would not be any need to input it explicitly.

3.2 Context is Key

Context-awareness is a property of computing systems to be aware of the context. Hence, the ideology of context-awareness is firmly depended a proper understanding of the context term. Thereby the context term is the key to context-awareness and subsequently the base for context-aware application adaptation. A deficient definition of the notion of context limits the advancement of context-aware applications development. For example, as it was discussed in Subsection 3.1.2, Schmidt has pointed out the limitation of context definition [Schmidt et al., 1999] in his work "There is more to context than location". Before that work researches were mainly focusing on location as the only relevant contextual element, but afterwards the variety and diversity of context-awareness applications have increased.
The knowledge of the context is considered as a source and motive for adaptation, so it is highly important to properly define the context notion in the scope of development of adaptive mobile applications. This context beneficial relation was noted by several researches[Schmidt, 2000], [Mäntyjärvi and Seppänen, 2003].

3.2.1 The Notion of Context

Even though the context term has been studied for many years, there is still no unified definition of the context as a part of human computer interactions. Schilit and Theimer firstly mention about term *context-aware*, in the same paper the authors defined context as a location, nearby people and objects identification and its changes [Schilit and Theimer, 1994]. Thereafter context defined as location and representation of nearby objects ([Brown et al., 1997], [Schilit et al., 1994]). Most of the early context definitions were synonymous of the situation or environment in which a computing system is found. These definitions did not cover many of possible cases and could not serve the growing needs of application developers. For instance, none of previous definition

comprised such data as the user's preferences or interests.
Dey and his colleagues summed up previous attempts of defining context and proposed a complete definition, which covers most of the possible aspects of data and implies the definitions given by other authors [Dey et al., 1999]:

"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."

Additionally, Courtaz defined the context not simply as a state but as a part of the process [Coutaz et al., 2005]. In other words, the context data has to be involved in human-computer interactions and be one of the representative of their result and status. So a system has to behave accordingly not only in given time and instant, but during the process in which users are involved.

3.2.2 Context Classification

As it was noted in Subsection 3.2.1 the definition of context is highly important for the context-aware application development. Context provides a structured and unified view of the world in which the system operates [Coutaz et al., 2005]. This orderly approach to context ensures a correct and complete usage of the context data. At the same time the context has to be categorized. Such systematic manner of context representation will help to comprehend and operate with it. Dey in his research introduced four principal categories of context information [Dey et al., 2001]:

- **Identity** - refers to the ability to assign an identifier to an entity. The identifier has to be unique in the application's namespace;
- **Location** - refers to the co-location, proximity or containment additionally to positioning information. All the positions in the space

related data (orientation, coordinates, name of the building or street, etc.) can be classified to this category;

- **Status** (or Activity) - identifies intrinsic characteristics of the entity that can be sensed. For instance, for a place it can be valued in temperature or noise level or luminosity. For a person, it can represent the physiological factors or mood or activity a person involved in;
- **Time** - refers to time component of entity description, helps to characterize the situation. The value can be presented as timestamp, or time span, indication of period, during which some other contextual information was detected or relevant;

The provided categorization is a modified context model proposed by Schmidt in 1999 year [Schmidt et al., 1999].
This information is aimed to characterize the current situation of an entity, which according to [Dey et al., 2001] can be a person, a place or an object. All of these entities can be plural: people, places (regions of geographical space: office, street, address, coordinates, etc.) and things (physical objects or software components and artifacts) respectively.
According to the classification presented by Dey, the correlation of some contextual information with categories was done and performed in the table (see Table 3.1). This table-based representation of context classification is constructed by the author of this Thesis. Here is an example with the following simulated situation: a student is on a meeting presenting results of his work in front of the audience.

Only contextual information about three entities has been taken to illustrate the principle of context categorization: the speaker (*person*), the auditorium where the presentation is going (*place*) and the presentation file (*object*). For the person entity in the *identity* category pointed his identifier (ID) in the university database; the address of the person's current site as *location*; his status as "meeting" represents the activity of this person; for the timing, the timeframe is indicated, since the meeting will last for two hours. A possible use case of this context data may be automatically setting the user's status

	Identity	**Location**	**Status**	**Time**
Person	64562299	Aalto University, Espoo	meeting	10:00-12:00 10.03.2014
Place	temperature	A2038	21°	10:00 10 of March
Object	present.pptp	60.184811, 24.829826	open	139443660000

Table 3.1: The table representation of context categorization. Situation: a student is on a meeting presenting the results of his work in front of the audience.

in the internal university system.
Of course this illustration is one of many possible cases of the same situation, even for the same entities. For the *time* category a timestamp can be used as an *object*, or latitude/longitude as location representation etc. So the format and performance of the data is not precise.
There is a number of other classifications of context information, but the categorization suggested by Prekop is considered more advanced and is used in the bounds of this paper [Prekop and Burnett, 2003]. The author suggested to categorize contextual data into two classes by the origin: *external* and *internal*. The *external* context dimensions retrieved from the surrounding environment and by using hardware sensors; on the other hand, the *internal* context dimensions are mostly specified by the user or can be gained by analyzing user's interactions with the software, i.e. user goals, tasks, business process, etc.

3.2.3 Adaptation to Context Cycle Process

The interface adaptation based on the available contextual information is structured as five steps process: system detection, system identification, sys-

tem selection, system transition, system execution. This breakdown of the adaptation process into five steps was proposed by Hanumansetty [Hanumansetty, 2004]:

- **System Detection** - Detection of the conditions for adaptation with references to available context;
- **System Identification** - Identification of the user interfaces appropriate to the detected context;
- **System Selection** - Selection of the user interface, which relies on a problem solving strategy. User assistance is accepted in this step;
- **System Transition** - Transition from the current user interface to the newly selected one;
- **System Execution** - Execution or performance of the selected user interface until next conditions of adaptation are met.

The five steps of adaptation process can be performed solely by the system, solely by the user or as a result of interactions between the user and the system.

3.2.4 Context-Aware Cycle Process

At the same time the data measured by sensors is processed in a context-aware application through three steps cycle. This context-aware circle has been proposed by Schilit [Schilit, 1995]:

- **Discovery** - Learning about available resources and their characteristics;
  The software has to define the available context, identify its characteristics and capabilities. All the correlations in computing the context, as well as new emerged resources are detected and processed accordingly by the system.

- **Selection** - Determination which resources to use;
  The system analyses available resources and finds out which of them can be used. The decision is concluded according to the software tasks and possible goals of a user.
- **Use** - Engaging resources in the software flow;
  Employing the selected resources might be done without involving the software.

A collection of the contextual information has to be well-defined for all the described above steps. The context is a representation of the surrounding state and system characteristics.

3.2.5 Implicit HCI in Context-Aware Application

Before processing contextual information, it has to be collected. The context can be gained implicitly or explicitly. In an implicit *Human Computer Interaction* (HCI) the context information is determined by monitoring the user-computer activity, on the contrary, to the traditional way of providing context data by the user explicitly.
The automatically collected contextual data reduces the explicit human-computer interaction, the interactions in which the activity of computer manipulated by the user (e.g. by using a command-line, direct manipulation using a *Graphical User Interface* (GUI), gestures, or speech input). Thereby, the ability to retrieve data automatically without the user's intention shifts the human-computer interactions from explicit towards to implicit [Schmidt, 2000]. The awareness of context, a property of mobile devices, provides the main component for implicit acquiring of context. Such implicit communications between the computer and the user make the application more attentive, serving the user's needs in a better way. In contrast to the explicit type of interaction, the implicit *HCI* endorse the idea of invisible, ubiquitous computing discussed in Section 3.1. Nevertheless, it is not an alternative

to the traditional user-operated definition of context, it just has a different area of application [Schmidt, 2013]. For instance, user location detection can be defined by the user manually or determined by the system or application automatically. The model on Figure 3.1 illustrates the explicit and implicit human-computer interactions

There are several ways to obtain contextual information implicitly. The list

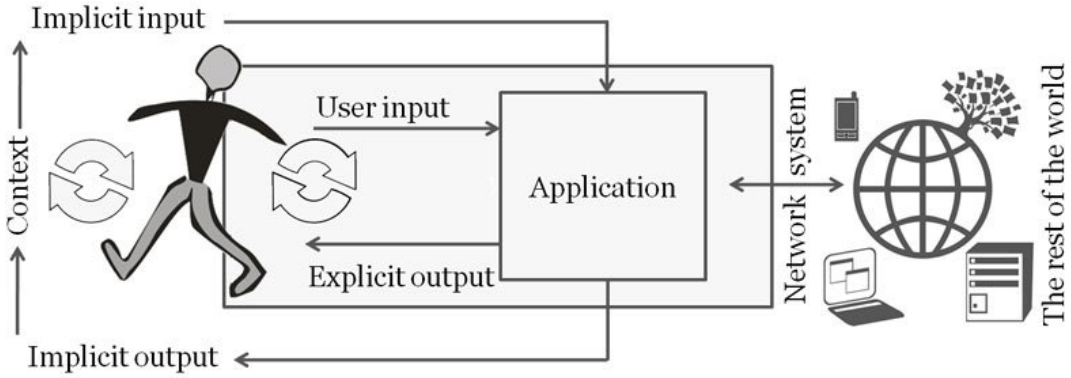

Figure 3.1: Explicit and implicit interactions scheme [Schmidt, 2013]

presented by Schmidt in his work [Schmidt, 2000] was taken as the basis and extended by some other cases and examples, which were missed from the list by the opinion of author of the Thesis:

- device-databases (e.g. calendar, TODO-lists, contact book, profile, *specific knowledge of the installed on the devices applications*, *operation system installed on the device*, *devices resources*, etc.);
- input to the application running (notepad - taking notes, calendar - looking up a date, etc.);
- sensing context using sensors (*accelerometer*, *thermometer*, *proximity sensors*, *Bluetooth*, GPS, etc.);
- active environments (IR-networks, etc.);
- databases located in the Internet [Mäntyjärvi and Seppänen, 2003];

The contextual data collected by a device in a certain period of time might be used for the adaptation on mobile devices. Therefore the obvious advantages of the implicit way of context determination, the balance between these two ways should be kept. Otherwise, user can be confused by the results of adaptation. The user must be aware of it and follow the logic, or it might bring lack of user's control.
The main source of the implicitly collected data on mobile devices is sensors. In the following section, the sensor term will be considered closer and the interrelation between sensing elements and context-aware system.

3.3 Sensors as Source of Context

The implicit way for obtaining of context information includes sensing the surrounding environment, system state, observation the interactions between system and user, etc. These actions can be represented via various sensors installed on a mobile devices [Mäntyjärvi and Seppänen, 2003]. The data collected by sensors provide knowledge about an illumination, temperature, noise level, device movements, etc. Afterwards, the row sensors data processing by the system and transferred into the comprehensible for future usage form (for instance, as the model presented in Subsection 3.2.2).
There is a summary of raw data and user input handling in Figure 3.2, which performs the vision of the adaptation to context cycle process (see Subsection 3.2.3) in adaptive systems by the author of Thesis. In most of the cases the term *'sensor'* implies an instrument, which is aimed to measure and convert a physical phenomenon into an electrical signal (notion of sensor given by Wilson [Wilson, 2004]). Such definition does not cover all the contextual data sources available on mobile devices. So in the scope of the context-awareness concept, an abstraction of sensor is used for a more general characterization. The classification of sensors offered by Indulska in [Indulska and Sutton, 2003] will help to understand the notion of sensor as

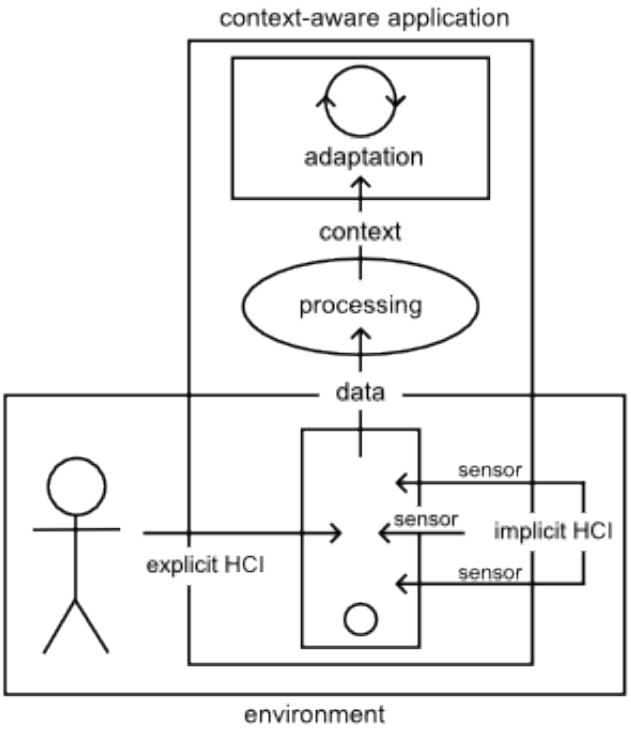

Figure 3.2: Data handling in context-aware application

a source of the context in context-aware systems. The author has presented the classification for the determination of location data, but it can be applied to any type of contextual information:

- physical sensors;
- virtual sensors;
- logical sensors;

The listed categories of sensors are reviewed more deeply below with examples and explanations.

Physical Sensors

This category of sensors refers to real devices which can capture physical

measurements. Physical sensors represent a part of the interface between the physical world and the electrical devices (computers, mobile devices, etc.). Due to the obvious benefits of physical sensors such as low-cost, small size, they are a part of most of the handheld hardware packages. A brief overview below lists some sensors with reference to mobile devices (based on [Schmidt and Laerhoven, 2001]).

Light Sensor: Single optical sensors (photodiode) supply information on the light intensity.

Camera: Using cameras, a wide range of information can be collected (e.g. prevalent colors, motion). By involving, additional algorithms additional data can be gained, for instance, such as color histogram, recognition of shapes and objects, faces and motion tracking.

Audio, Microphones: Microphones can provide not just voice input, but such data as noise level, type of input (noisy, music, speaking), and base frequency.

Accelerometer: Contexts like orientation or movement of the device can be defined by an accelerometer's data. For instance, such situation as having device stationary on a table, or the driving in a car, can be detected.

Location: Such sensors can determine position, location, collocation, and proximity of other users or devices. For outdoor position detection the Global Positioning System (GPS) is mostly used. The cellular network infrastructures such as the Global System for Mobile Communications (GSM) can provide location data as well. The indoor location sensors are typically embedded in the environment, as in the Active Badge system, [Want et al., 1992] or NFC.

Temperature: This sensor measures the temperature.

Touch: This type of sensors is a hardware characteristic of smart phones, tablet, or other handheld devices having display screens with touch input.

This is not a complete list of physical sensors, moreover with future hardware development it can be extended by other types of sensors.

Virtual Sensors

The source of data, collected by virtual sensors, has its origin in application data rather than physical devices and infrastructure. This category of sensors retrieves information by monitoring application data, operating system or network events: calendar, TODO lists, contact book, etc.
There is another meaning of the *virtual sensors* term, which means software abstraction or approximation of physical sensors, however in the scope of this work the definition given in [Indulska and Sutton, 2003] is used. The following situation will illustrate the approach utilizing virtual sensors: a user has scheduled an event in his calendar to a particular date and time. So the system can assume that at the specified time the user will be at the place associated with the event. Such algorithm determines possible user's location without practical measurements by physical sensors. In addition, the user's current activity can be recognized as being on an event, so this data might affect on switching device to the silent mode. The Google Now application can also illustrate utilization of virtual sensors (see Section 2.1.2)

Logical Sensors

Logical sensors gain context data by analyzing data from several sensors: physical or virtual or their combination. This process might involve computation, artificial intelligence or other type of logic, in order to retrieve the actual data.

The following situation can demonstrate a use case of logical sensors: user has scheduled several meetings to the same time. Moreover, the premises, where the meetings will be held, are at sufficient distances from each other. The available physical sensor can allocate user only with low proximity: hundreds of meters (e.g. mobile phone cells). None of these sensors can independently give accurate information about user's location. But the combination of data from physical sensor with the list of meeting locations from the a virtual sensor can give more exact information about user's whereabouts.

Chapter 4

Standardized Approaches to AUI Design

The previous chapters (Chapter 2 and Chapter 3) reviewed the key definitions of the Thesis with explanations of their interactions and roles in adaptivity processes. As it was shown, the idea of user interface adaptivity is powerful concept for user interfaces design. It especially benefits on mobile devices due to the property of context-awareness and especially suitable for them in assisting with overcoming mobile device limitations. However, as it was discussed in Subsection 2.1.3 some of user-oriented interface adaptivity implementations are not as successful as it is assumed on the designing stage. Straightforward adaptivity is not always the beneficial solution. This fact suggest the idea of guidelines or patterns creation potentially useful. A guideline may lead to standardizing the approach of *AUI* design and will help to avoid hidden pitfalls of the process. Despite of a major progress in the *AUI* research, there is still no methodology for determination when and how UI adaptivity should be implemented [Lavie and Meyer, 2010] as well as there is a lack of standardized approaches.

In this chapters the potential of the guidelines and patterns for adaptive user interface creation will be discussed more widely with review of the work related in *AUI* guidelines design.

4.1 Defining Key Concepts

Guideline

User Interface Guidelines are a set of documentation which presents recommendations for application interface development. Guidelines help to establish rules for coordinating individual developers and designers. Such approach facilitates the application development process by defining detailed requirements which eliminates reinventing them over and over again individually. In mobile application development, a guideline is applied to all embodiments and provides consistency across all products. This creates a portable branding experience and recognition appearance for mobile applications.
Guideline recommendations or rules in user interface application development aim to assist in several situations (based on [Smith and Mosier, 1986] and [Kotzé et al., 2006])[1]:

- for software developers as a starting point for development and a base of expert knowledge (Example: Android Developer Guide [AND, 2014]). The guideline documentation lists rules for desirable qualities of the internal structure and behavior of software, etc.;
- for user experience designers, The guideline can regulate the following aspects of application development: usability, behaviors, policies of human computer interactions, interface components, etc.;
- for user interface designers as a guide of look and feel and visual style of the potential application (Example: iOS Human Interface Guidelines [IOS, 2014]). The guideline may include set of restrictions, which the designer should follow to keep manufacture specific user interface style;

[1]The cases extended with examples of mobile application development area

- for managers, responsible for user interface software design, as a reason for more efficient design process planning.

The guideline may contain expertise of different approaches with different software/OS versions, device dimensions, use cases, etc., which are operable and actual for particular operation system, browser, service or technology.

Design Patterns

A *Design Pattern* is a general reusable solution to a particular problem within the scope of a given context (see Subsection 3.2.1 for context definition) [Bushmann et al., 1996]. A pattern usually have a form of description or template of a task solution, which can be used to solve recurring problem in several different situations. The following rule from the Firefox OS design guideline gives an example of UI design pattern: "Search areas are background sensitive and have been designed for implementation with both light and dark apps. They are placed just below the header." [Fir, 2014]. The visual component of search area pattern is presented on Figure 4.1.

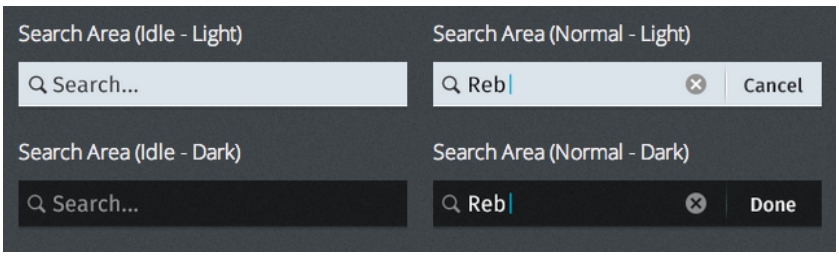

Figure 4.1: Visual component of the search area pattern for Mozilla OS[2].

Design patterns in guidelines development are distinguished by the following purposes [Hong et al., 2009]:

[2]The image is taken from Mozilla website [Fir, 2014]

- to help designers address high-level and low-level problems by indicating the hierarchical structure between design patterns;
- to capture the essence of recurring problems and their solutions in a compact form;
- to have many examples of actual designs, alternatives to apply the solution, and some of the trade-offs in applying the solution.

Whereas guidelines have been identified as a standardized approach for sharing design knowledge, patterns record the meta-information surrounding the simple imperative instruction generally found in guidelines [Griffiths and Pemberton, 2005]. Numerous researchers pointed out the importance of pattern-based design as a way of sharing solutions to context-aware computing design problems ([Chung et al., 2004], [Landay and Borriello, 2003]). Since the context-aware computing is on the nascent stages, development of pattern is the most promising approach at the moment ([Chung et al., 2004]). In the bound of this paper a pattern for each guideline rule is presented as a proof of applicability of a guideline principle. The figure 4.2 presents the dependency the ultimate result of the guidelines design based on a system existing problem. The higher level in the pyramid the more narrow case it is applicable to, but more variations of such design element there can exist. For example, several recommendations can be defined for the same *AUI* design issue, based on a guideline rule a list of patterns can be designed, each of which can be a base for creating a component.
In this Thesis the author introduces a guideline rule for each issue from a set of discussed in this section limitations of the *AUI* design for mobile applications. By following the recommendation of a guideline, a pattern for particular case is created and possibly a component as a visual solution for the user interface problem.

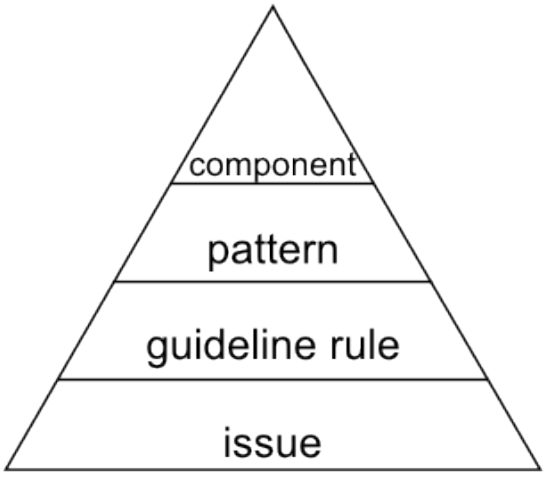

Figure 4.2: Scheme of the user interface design elements dependency

4.2 Related Work

In order to reduce the complexity of context-awareness support multiple frameworks ([Motti and Vanderdonckt, 2013], [Biegel and Cahill, 2004]), toolkits ([Dey et al., 2001]) and middleware ([David et al., 2011]) have been proposed. Most of existing approaches have adopted an infrastructure-centered view, presenting the ways for gathering, managing and distributing contextual information [Henricksen and Indulska, 2004]. A deeper analysis of the topic shows that, despite understanding the concept of context-awareness and handling the context in such systems, context-aware adaptation is result of an ad-hoc practice [Rossi et al., 2005]. So the engineering aspect still lacks of context-aware adaptation research.

On top of that, a majority of presented solutions tend to have distinguishing features or offers theoretical base and do not presume reusable mechanisms nor offer a standardized approach like guidelines or patterns. Despite a notable interest in the topic of user interface adaptation on mobile applications,

there is quite limited amount of guidelines or collection of patterns for creation of adaptive systems. Although, there are numerous publications and researches related to guidelines, context data management, and algorithm to develop context-aware application and systems [Hong et al., 2009]. But most of these works concentrate on some generic, superficial solutions, which do not consider mobile devices specifics ([Julien et al., 2004], [Anhalt et al., 2001], [Hong and Landay, 2001]), or in the opposite, focused on relatively narrow cases ([Avrahami et al., 2007], [Bolchini et al., 2007], [Casas and Cuartielles, 2007]). Research among context-awareness has so far been very much concentrating on system level architectures, context-recognition, or demonstrating application concepts [Mäntyjärvi, 2003].
Nevertheless, there were several attempts to produce guidelines and patterns for creation of context-aware application. Designing of patterns and guidelines by researches is quite independent and not always passes validation of the efficiency of the proposed solutions.
The rest section reviews existing guidelines and patterns for mobile applications design, concentrating on the ones, which are suitable for adaptive interfaces design.

Existing Guidelines

Currently, the field of guidelines for adaptive interfaces based on context-awareness is not well investigated. In the research on context-aware computing there are different design concerns, observed over the years by the researchers. The solutions to each of them can be considered as a guideline. The widgets presented in [Salber et al., 1999] can be examined as an abstraction of a guideline rule. So the solution of a problem presented as a widget in this paper can be represented in a way close to a guideline recommendation. For instance, the idea of distinguishing sensors from the rest of the software by inserting middleware or framework, can be re-written as particular reusable rule. Similarly, such abstract ideas can be discovered in other

works related to context-aware computing and be rephrased in the form of guideline rules. Nevertheless, only few research publications can be identified as not individual recommendation for a particular problem, but guidelines. This part of the Thesis presents the discovered guidelines for the adaptive user interface design.

Gong et al. presented guidelines for handheld mobile device interface design [Gong and Tarasewich, 2004]. The designed guidelines are modification of the traditional guidelines for desktop user interfaces for mobile devices, introduced by Shneiderman [Shneiderman, 1992]. In addition, the author extend the fundamental recommendations transformed for mobile devices by additional guidelines for mobile interface design. The handheld devices specifics are taken into consideration with complementary rules in order to solve some of the device limitations, such as "Design for Small Devices" and "Design for Limited and Split Attention" rules. Moreover, the authors appreciated the handheld devices specifics, such as mobility and context-awareness, as well as included up some context-based adaptation guidance. In spite of the work of guideline creation the authors omitted an evaluation of the proposed rules. This shifts the results of their work to theory.

The guidelines offered by Häkkilä and Mäntyjärvi pay significant attention to creations of rules for the human-computer interactions and usability issues [Häkkilä and Mäntyjärvi, 2006]. The designed guidelines do not present solutions for the handheld devices limitations and do not use the full capability of the device properties. A strong point of the presented by the authors' recommendations is the evaluation. The main objective of the validation was identification of usefulness of the proposed recommendations. The presented guidelines were evaluated with two separate design assignments, which were carried out in different circumstances. This approach increases the validity of the validation according to the authors' opinion.

The research showed that guidelines in general are useful, in spite of that some of the rules were intuitively used by the participants without references to the guideline and some were found very controversial and difficult for un-

derstanding and applying in practice. The guidelines were updated in the successively published paper [Dey and Häkkilä, 2008] taking into account all deficiencies by the test group.
In the opinion of the author of this Thesis the presented guidelines are quite limited and mostly concentrated on the usability issues, not the mobile devices specifics. In addition, some of the rules are repetitive and superficial. Nevertheless, the author of this Thesis takes the result of these research as a reference for creation of own guidelines, especially the most valuable is the feedback of the test groups. This decision is based on the uniqueness of the result of this research and it can be called the first and the only attempt of guidelines creation for context-aware mobile applications. This statement was also claimed by the authors [Häkkilä and Mäntyjärvi, 2006], but the in the opinion of the author of this Thesis, [Gong and Tarasewich, 2004] already contains some rules for context handling (For instance: "Design for multiple and dynamic contexts" rules). In other words, these two papers are considered as fundamental in terms of guidelines for adaptive interfaces for mobile device creation. In the bounds of this paper, these papers were taken as a blueprint of *AUI* guidelines for mobile application design.

Existing Patterns

Design patterns were identified as a potentially benefiting solution for ubiquitous computing (or context-aware computing) ([Chung et al., 2004], [Riva et al., 2006]). There were variety of attempts to create user interface patterns for mobile applications, even *UI* pattern libraries (e.g. Android Patterns [pat, 2014]). The majority of created patterns are oriented on the *GUI* perfection rather than on the functional part of the task. There are also solutions, which are aimed to solve the limitation of the mobile devices [Nilsson, 2009] or to resolve particular issues [Pearson and Shen, 2010] or particular task-oriented (For instance, patterns for building adaptive spoken dialogue systems [Jokinen and Rissanen, 2002]).

The quantity of the patterns, which serve user interface adaptivity interests or can take into account the context-awareness property of the mobile devices is quite small. Moreover, few are widely known and are included into specifications or guidelines. This part of the Thesis includes a result of investigations in the pattern design of context based adaptive user interfaces field.

There were numerous attempts to present the design experience in context-aware and mobile computing in a form of patterns. For example, Landay and Borriello performed different strategies for using ubiquitous computing in pattern form [Landay and Borriello, 2003]; Roth retrieved successful experience from mobile applications and presented in mobility patterns in [Roth, 2002].

The ideas presented in [Landay and Borriello, 2003] were extended beyond in [Chung et al., 2004], which produced a wide collection of patterns for context-aware computing. The final set of patterns was composed after several rounds of sorting based on the effectiveness evaluation, feedback and other criteria. The designed patterns cover the main issues of context-aware application development, such as interaction techniques with sensors and devices, policies and mechanisms for managing end-user privacy, ways to merge physical objects and spaces with virtual. Unfortunately, there is no possibility to investigate and decompose the whole pattern collection due to the fact of the unavailability of the online resource indicated in the paper. Nevertheless, if may be noted that the listed in the article patterns have quite narrow application and aimed to solve specific context-aware computing tasks (For instance, "Enabling Mobile Commerce", "Find a Friend" [Chung et al., 2004]). This restricts the re-usage of the result of work due to limited cover of the context-aware problems. In other words, it can be characterized as a collection of the solutions for particular tasks and problems in context-aware system design, but not as standardized solution. Even so, the patterns received positive feedback and evaluation from test groups.

There were attempts to create patterns based on previous experience in the

context aware adaptation field. Rossi et al. analyzed existing applications and on this base presented patterns for context-aware adaptation [Rossi et al., 2005]. The obtained patterns constitute a set of recommendations for the system adaptation based on the different cases: "typified context" (for instance, adaptation based on type of device); "active context element" (for instance, adaptation based on current location); "rule-based adaptation" (for instance, adaptation based on time intervals); "context wrapper" (for instance, adaptation based on context of use or user role). Another example of such approach of patterns creation is [Riva et al., 2006]. The authors provided a set of solutions for infrastructural issues to support context-awareness. The results of both works discussed above, were not evaluated nor tested, so they carry more theoretical knowledge of context-aware adaptation implementation on the designing stage. Concluding, the presented patterns have more heuristic approach to application structure design rather than concrete solutions and actions.

This section reviewed guidelines and patterns, which are the most relevant for the topic of the Thesis. The research and investigations in the filed of context based interface adaptation for mobile applications showed the deficiency of existing solutions. As it was pointed, only a few of the presented guidelines and pattern collections can be applicable on a generic level of the context adaptive system design and used as standardized recommendation or reusable templates. Whereas, the goal of this Thesis to create a standardized recommendation for designing context-aware applications with respect to the high-level issues.

4.3 Challenges of Context Adaptive Systems Development

In spite the obvious advantages of context adaptive application implementation on mobile devices, there are numerous of technical challenges that intricate design of such systems from a hardware and a software point of view. There are several origins of the impacts, which affect on the context adaptive mobile applications design: complexity of the adaptive systems design and context-awareness related issues. Adaptivity brings the issues with correctness, well-timed, transparent flexibility of interfaces and services. With context-awareness, the job of the user interface design becomes even more complex due to multiple situations and context [Schmidt, 2013]. Moreover, device mobility and device limitations has to be taken into account as a factor that affects the user interface design for mobile applications.
This section outlines the challenges that must be overcame before a context adaptive mobile application can be built. For each issue the origin domain is pointed: context-awareness or interface adaptivity or both.

1. **Sensor Engaging Issues** (Context-awareness)
   The process of gathering data from available sensors has to cover all available types of sensors (see Section 3.3 as type of sensors reference). This case includes such issues as integration of heterogeneous sensors [Riva et al., 2006], engagement of external sensing elements [Hong and Landay, 2001], awareness of the context variation, choice of the sensors to be used [Santos and Cardoso, 2010], etc.;
2. **Sensing Conflicts** (Context-awareness) [Hong et al., 2009]
   The results from different physical data sources might not match. For instance, if the coordinates of GPS and spotting of a camera are varied, then sensing conflict is generated;
3. **Context Handling Issues** (Context-awareness) [Hong and Landay, 2001]

This challenge is considerable and includes all the actions on the path from obtaining raw data from sensors to the context usage:

(a) *proceeding* and *representing* raw sensors data as contextual information (For instance, definition of the context processing components issue [Riva et al., 2006], management of context hierarchy [Santos and Cardoso, 2010]);

(b) context data *usage* (check Subsection 3.2.4), separation between application logic and context-inference code [Santos and Cardoso, 2010]);

In addition, this issue includes the necessity to handle situation when context data is not available at all. Such a case might appear due to privacy preferences of a user (a person does not want to share certain private information), or due to unavailable Internet connection or sensing elements, or limitation of an access rights, etc.;

4. **Privacy Issue** (Context-awareness, Interface adaptivity)
This issue is closely linked to the previous one as part of the context handling process and equivalently relevant for context-awareness and interfaces adaptivity. The system has to protect the collected data according to the user's privacy settings and preferences. Such contextual information as activity or current location have to be published with accuracy and respect to the user's privacy politics [Bobek et al., 2013];

5. **Defining Roles** (Context-awareness, Interface adaptivity)
The roles have to be defined for both processes:

(a) *adaptivity* [Dieterich et al., 1993] - what agent of *HCI* process has to take a main role in the adaptation process (see also Subsection 2.1.4)

(b) *context-awareness* [Hong and Landay, 2001] - what part of the system: devices or infrastructures or user, should generate the data obtaining: the right balance of responsibilities between key

elements of the system has to be found;

6. **Device Related Issues**
   Mobile device interface design is more restrictive than desktop interface design due to the following factors:

   (a) *hardware specifics* - relatively limited computing and communication power, smaller platform sizes [Pauty et al., 2006] (see Subsection 2.2.2 for complete list of mobile device limitations), unstable and dynamic network and accessibility to sensors [Bellavista and Corradi, 2012], variety of mobile devices (This includes the abundance of *OS*, screen dimensions, etc.);

   (b) *user specifics* - smaller amount of user attention [Gorlenko and Merrick, 2003], time as more critical factor for a mobile device user this have to affect of system time responsiveness [Poupyrev et al., 2002];

   (c) *usage environment specifics* - always-changing context due to mobility, so no delays are admissible in processing contextual data [Bobek et al., 2013], huge variety of contextual information [Bellavista and Corradi, 2012];

7. **Interface Adaptivity Issues**
   This challenge includes the issues determined by the interface and system flexibility. The uncontrolled adaptivity can affect on user's satisfaction by the application due to the following concerns:

   (a) *transparency of adaptive behavior* - a user has to understand the idea behind the changes and follow the logic of the adaptations;

   (b) *predictability of adaptive behavior* - the system competence can be estimated incorrectly by the user, so user can rely on system too much or presume adaptation results as incorrect or unexpected;

The discovered challenges of such combination as context-awareness, adaptivity and mobile devices properties are the main direction for designing guide-

lines for context adaptive interface development for mobile applications.
The next chapter attempts to create guideline rules for designing adaptive user interfaces for mobile applications. As a proof-of-idea an associated with each guideline rule design pattern is presented.

Chapter 5

Guidelines and Patterns

As it was discussed in the section "Challenges of Context Adaptive Systems Development" (check Section 4.3 for the reference), development of adaptive interfaces based on context-awareness property of mobile devices contains a set of risks. At the same time development of an adaptive system can benefit from the specifics of mobile devices, such as mobility or context-awareness. In order to provide this value to end users and avoid negative design experience the guidelines must be created.

This chapter includes the result of investigations, literature research, analysis of existing solutions and author's own experience in mobile applications design and development including adaptive techniques. The outcome is presented in the format of guidelines rules and associated with each of them one or multiple patterns. The patterns presented for each guideline rule are projections of a particular problem.

The proposed guidelines are oriented to the developers and designers with some knowledge of application development. Moreover, the recommendations are formulated with references to the possibility of the developers being unfamiliar with the context-awareness concept.

The template for guidelines and patterns representation is a modified version of the format used in [Rossi et al., 2005] for patterns presentation. The template contains:

1. *problem section* with the formulation of problem with a possible reference to a challenge from Section 4.3;
2. *solution section* presents the actual recommendation or guideline rule for outlined problem;
3. *applicability section* indicates a possible usage of the recommendation in a form of pattern with details in what kind of situation this pattern might be applied (for instance, context, available resources, etc.).

For all the presented guideline rules the same application functionality will be used: location detection. Nevertheless, the conditions and an application design specifications will vary, this will be reflected on the designed patterns. The presented patterns (*Pattern* section) are a result of applying guideline recommendation to the location detection functionality in some determined conditions (*Application specification* section). Finally, the *Known Uses section* will list some examples of the pattern usage from industry or research papers. Some of the sections might be omitted due to obviousness or inability to present any information. The order of guidelines is free and does not reflect their priority or any other meaning behind.

5.1 Define data format and protocols

Problem: Context handling issues, see details in section 4.3.
Solution: There is a need to make a distinction between the sensor data and the context data. The raw sensors data is not always convenient for direct usage by an application. A middle layer or logical sensor intercalation handles this task with strict designation of data formats and interaction protocols between sensors and application logic. Additional purpose of such software architectural solution is handling a situation when data or sensing element is not available. An example of managing context information on mobile device with proposed format sample can be found in [Korpipaa et al., 2003]. Another simpler example was presented in this work in Table 3.1.

In addition, such attributes as precision, granularity, and accuracy affect how it is interpreted as a higher-level context data [Hong and Landay, 2001]. A variation in a set of repeated measurements is handled by precision, the smallest unit that can be measured - by granularity, the difference between the calculated value and the actual real-world value - by accuracy.

5.2 Use all sensors

Problem: The current mobile devices are equipped with numerous the physical sensors, in addition to virtual sensors (Section 3.3). It could happen with high probability that a set of sensors can provide ambivalent results on the same time frame. Such sensors conflicts might create a spurious reality performance and as a result to lead to an incorrect application behavior.

Solution: Do not neglect by any of measurements, none of the sensors has to be passed as a source of contextual information. In order to resolve this task, more complex and wide approach has to be considered. In other words, an intermediate logical sensor layer has to be implemented on top of the sensor layer. The author of this Thesis proposes to use a table as representation of different sensors characteristics, which have to be taken into consideration for decision making (see the Section 5.1 for more details). This table should include for example such meanings as trust, weight and sensor time reaction and so on. A sample of such table visualization can be found in Table 6.2. Depending on the situation or application subject, sensor weight or importance is defined and the table is updated with a new value. In most of the cases, table filling has to be done manually on the development phase with respect to privacy, device resources, source of the context, goal. etc. For instance, an explicitly gained context might have a higher priority than implicitly depending on the application purposes due to the user's privacy preferences or aspiration. The rest of the table is filled in the same way. So, this process has to take into consideration all the data

and data characteristics for choosing which data and when it can be used for interface adaptation based on the task or goal. To conclude, none of contextual data sources have to be skipped, but its usage should be done according to the task or application's goal with respect to privacy policy and device resources.

Applicability (case 1): *Application specification:* According to the application specific the most accurate user's position is required.
The user's device can be located using various techniques (sensors) each of them has pros and cons of usage:

- WiFi;
- GPS;
- Network;
- manually indicating location (For instance, by using a map or social services);
- last detected and saved location;

Each of the listed context sources (or sensors) has a different time of reaction, detection accuracy, dependence of the environment.
Pattern: By applying the guideline rule to the specified condition, the sensor with the higher accuracy (in this case, last data from the GPS), according to the table meanings, has to be taken as a source for user's location contextual data.

Applicability (case 2): *Application specification:* The application switches a profile based on two possible user's location: home and work (other locations are out of application interests).
Pattern: The detection of the location is done predominantly based on availability and strength of WiFi networks. In this case the wireless detection sensor will have a higher weight in the table than a GPS detector in the bounds of particular application.

5.3 Privacy

Privacy aspect was indicated as quite important component of the context-aware computing due to the mobility and device interactions with the environment/surrounding devices ([Schneider-Hufschmidt et al., 1993], [Baldauf et al., 2007]). Nevertheless, in this paper the privacy aspect will be omitted since this topic has been discussed in many other researches. The privacy regulations were included in most of the guidelines ([Dey and Häkkilä, 2008], [Häkkilä and Mäntyjärvi, 2006]) and most of the patterns collection include wide range of the privacy related patterns ([Chung et al., 2004], [Pearson and Shen, 2010]). So by the opinion of the author of the Thesis these works include quite complete guidance for privacy achievement in context-aware applications design.

Applicability: *Application specification:* User prefers to keep his location private.

Pattern: In the specified condition the user's state has to be hidden for the rest of the world or according to his privacy settings. And for example the interface, presented for other users, has to be adapted accordingly: the user's position should be hidden on a map or his activity has to be excluded from the news feed.

5.4 Transparency of Adaptation

Problem: One of the main goals of the adaptive behavior on mobile devices is to adjust the user interface and system behavior with little or no user interference (see Section 3.1 for more details). But such tendency can cause *Interface Adaptivity Issues* (see Section 4.3).

Solution: *AUI* has to be clear and understandable for the end user even through his involvement in the process is marginal. The self-acting variation of the user interface followed by context changes should not be discrete, but

rather smooth and transparent. On top of that, the adaptation of *UI* and system as a reaction to explicit *HCI* have to be predictable and intelligible. In other words, the user has to understand what the current system status is and what is going to happen after he takes an action at every step of interaction with the system. But the level of transparency should not transcend the *HCI* bounds, there is no need to keep visibility of all the adaptation changes. In case if there is an uncertainty of possible changes being transparent, or there are significant changes of functionality, the user interface has to indicate the completed changes by visual, text or color notice.
Such transparent behavior of the adaptive user interface increases the user's satisfaction [Gajos et al., 2008]. An unclear and confusing application behavior can confuse the users and turn away from using the application. The more predictive is adaptation mechanism the more effective is the user interface adaptation [Tsandilas and Schraefel, 2005]. The user interface has also indicate a user about the possibly faulty or undefined contextual information. A similar idea (GL9 - "Visibility of system status") was explicated in the guidelines for context-aware mobile applications design presented in [Häkkilä and Mäntyjärvi, 2006] and was indicated as the most useful by the development groups.

Applicability (case 1): *Application specification:* An application uses location contextual data as a cause for adaptation.
Pattern: The detected location or location which is used for adaptation process, as well as the processing of this data, has to be indicated in the user interface (For instance, showing a "locating" status during automatic user's position determination or "location is unavailable" in case the position can not be defined). But at the same time if the source of the location contextual data is not so significant then it can be omitted in order to avoid overloading the user with redundant information.

Applicability (case 2): *Application specification:* The principal application functionality is not available due to some reason (for instance, due to privacy or device physical limitation).

Pattern: The adaptation of the user interface should not hide the principal functionality elements in case the functionality is not available. It should rather indicate the functionality inability by color, which indicates disabling of the UI element (see Figure 5.1). If there is a possibility that the reason of why it is disabled is unclean then some kind of hint has to be provided.

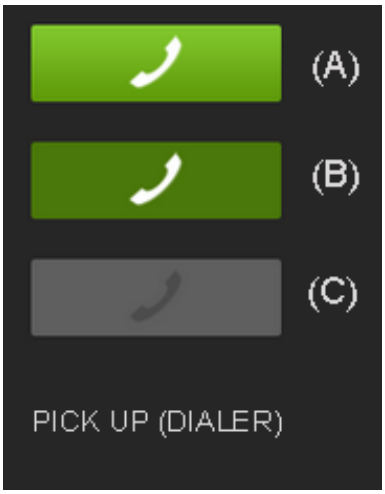

Figure 5.1: A dialer button examples: (A) - Idle, (B) - press, (C) - disabled[2].

Known Cases: An example a weather forecast application presented on Figure 5.2. The application indicates the automatically detected user's location for which the content is shown.

5.5 Level of Adaptation

Problem: The adaptivity serves in creation of clear and convenient user interface, which meets user's needs. One possible way to achieve that is an aspiration of the context adaptive application to decrease user's involvement in the adaptation process by taking all the work away from the user entails the *Interface Adaptivity Issues* (see Section 4.3). The original problem for

[2]The image is taken from Mozilla website [Fir, 2014]

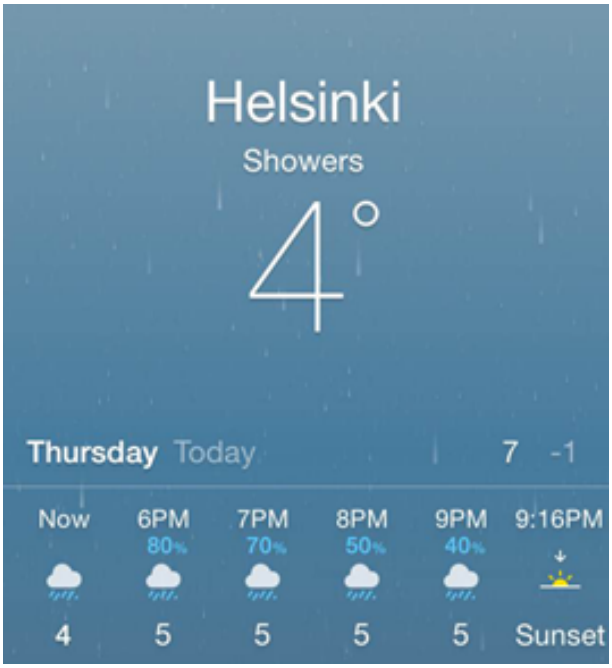

Figure 5.2: Screenshot of an iPhone weather forecast application.

this recommendation is same as for Section 5.4: uncontrolled or incomprehensible interface adaptivity can affect on user's satisfaction negatively.

Solution: The level of adaptation has to be balanced between: adaptive and adaptable system, between explicitly and implicitly gained contextual information. Nevertheless, the adaptation of user interface of context-aware systems has to aim to maximize automation for the user comfort and possibility to avoid meeting the device input limitations, such as non-availability of a full value hardware keyboard. Moreover, the higher level of automatic adaptation is more efficient for complex tasks and systems, for routine tasks and for repetitive tasks for reducing number of operations needed to perform in regular or recurring situations (for examples, the same context, conditions), which already involved the user's intervention.

Nevertheless, the application has to have a possibility to change results of the adaptation as well as automatically detected context. Extending the application with such functionality will help to maintain user's control over the

adaptation and context-awareness and keep the balance between adaptivity and adaptability. Such adjustment of adaptation levels assists to avoid undesirable application behavior due to context-recognition errors. User control might be implemented, for example, with functional dialogues or via application or profile setting.
Such application functionality ensures controllability and avoids unobtrusiveness of user interface. These options can be referred to "GL7: Secure the user control" and "GL2: Prevention from interruptions" guidelines presented in [Häkkilä and Mäntyjärvi, 2006].

Applicability: *Application specification:* Application includes automatic location detection functionality.
Pattern: The application has to automatically locate the user by using all available sensors (physical, virtual and logical), but the result of the detection has to be editable by the user. At the same time the source of the location information should be indicated in order to keep transparency of the detection and adaptation process based on this contextual information respectively.

Known Cases: Time Zone option in Date and Time settings in iOS. The time zone is defined automatically by the operating system based on the users location ((A) on Figure 5.3), but can be changed manually, (B) on Figure 5.3).

5.6 Adapt with Reference to Device Resources

Problem: This pattern is determined by the following challenges of *AUI* design: narrowness of screen space, limitation of device resources, variety of mobile devices (see Subsection 2.2.2).

Solution: The following recommendations assist in solving the mentioned issues of *AUI* design for mobile applications:

- extensive of use of compact presentation (graphical representation, an-

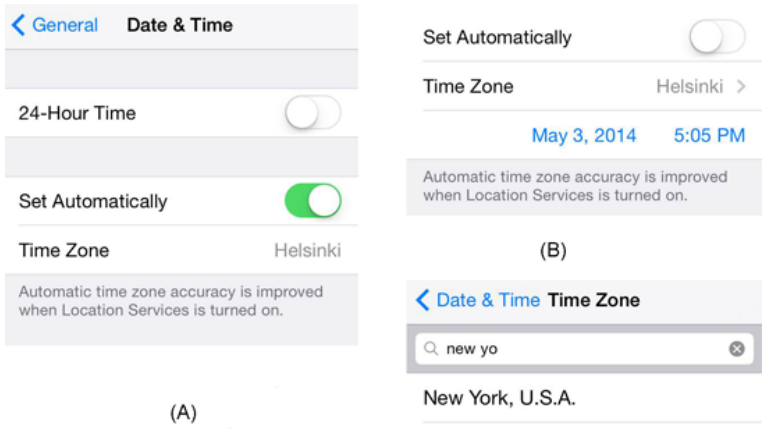

Figure 5.3: Screenshot of the Time Zone option in Date and Time settings on iPhone.

imation, color differentiation, etc.) of functional or status content performance in order to save screen space;

- take into consideration the device manufacture design *UI* guidelines for possible *UI* optimizations (for instance, exclusion of software functionality elements);
- keep the consistency across multiple platforms and devices for the same application ([Chan et al., 2002]);
- adjust layout, interface elements and content based on the type of device, manufacture and dimensions;
- take into consideration the device manufacture specifics and limitations such as battery life, network connectivity, hardware specifics, etc.;
- adapt system behavior to serve user fast, since time is a more critical factor for a mobile device user [Poupyrev et al., 2002];

Applicability (case 1): *Application specification:* Design an application with a reference to battery life.
Pattern: The balance between conservation of device resources and the goal must be kept. The usage of battery consuming techniques, such as wireless

or Bluetooth connectivity on a permanent basis has to be avoided as well as cycles in attempting to obtain access to a device's resource.

Applicability (case 2): *Application specification:* Application requires text input functionality.

Pattern: The interface must be adapted to use the most convenient way for data input: hardware keyboard in case it is available, voice input, software keyboard. Moreover, the software keyboard can be adapted according to the needs: the type of keyboard can be preselected based on the expected input data type (text, digital or browser kind of keyboard), the proper keyboard language association, etc.

Applicability (case 3): *Application specification:* Design an application for Android and iPhone platforms.

Pattern: An Android device has the "back" and the "menu" hardware buttons, so *UI* has to be designed accordingly. The iPhone hardware does include such options, so this functionality can be served by software button.

Known Cases: Google Maps application for iPhone. The automatic user location button is disabled due to unavailable sensors (check Figure 5.4).

5.7 Adapt According to Context

Problem: One of the main mobile device limitation is the narrowness of the screen space (see Subsection 2.2.2).

Solution: Despite the fact that the screen size is a limitation of handheld devices *AUIs* are more beneficial when screen real estate is constrained [Findlater and McGrenere, 2008]. The main idea of this recommendation is to limit all types of content (functional, informational, etc.) relevant for the current context such as:

- **orientation mode** - use an advantage of extra space in landscape mode (For instance, adaptation of *UI* by functionality and content extension or difference in layout);

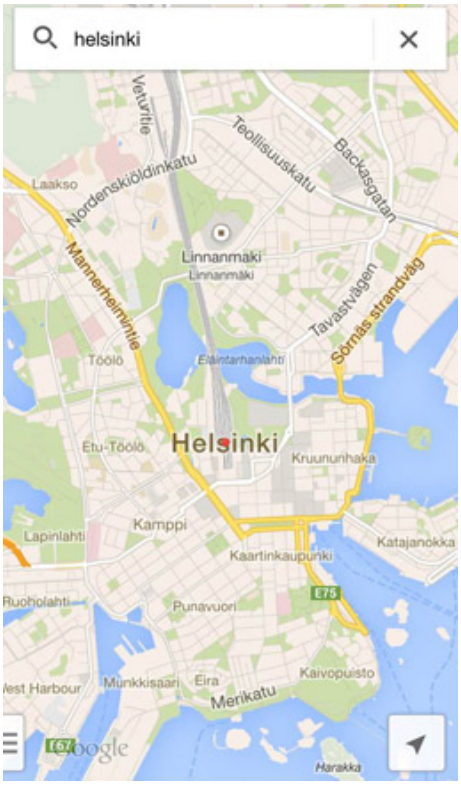

Figure 5.4: Screenshot of the Google Maps iPhone application for iPhone with disabled button for automatic location detection (right bottom corner)

- **user's personal data** - limit amount of content by applying personalization or user modeling approach;
- **element activity** - adjust *UI* element size according its activity (expand if active, collapse in other cases);
- **goal or task** - adapt functional elements according to theme, active task, goal, user's intention;

Applicability: *Application specification:* No special requirements.

Pattern: All the application functionality must be divided by *screen*, *goal* and *active task* in the same priority order. This division affects the functional elements presentation, if all conditions abode the functionality element is shown (see Figure 5.5 as an implementation of this approach).

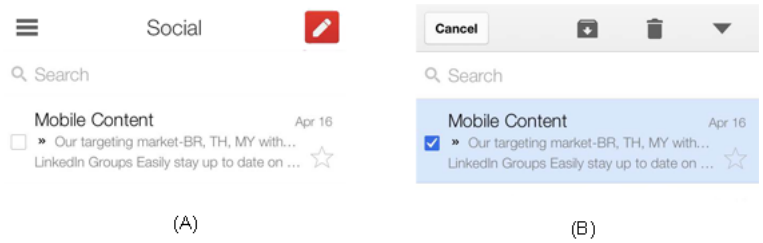

Figure 5.5: Screenshot of the Gmail application on iPhone: the normal mode (A), the functional bar updates functionality elements with the active task: an email is selected (B)

But such reduction of options must be done with an extra care, since the missing options might give less knowledge about system functionality.
Known Cases: *UI* elements such as expandable input text box, expandable search box, the edit option on the Gmail application (see Figure 5.5).

This section presents the results of research in the form of guidelines for the adaptive user interface design. In the next chapter a validation of this work is given.

Chapter 6

Validation: Application of Guidelines

The previous chapters include a theoretical base of the Thesis, investigations of the issues in *AUI* design for mobile applications and a proposal of solution for the discovered problems. This chapter is devoted to the practical part of the Thesis, proof-of-applicability of the proposed guidelines and the patterns of *AUI* design for mobile applications. Each of the proposed guideline rule will be considered as a recommendation for *AUI* creation/modification for two cases: creation of a brand new application and for a n existing application.

6.1 Use of Guidelines for Designing a New Application

Firstly, the designed guidelines and patterns will be applied to a brand new application. The aim is to verify the applicability and validity of the proposed solution.
The concept of a new application has been selected with a reference to adaptivity and maximum utilization of the context-awareness property of mobile

devices. The aim was to verify and examine the maximum of the designed recommendations. Thus, the guidelines are reviewed with a reference to the purpose, concept and idea of the developing application and some patterns are proposed as a solution for this particular case. Consequently, several components are presented as user interface elements, visual guidelines and patterns representation. This section describes the results.

6.1.1 Brand New Application Case: "Ubeel"

This subsection recites the concept, use cases, structure, functionality of the brand new application and the service of which the application is part of. The mobile application designed as a part of the service is the platform for verification of applicability of the designed guideline recommendations.

"Ubeel": Service Description

Ubeel is a service which provides a video blog functionality with automated annotation. It consists of two parts: a mobile client application and a server backend, which includes an *API* server and a Facebook [**?**] application (see Figure 6.1). The mobile application is supposed to capture video clips and then upload them to the server. The application collects various possible data from a number of sensors during video recording, such as *GPS*, Wireless, and Bluetooth, and attaches it to the video clip metadata.
In addition, it contains a set a functions to manage recorded video clips, e.g. video uploading, adding/removing tags, changing title, etc. The Ubeel server collects videos from signed up users. In addition, there is a web-based user interface which provides some advanced functionality for managing video clips collections (e.g. better tagging, settings, social features, etc.). A schematic representation of the Ubeel service is presented on Figure 6.1.

"Ubeel": Application Description

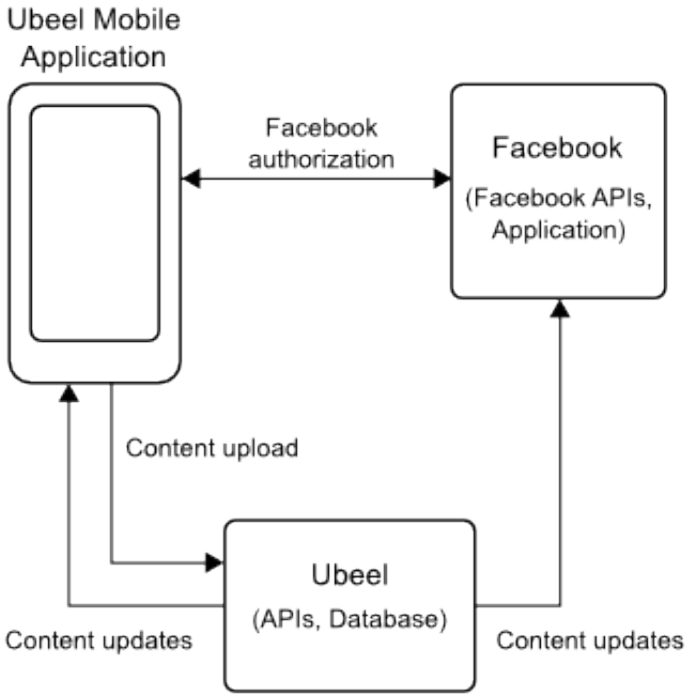

Figure 6.1: Ubeel Service Architecture

The original Ubeel client application prototype was developed for the Android platform. It contains two views: video recording functionality and video clips gallery/playback view. The screenshots of application can be found in Appendix of this paper.

The main purpose of the service is to track an event activity: wedding, student, bachelor(ette) parties, and other cases where the video capturing feature might be useful. A possible usage scenario is to capture moments of the event, members, location, congratulations etc. While capturing a video, the mobile application tries to locate other devices around, this data is assigned to the video clip and is used for tagging people who are also members of the Ubeel service, assuming that these people were filmed on the video. This association is editable.

The application includes the following functionality features:

- **video recording** - the capability of application to record a short video clip;
- **video playback feature** - possibility to watch any video clip recorded by the service from the local gallery or feed;
- **automated device tagging** - the application automatically allocated the devices via Bluetooth and automatically adds tags to the video clip;
- **uploading video to the server** - a video clip can be uploaded via the mobile application to the server at any point. In order to share event moments with friends, classmates, colleagues etc. As the result of posting a video clip, all the tagged and signed up users get a notification and can confirm or remove their presence in the clip.
- **search** - allows users to find out events, or other users of the Ubeel service, or videos by name, title, event or location. The received data might be used for hash tagging, or exploring event/users nearby;
- **subscriptions** - extension of the service which allows to subscribe to an event, other user's feed, videos with specific tags and receive updates by request or periodically. Moreover, the application present the subscription based feed with latest video clips based on user subscription preferences.

The functionality and concept of this application is suitable for evaluation of the designed recommendations. The complexity of the service, multi functionality and the variety of operational interface elements makes design of such application quite challenging. In addition, the wide usage of the sensors-provided data, both physical and virtual, complicates the design of such application system. Hence, by applying the proposed guidelines for *AUI* design of the "Ubeel" application it can be verified the effectiveness of the proposed solution.

6.1.2 Guideline in Practice in "Ubeel" Case

This section describes the application of each guideline rule in process of the design and implementation of "Ubeel" application.

Define data format and protocols (cf. Section 5.1)

The Ubeel application includes a middleware, handling the raw data from physical sensors and application logic. This level implements interface classes for processing the next contextual information: location and Bluetooth device identification. These interfaces classes modified the sensors data to more suitable for processing data format. For instance, the location interface can provide city or street name independently, besides the raw coordinates.
The location data is presented by two sources: physical (*GPS*) and virtual sensors (Facebook *APIs*). The location model interface has an indication of the data origin. This reference gives an additional base for adaptation and indication of the source of location contextual data for adaptation transparency (guideline rule described in Section 5.4).
An analogical implementation is done for the Bluetooth device identification data representation. By following the idea proposed in [Korpipaa et al., 2003] the cascade naming for contextual data managing was implemented. An example of device contextual data is presented in the Table 6.1. The Ubeel server, Facebook APIs or device local database act as the recognition services in the Ubeel application. The *confidence* value for non original sources (For instance, local database or "Ubeel" server for Facebook identifier) is calculated based on the detection datastamp.
In addition to the internal application context format, a data format for the client-server interaction has been defined.

Use all sensors (cf. Section 5.2)

By applying the guideline recommendation (see Section 5.2) a logical sensor

	Context Type	**Context Value**	**Confidence**	**Source**
1	Device:inRange	true	0.8	Device Sensor
2	Device:Name	Evgenia Samochadina - iMac	1	Device Sensor
3	Device:Name	My Friend	1	User
4	Device:Address	00:23:12:3B:C9:7B	1	Device Sensor
5	Device:FBReference	1281528529	0.7	Recognition Service 1
6	Device:FBReference	12787878529	1	Recognition Service 2
7	Device:UbReference	003429	1	Recognition Service 3
8	Device:UbReference	003429	0.5	Recognition Service 2

Table 6.1: Examples of contexts from various situations for device contextual data (based on [Korpipaa et al., 2003]).

	Source	**Availability**	**Precision**	**Priority**
1	User	true	15m	1
2	GPS	true	7.8m	2
3	Network Provider	true	1500m	3
4	WiFi	false	none	0
5	Facebook API	false	none	0
6	Calendar	false	none	0

Table 6.2: Visual representation of the weight table of location data sources.

intercalation has been developed in the "Ubeel" application. The purpose of this middleware is to reformat the contextual data to the appropriate format (Subsection 6.1.2 describes this part of the middleware operation) in addition to making a choice of the source usage based on available context and application specification. All available sensors are considered as a source of contextual data, particularly, device identification and user location information.

The application specification requires the most accurate user position, since this data is assigned as metadata to the video clip and is a subject for search and allocation of the other service users. Based on this application specification the physical sensors have a higher priority, at the same time among the available physical sensors the highest priority has the one with higher accuracy. An approximate example of such weight table for location context source is presented in Table 6.2.

The application logic chooses the first available location data source based on the priority value. Nevertheless, the explicitly gained location data has a higher top priority according to the application specification and "Level of Adaptation" guideline (see Section 5.5). In other words, after the user pointed his location on the map or selected the one provided by the Facebook API, the priority of other sources decreases and the explicit source of location data gets the top priority.

Privacy (cf. Section 5.3)

The most of the privacy policy implementation of the Ubeel application relies on the Facebook user's settings. Such social connections and privacy related to these groups are taken from Facebook and are customizable via Facebook interface. The Ubeel application uses these settings for interface adaptation, content filtering and etc. For instance, the application displays video clips according to the tagged user's privacy settings. In other words, if the user prefers to not display a video to other users of the service, it will be excluded from all feed updates regardless of the subscription preferences. The same concept spreads to the Facebook event's video clips.

Transparency of Adaptation (cf. Section 5.4)

The Ubeel application behavior and interface were built with a reference to the "Transparency of Adaptation" guideline rule (see Section 5.4), since similar ideas were indicated as the most useful and influential in [Häkkilä and Mäntyjärvi, 2006] and by other research papers.
The maximum level of the transparency implementation of the interactions between user and application. Moreover, the transparency of the current application and adaptation status were integrated in the Ubeel application as well as feedback to the user's actions (see example on the Figure 6.2).

The result of the guideline rule in the practice is presented on the Figure 6.3.

The mockup shows the interface design of the playback screen of the Ubeel application. The *UI* includes indications of current user location, an active video clip characteristics (duration, title). Moreover, there is a right side menu component, which contains information about users tagged to the video clip, such as name and photo thumbnail. In case if there are several sources of the context, the origin of it is also indicated as an icon (For instance, source of location, tagged user name and other contextual data). This design decision is following the guideline rule described in Section 5.6.

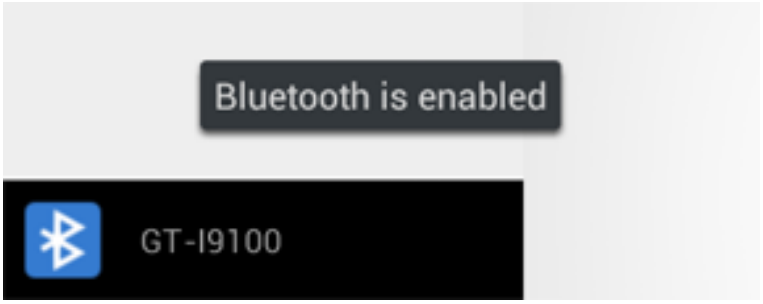

Figure 6.2: Indication of the Bluetooth sensor current status with a visual feedback to enabling the sensor

The right side menu component is not static. In most of the cases it is closed and only the number of tagged users is indicated on the top status bar. The menu can be opened by the user for more details and its state is also regulated by the application. For instance, it is closed automatically in the event of video payback is started or recording is finished. This behavior also was created accordingly to the designed guidelines, in particular "Adapt with Reference to Device Resources" (see Section 5.6) and "Adapt According to Context" (see Section 5.7).
In the open state the component described above provides additional functionality. A tagged person can be removed as detected on the video clip, as well as a new user can be added to the tagged list from Facebook friends. This functionality is a part of the applying the "Level of Adaptation" guideline rule (see Section 5.5).

Level of Adaptation (cf. Section 5.5)

According to this guideline recommendation (see Section 5.5) the application adapts the interface based on the automatically detected context. The application aims to provide the user with a maximum comfort by implementing maximum automation of context detection and interface adaptation. Nevertheless, all the automatically detected contextual data is editable: location,

Figure 6.3: Mockup of video playback screen

automatically detected Bluetooth devices, devices' names, video clip name etc. This provides the user with a full control over the adaptation.

Adapt with Reference to Device Resources (cf. Section 5.6)

This recommendation is wide and can be applied in many situations and adaptation principles. In this chapter some cases of interface adaptation based on the guideline rule were described, excluding the most obvious ones (such as phone screen dimensions adaptation). Therefore, there are several examples of adaptation implemented in the Ubeel application, which can also illustrate the applicability of this recommendation.

The main functionality of the application is utilizing the Bluetooth sensor,

which is quite power consuming hardware part. In order to save the phone resources the application switches off Bluetooth in case of inactivity of the application (then it runs on the background). This functionality is additional to the capability of turning of the functionality manually by the user (see Figure 6.2).

Adapt According to Context (cf. Section 5.7)

As it was already discussed above all the automatically gained contextual data is editable and can be redefined by the user. Some of the data is provided by the Facebook API or the application database. The lists of available contextual data are filtered based on the conditions. The following example can illustrate the application of the guideline rule. The *UI* of the Ubeel allows to choose user's current location from the places/events, provided by the Facebook API. This list is composed based on the user's location coordinates, so this limits the places/events selection.
This guideline also can help to adapt the user interface by reducing the visible functional UI elements based on the user current goal or task. In this case the user's intention has the role of contextual information. The application logic aims to provide only functional *UI* elements actual for the current task, system conditions, location, in a period of time. In order to create such logic, all the application functionality has to be divided into groups based on these parameters and displayed in case all the conditions are complied.
A practical usage of the idea can be illustrated with the record/playback functionality of the Ubeel application. The table 6.3 represents the results of using this idea.

The result of applying the presented functionality and indication division in the practice is presented as a mockup of playback screen of the Ubeel application on the Figure 6.4.

By working on this case and guideline rule application a new *AUI* component was created. It was named "expandable panel". The main idea of

Task	**State**	**Indication**	**Functionality**
Playback	playing	current location, duration, number of detected users (updated in a period of time), functionality state (playing)	stop, pause
-	stopped	current location, duration, detected users (*), functionality state (stopped), video clip name (*), location assigned to video clip (*), event (*), favorite (*)	play, upload, delete, mark as favorite
-	paused	current location, duration, detected users (*), functionality state (paused), video clip name (*), location assigned to video clip (*), event (*), favorite (*)	play, upload, delete, mark as favorite
Record	recording	current location, duration, number of detected users (updated in a period of time), functionality state (recoding)	stop
-	stopped (**)	same as for Playback/stopped	play, upload, delete, mark as favorite

Table 6.3: Table with results of application guideline rule on the playback/record Ubeel functionality.
(*) This date is editable.
(**) This state is equal to Playback task in the "stopped" state.

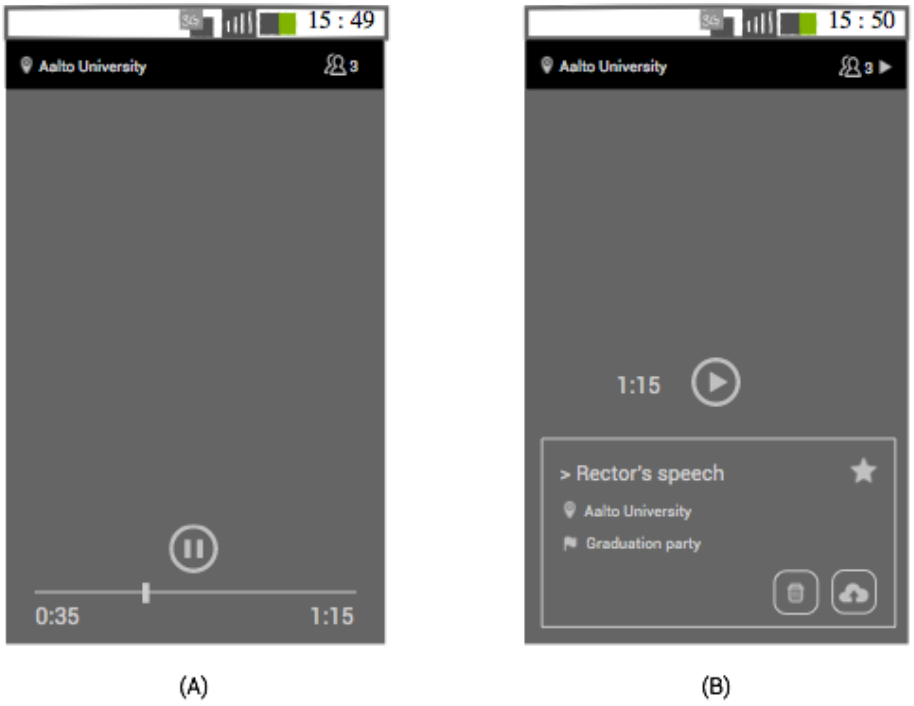

Figure 6.4: Mockups of video playback screen of the Ubeel application in playing (A) and stopped states (B)

the component is the flexibility of dimensions, which depend on the included functional *UI* elements. In the described above case the panel is completely collapsed for the playing state, since there is almost no functionality for this state defined in the division table. Moreover, the main focus of the activity is to observe the picture being recorded, so there should not be any visualization obstacles. In the stopped state the panel is in contrary expanded to fit according to the required functionality. In addition, the expanded size of the panel is not static, it is adapting based on included elements. Particularly, in case some of the functional element can not be provided due to situational context, the size of the panel will be adapted based on the included into it functional elements. A possible illustration of the case is: the user is not logged in to Facebook, so an event association for the video clip can not be

created.

6.1.3 Results

The designed guidelines were taken into account in the *UI* and system design of a brand new application. Particularly, the recommendations were applied on the planning phase of the application creation. This helped to avoid some of the possible issues of the *AUI* design for mobile applications.
Based on the results of the work described in this section, it can be concluded that the designed guidelines are applicable and working for *AUI* creation for mobile applications. Personally, it was remarked the facilitation of the *AUI* techniques involvement, then the recommendations are presented in structural and organized way. Such approach helps to take into consideration all the aspects of the *AUI* development for mobile devices.
Moreover, the components and patterns, created based on the designed in the bounds of this work guidelines, can be referred as *AUI* components.

6.2 Use of Guidelines for Improvement of a Existing Application

The second goal is to prove the efficiency of the result of the work on an existing application. The published application with user base is enhanced by applying the guideline recommendations. The measurements of user retention of the modified version determines the performance of the proposed approach.

6.2.1 Existing Application Case: "Spot in Helsinki"

This subsection shows the concept, structure, and functionality of the existing application used for evaluation of the proposed guidelines. The appli-

cation is a tour guide named "Spot in Helsinki", which has been developed for the Android platform. The author of the Thesis is one of the developers of "Spot in Helsinki" application. Particularly, the *UI* and *GUI* design and implementation was part of her task in this project. The application was published approximately 3 years ago and currently is available in the Google Play [sih] application marketplace. The application has around 7000 installations and approximately 30% of them are active users[1] by the time this paper was in progress. Moreover, the selection of the existing application is justified by the prevalence of research on location-based application. Tour guides and navigators are well explored examples of adaptive systems.

"Spot in Helsinki": Application Description

"Spot in Helsinki" is a location-based application, which provides information about attractions and events of the Helsinki region (Helsinki, Vantaa and Espoo cities). The application includes the following functionality features (see Figure 6.5):

- **listing of the region's attractions** - list of the main attraction of the city with detailed information about each of them (see case (A) in Figure 6.5);
- **listing of the region's upcoming events** - list of the upcoming and ongoing events of the city with detailed information about each of them;
- **events/spots sorting** - possibility to sort a list by a place distance from the user's current location or alphabetically/by date and time of an event;
- **events/spots filtering** - possibility to filter a list by category or period of time;
- **search** - search of events/spots;

[1]The instances of installations, which were not removed and are in use actively

- **creation of personal spots** - possibility to create a personal point of interests for later usage as a destination;
- **route calculation** - route calculation to a spot or event location, based on the user's current location and destination address. The services provides a selection between public transportation, walking or biking routes (see case (B) in Figure 6.5);
- **map** - visual representation of the events/spots lists on a map;
- **additional features** - such as ordering a taxi, buying a public transportation ticket, content update, etc.;

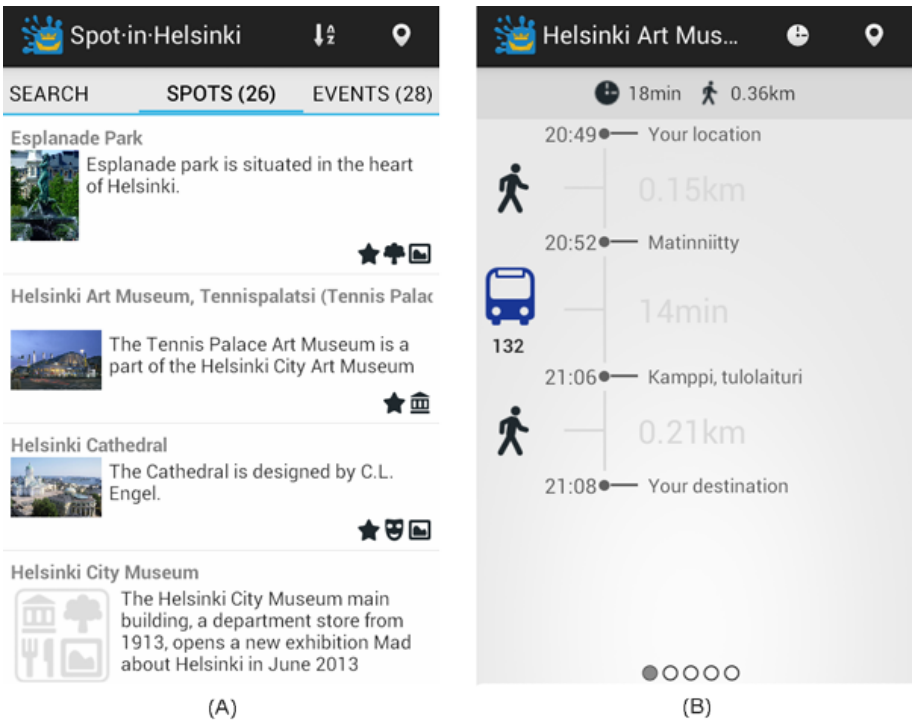

Figure 6.5: Screenshots of the "Spot in Helsinki" application: (A) spot's list and (B) route screen

All listed above features are available in the offline mode.

6.2.2 Guideline in Practice in "Spot in Helsinki" Case

Define data format and protocols (cf. Section 5.1)

There were no possible improvements found for the "Spot in Helsinki" application according to this guideline rule. The context format and interaction protocols were defined and implemented already on the design stage of the application development.

Use all sensors (cf. Section 5.2)

Originally "Spot in Helsinki" used any available source for determining user's location regardless to the precision and sensing element time reaction. Such system design decision brought delays of user interface reaction. One of the recommendations of the "Adapt with Reference to Device Resources" guideline rule states that an application has to be designed with a reference to time factor. In other words, the application response time should be as minimum as possible.

By following the designed guidelines and taken into consideration the concept and accessible sensors, a new logic for location detection was proposed. The measurements of Wireless and Network providers sensors are used until *GPS* sensor will return a result, since they have shorter time of reaction. However, the accuracy of these sources is more than 50 meters and this can be very rough for route calculation functionality. The *GPS* provider has the most exact measuring in case of auspicious environment. As soon as results from *GPS* sensor will be gotten, the logic replaces the current value and asynchronously informs the application flow. The table of priorities for "Spot in Helsinki" case for location detection functionality is presented in the Table 6.4.

The concept of the application includes a permanent knowledge of the user's location. In order to achieve that with minimum time consuming the application location detection logic uses the last detected user's location. This

	Source	Availability	Precision	Priority
1	Local database	true	7.8	1
2	WiFi	true	50m	2
3	Network Provider	true	1500m	3
4	GPS	true	7.8m	4

Table 6.4: Visual representation of the weight table of location data sources for the "Spot in Helsinki" case.

value is saved in the local device database for a reasonable period of time after which it becomes obsolete. After the *GPS* sensor returns new coordinates, the stored value is replaced. Only measurements from *GPS* are used due to its higher detection accuracy.

Privacy (cf. Section 5.3)

The "Spot in Helsinki" application does not have any social aspect as well as social networks integration. Moreover, the application service does not collect or share any user data due to concept of the application, so this guideline rule is not applicable for the "Spot in Helsinki" case.

Transparency of Adaptation (cf. Section 5.4)

The application has a map mode, which displays a map with indication of spots or events locations. Based on the application's architecture and concept the OpenStreetMaps [Ope] are used in order to support the offline mode. According to the guidelines the map mode button was disabled if the map is not completely downloaded and can not be presented. The relation of functionality inactivity and missing content can be not so clear for the end user. Due to this fact and according to "Transparency of Adaptation" guideline recommendation the action feedback was proposed. If user tries to use the map mode in case the button is deactivated (see case (A) in Fig-

ure 6.6) the application gives a feedback in form of a popup with current content download status (see case (B) in Figure 6.6).

The application *UI* was updated with a similar feedback in several places,

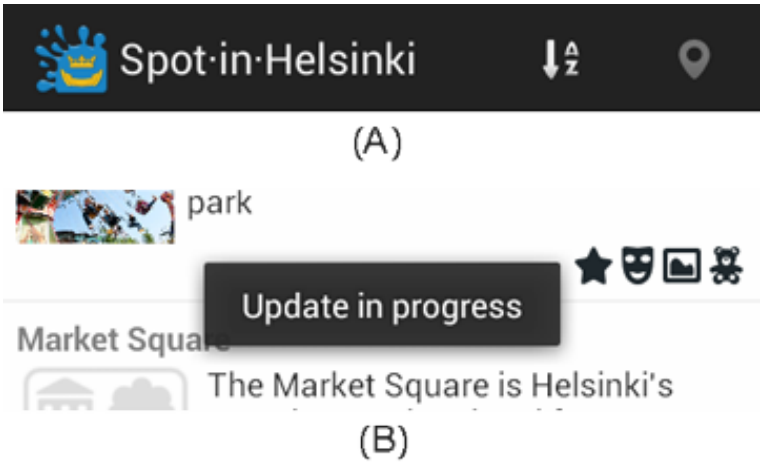

Figure 6.6: Screenshot of the "Spot in Helsinki" application with the map mode button deactivated

where the results of automatic adaptation can be unclear for the end user.

Level of Adaptation (cf. Section 5.5)

Such application functionality as filtering and sorting gives the full control over the attraction and event lists adaptation (see Figure 6.7). The user can filter and customize displayed content, in addition to the automatic lists flexibility based on user's preferences, current location and date. Such adaptable functionality gives the user a control over the automatic adaptation of the application content. In other words, most of the application solutions already followed this guideline rule.

Nevertheless, the supplementary automation was added according to the "Level of Adaptation" guideline rule. By removing an extra step in the route calculation, additional intention and redundant actions can be avoided (see case (A) on Figure 6.10). The application can save user's previous trans-

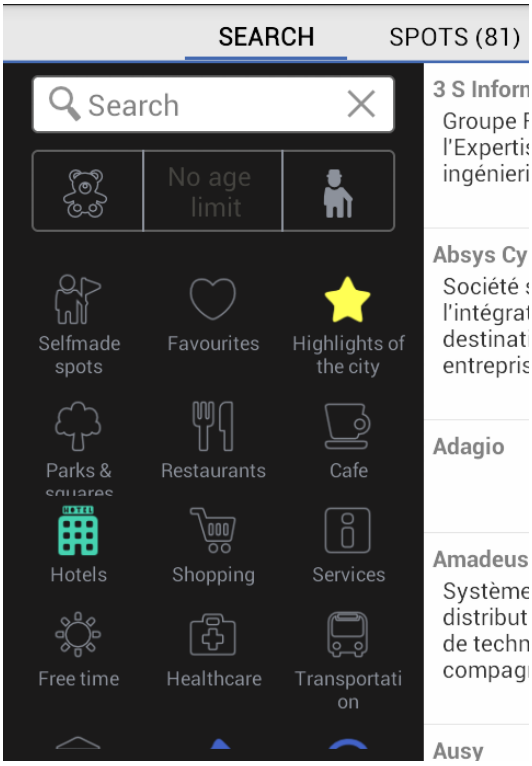

Figure 6.7: Screenshot of "Spot in Helsinki" with the filter menu of the attractions list

portation preferences and calculate an immediate route to the destination based on the saved context and user's current location (if such available). But at the same time the possibility to open the preferences screen has to be kept to insure the user's control.

Adapt with Reference to Device Resources (cf. Section 5.6)

The review of the application service and user interface with reference to "Adapt with Reference to Device Resources" guideline rule gives numerous possible improvements of the application user interface.

The sorting button in the both spot and events lists has two states: sort-

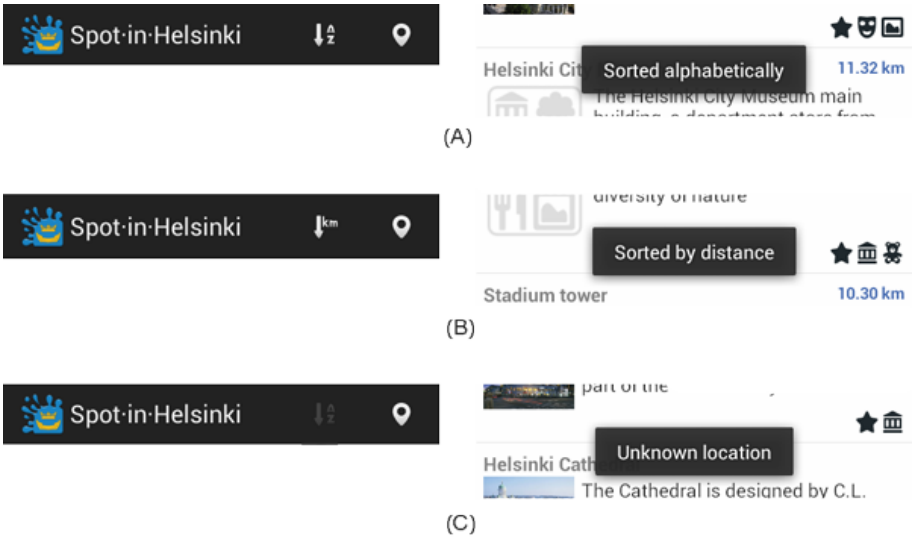

Figure 6.8: Screenshot of the sorting option in the top menu of the "Spot in Helsinki" application: the feedback and icon when list is sorted alphabetically (A), by current user's location (B) and in case user's location can not be determined (C)

ing by distance and by date for the events list and alphabetically for the attractions list (see cases (A) and (B) Figure 6.8). Both sorting options are available even if user's location can not be determined automatically. An explicit location indication is not allowed due to the application concept. According to the guideline recommendation this feature has to be excluded if the device can not automatically allocate the user. Nevertheless due to "Transparency of Adaptation" the functional user element should not be removed, since this functionality is significant and its absence might confuse the user. A way to handle this situation is to indicate the active sorting option (alphabetically for spots and sorting by date for events) and disable the functionality and give a feedback to user's action about unavailability of the location date (see case (C) Figure 6.8).

Following the same principle, a logic for adapting menu was created. Some

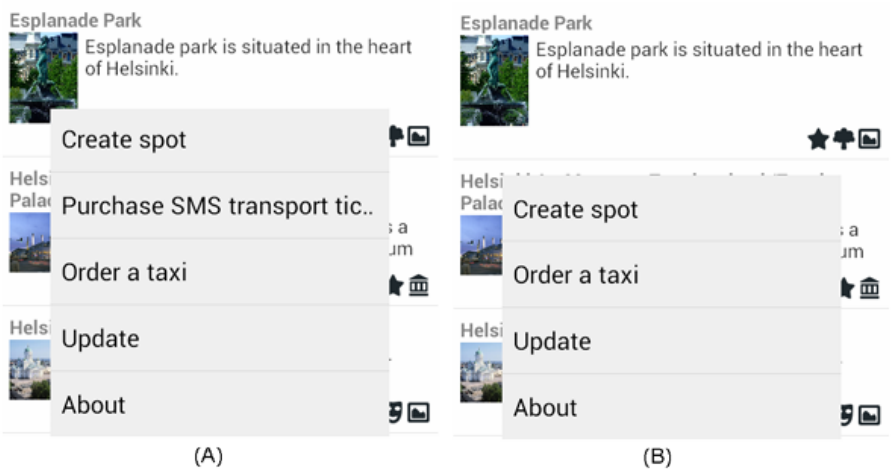

Figure 6.9: Screenshot of the "Spot in Helsinki" application menu: before applying guideline recommendations (A) and after (B)

of additional features are removed in case the functionality can not be provided due to application situational context. For instance, such feature as "Purchase SMS transport ticket" is removed from the menu in case the user is outside of Helsinki or his location can not be established (see Figure 6.9). Currently the *SMS* ticket service is working for the public transportation within Helsinki only. The option is removed from menu not disabled, since this functionality is not application decisive. Moreover, the described above enhancement suggestions are the two out of the set proposed *UI* improvements with adaptive techniques usage. The idea of these improvements are the same, so the description of these will be omitted.

Adapt According to Context (cf. Section 5.7)

The "Adapt According to Context" guideline rule brings a set of possible enhancements to "Spot in Helsinki" application. There are two significant external contexts, which are used as a base for *UI* adaptation in this application: user location and content data (feeds of city attractions and events).

Despite multiple of possible small corrections to user interface of "Spot in Helsinki" application, the only two based on the main context are described for this guideline rule.

Firstly, one proposed improvement for the user interface is for the route calculation settings screen (see case (A) on Figure 6.10). The option "Departure from the current location" for the starting point selection should be removed in case the current user's location can not be determined. And the radio button functional element is replaced by a button with the map icon (see case (B) on Figure 6.10). So the interface hides the valueless functionality based on the location context.

The second improvement is related to the listings of spots and events. The

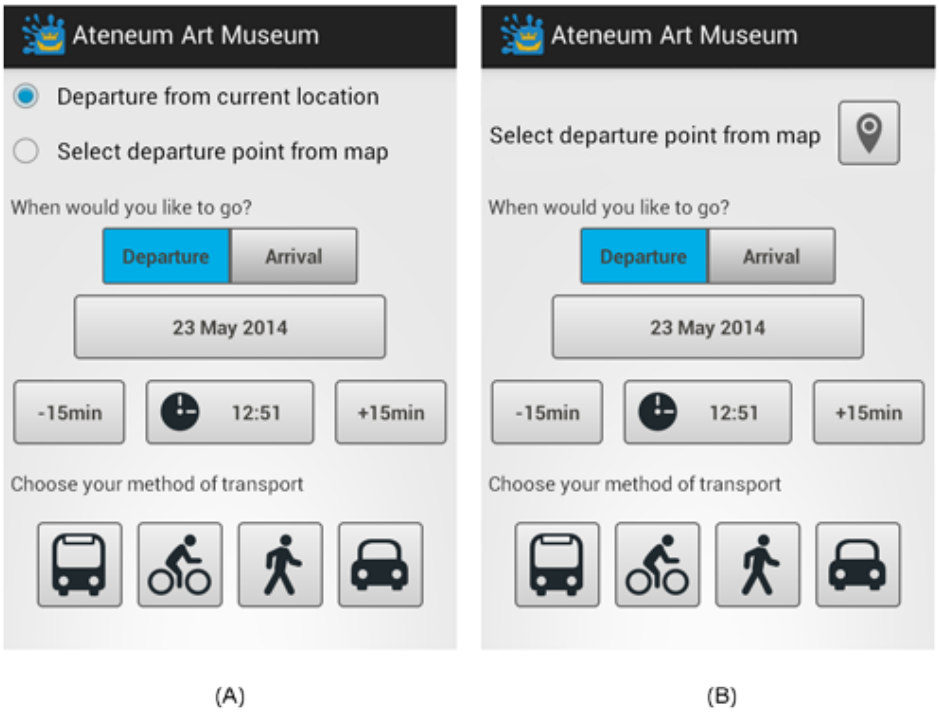

Figure 6.10: Screenshot of the route calculation settings screen of the "Spot in Helsinki" application: before applying guideline recommendations (A) and after (B)

places, which are already closed for the time the list is reviewed by the user, has to be removed from the list. This information about the availability status is calculated based on current time and the place schedule. Moreover, the automatic adaptation of these lists is configurable, so the user can turn off this filter. This user interface modification follows the "Level of Adaptation" guideline rule. In case the list is getting empty due to this filtering option (For instance, the current time is too late or early for business hours), the interface gives a hint about a possible reason of that. This improvement was proposed according to the "Transparency of Adaptation" recommendation of the designed guidelines.

6.2.3 Results

"Spot in Helsinki" application has been republished to testing environment with the implemented modifications, based on the guidelines. The download rate has not changed significantly after the application update. The download rate remained at the same level. An insignificant change was noticed in the uninstall rate, it decreased from 54% to 47%. Thus, the user retention decreased slightly. But this behavior can be explained by the high tourist season. No feedback has been received either after the modified version was published. Nevertheless, based on the previous author experience, the feedback is usually given only after significant changes in the functionality or user interface. Whereas, the guideline based changes proposed and implemented in the bounds of this work are small interface enhancements. Personally it was noticed, the increase of the application respond time due to changes related to location detection sensors' engagement logic. Moreover, the user interface became more clear, since some of the functionality presentation is adapted according to context.

6.3 Discussions

Based on the developer experience gained by applying designed guidelines on two cases, the further conclusion was done. The guidelines work effectively on the designing phase of the application rather than applying them to the existing application. This statement is inferred based on the practical part of the Thesis. Applying adaptivity techniques to an application with an existing user base can bring the following drawbacks:

- Significant enhancements of the *UI* cause dramatic changes in the application architecture, which might involve a complete redesign of the application architecture. Moreover, even a relatively simple implementation of adaptivity can be confined by the current architecture.

- On the design phase application model can be changed to follow guideline at any point, if any of *AUI* design aspects was missing.

- A significant change in the user interface can disorient users and as result distract users from using the application.

Chapter 7

Evaluation

The main goal of this Thesis was to identify guidelines and patterns for *AUI* design as a possible standardized solution, which could facilitate the future design of adaptive systems. For the verification of the workability and effectiveness of the created recommendations, they were applied in two cases: for design of a brand new application and improvement of an existing application. As the confirmation of success of the designed *AUI* and guidelines consequently the results of this work have to be evaluated.

Evaluation of a user interface is not an easy and trivial task and it becomes more complex for the adaptive user interface case [Höök, 1998]. The *AUI* intricacy grounds on the flexible interface origin. In other words, it can serve well for some users and not satisfy others, adapt accordingly to some context and not for another, behave correctly in some situations and not in others. The variety of the adaptation results makes an evaluation of intelligent (adaptive) systems a complex task.

Most of the studies of adaptive systems propose to compare systems without and with adaptivity [Höök, 1997] for evaluation. The same idea has been selected as one of the ways to evaluate the result of the designed guidelines. It was decided to take an existing application, to modify the *UI* according to the designed guidelines and to compare the two versions of the same application in terms of presentation and user satisfaction. Nevertheless, instead

of a user study, the measurements of user retention of the modified version has been selected for determining performance of the proposed approach.
A complete and comprehensive validation of the guidelines application results can not be fitted into the scope of this work. Another goal of this Thesis was to prove that guidelines for adaptive techniques based on the device context-awareness can be applied in practice for prototyping mobile applications. This was demonstrated successfully by applying them to the two mobile applications.

Chapter 8

Conclusions

This Chapter concludes the results of this Thesis work. It revises research questions and gives a brief summary of the answers. Later, it presents general conclusions of the work and findings related to the topic of this Thesis. At the begging of this paper the following questions were underlined as determinative of this work:

Question 1: How can adaptive user interface techniques be used for interface design and prototyping of mobile applications?

Question 2: What are the existing approaches to adaptive user interface development for mobile devices?

Question 3: Can a standardized approach to *AUI* design for mobile applications be developed?

The *AUI* was evaluated as a technique for mobile application user interface design by providing answers to the *AUI* base questions discussed in Subsection 2.2.3. The author of the Thesis answered the questions based on the completed research in the *AUI* field and her personal expertise in the mobile application development. According to the results of this questionnaire, the

adaptive techniques can be used not only as an effective technique for mobile application user interface design, but it also benefits from the specifics of mobile device. This part of the Thesis provides the answer to the main research question of the work: **Question 1**.

The Section 4.2 work reviews existing solutions for *AUI* design and answers **Question 2** of research questions of the Thesis. According to this research little to nothing has been done in the direction of standardized approach creation to adopting adaptive techniques for mobile application design.
Nonetheless, this review helps to conclude that the most operable and appropriate way to document a standardized approach to *AUI* for mobile application design are guidelines. This conclusion contradicts with the general recommendation to use patterns instead of guidelines ([Griffiths and Pemberton, 2005]). This fact can be explained by convenience of design and usage of standardized documentation for *AUI* design for mobile applications. Since a design pattern concentrates on relatively narrow cases and provides a solution to a particular problem within the confines of a given context. Due to variety of possible situations on mobile devices, insured by significant variation of context, the quantity of patterns for each problem increases. In other words, the pattern level of detail does not suit well for a unified solution for *AUI* design.
Consequently, guidelines are more general solution and they are a more appropriate form for sharing adaptive user interfaces design experience or knowledge for mobile applications. Based on this conclusion the main emphasis was done on guidelines development. For some of the guideline rules a pattern complementary was designed as a proof-of-idea.

The designed guidelines verified the possibility of creation of a standardized approach to *AUI* design for mobile applications and cover the **Question 3** of this research.
The designed guidelines (see Chapter 5) aim to provide common recommen-

dations and facilitate the *AUI* techniques implementation for mobile application design and development. However, as it was concluded in Section 6.3, the *AUI* techniques guidelines are more effective on the design phase of mobile application development.

The results of this work might possibly be used to influence further development of *AUI* methods into guidelines and specifications for development of mobile applications. Moreover, the designed guidelines can be taken as a base for *AUI* design specifications by the OS vendors or web communities.

Chapter 9

Future Work

The further works based on the results of this Thesis may include an evaluation of the designed guidelines by giving them to developer and designer groups for creating new applications. In order to illustrate the proficiency or lack of the created guidelines, two versions of the application have to be created by the groups: one without usage of adaptive techniques and other one by utilization of the results of the designed guidelines. Both of these versions have to use same user interfaces and to be visually as close as possible. This idea will polish the difference between applications and will make an attempt of guidelines evaluation more accurate, since a comparative analysis may depend on users' expectations and preconceptions (a new version is always considered to be better, or a better *UI* can give illusion that the application works better).

The both applications have to be given to a test group for a period of time in order to properly evaluate the results. At the end of the test period a survey or an interview of the users group will help to collect information about user experiences and a feedback for collation of versions.

The feedback obtained from the developer and designer groups might help to improve the recommendation statements and find out more and less effective. In addition, the guidelines can be applied to an existing application in order to compare the same application with and without adaptivity. In

the bounds of this work, such idea could not work well since the designer of the first and modified versions as well as the guidelines is the same person. This fact could misrepresent the results due to minor changes between the versions.

In addition, a comparison of such values as the time of application usage, the amount of goals achieved, the percentage of the used functionality out of the total amount of functions (actions), etc., can give an overall picture of the benefits or drawbacks of the adaptive guidelines for mobile applications prototyping.

The future of *UI* development and design for mobile devices should abstract from the usage of desktop paradigms, such as windows, keyboards, popups, clicks, etc. It has to be more oriented to adaptivity to the user itself to his motions, etc. In other words, the user's body has to be utilized as a part of the technology and as a way to gain more context as the base for *UI* adaptation. There is a shift to this direction that can be observed from the very beginning of the mobile devices invention. This statement can be illustrated by following cases: user's body as antenna for better network connection, finger as mouse substitute, voice and motions as input, etc. In other words, a part of the body replaces the technology. Hereafter, this approach will progress more, leading to a united system with seamless interactions giving wide a base for interface adaptation for serving user needs in a better way.

Appendices

Appendix A

Screenshots of "Ubeel" Application

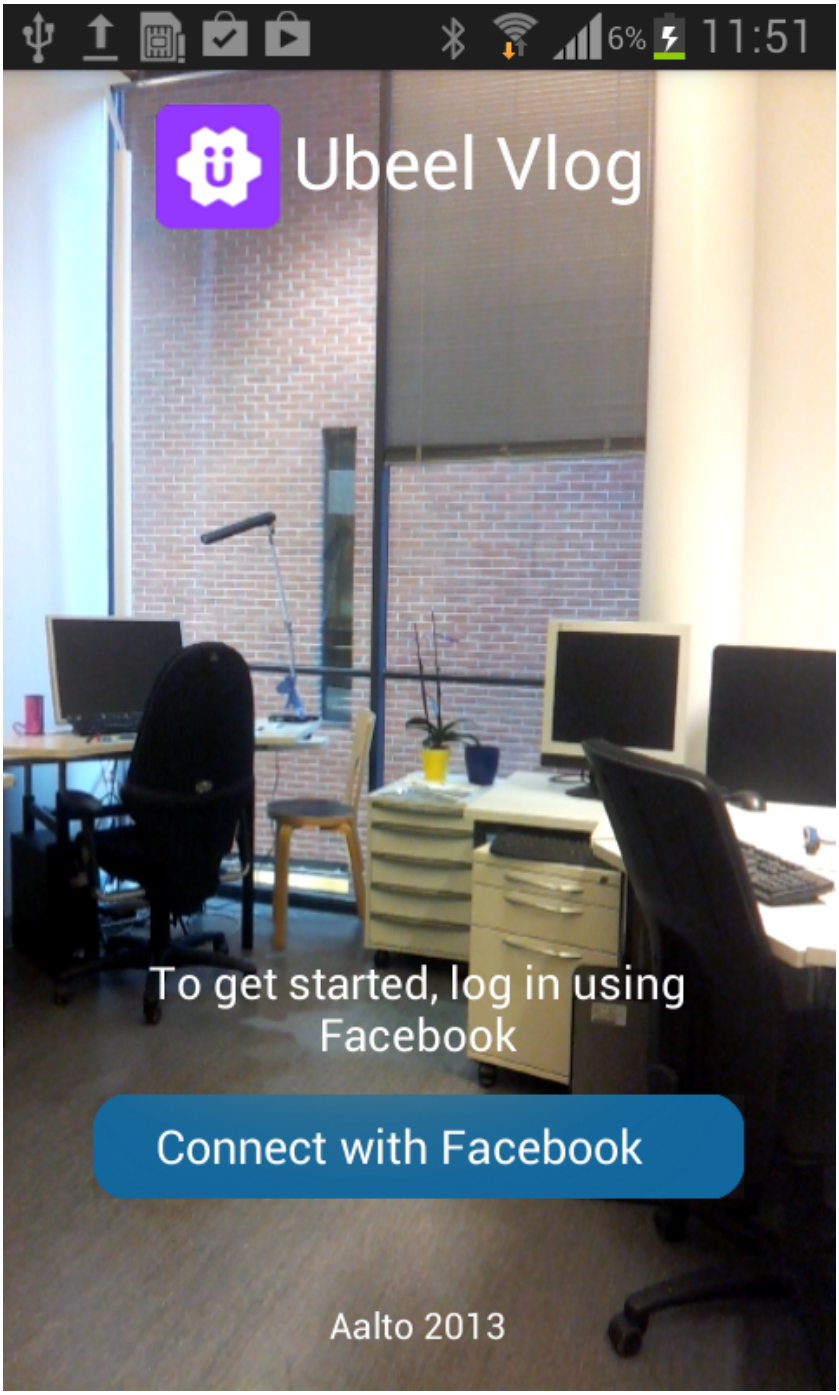

Figure A.1: Sreenshot of login screen of "Ubeel" application

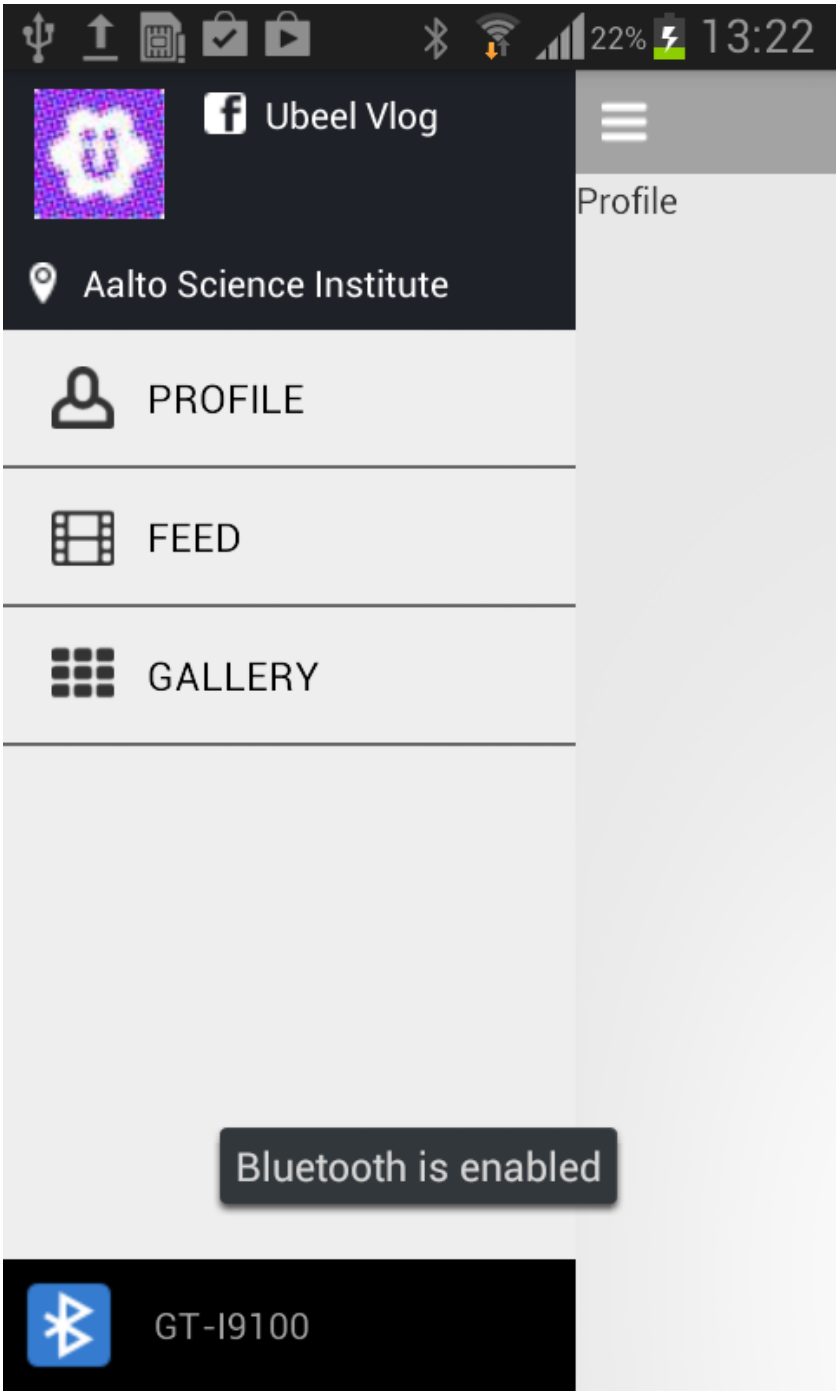

Figure A.2: Sreenshot of main screen with menu expanded

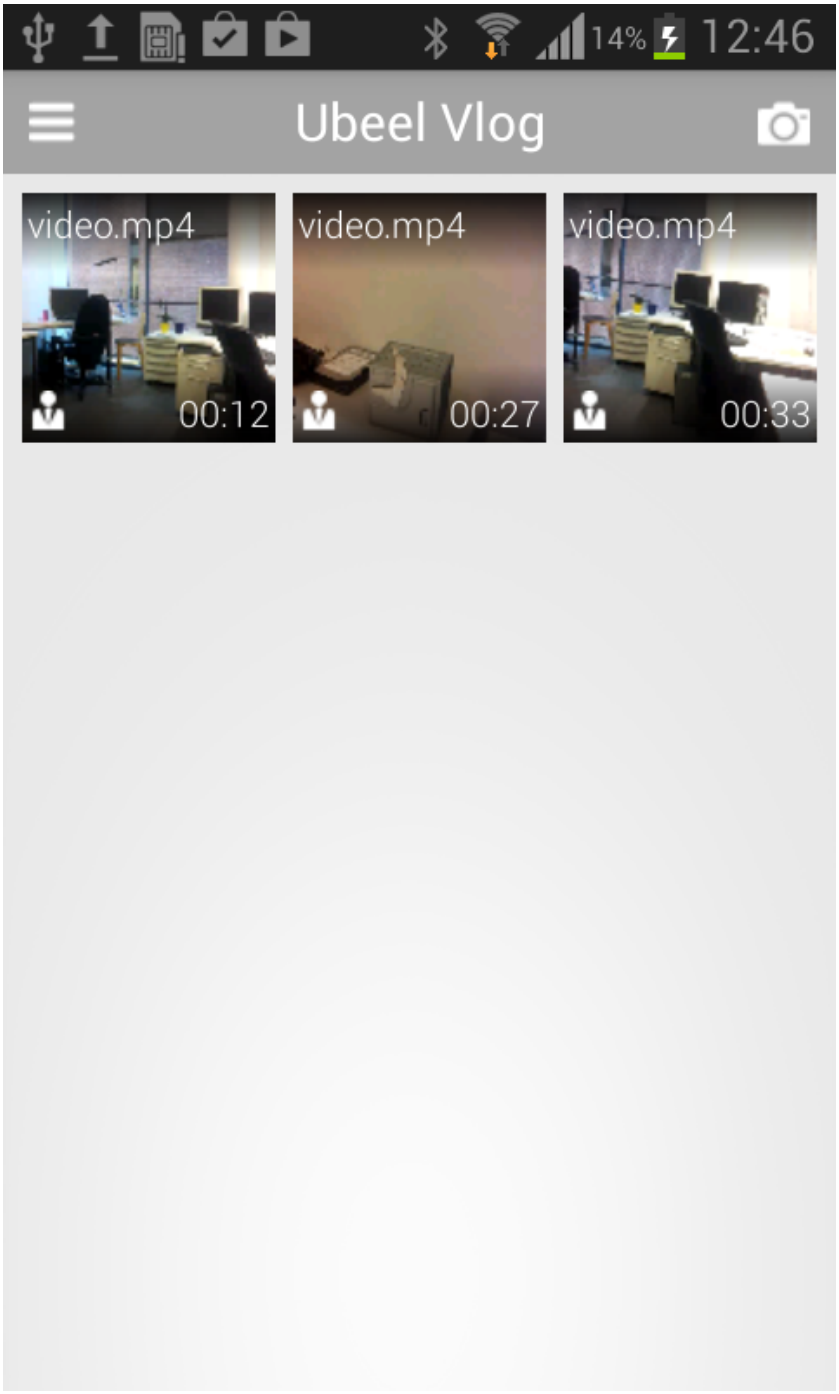

Figure A.3: Sreenshot of gallery screen

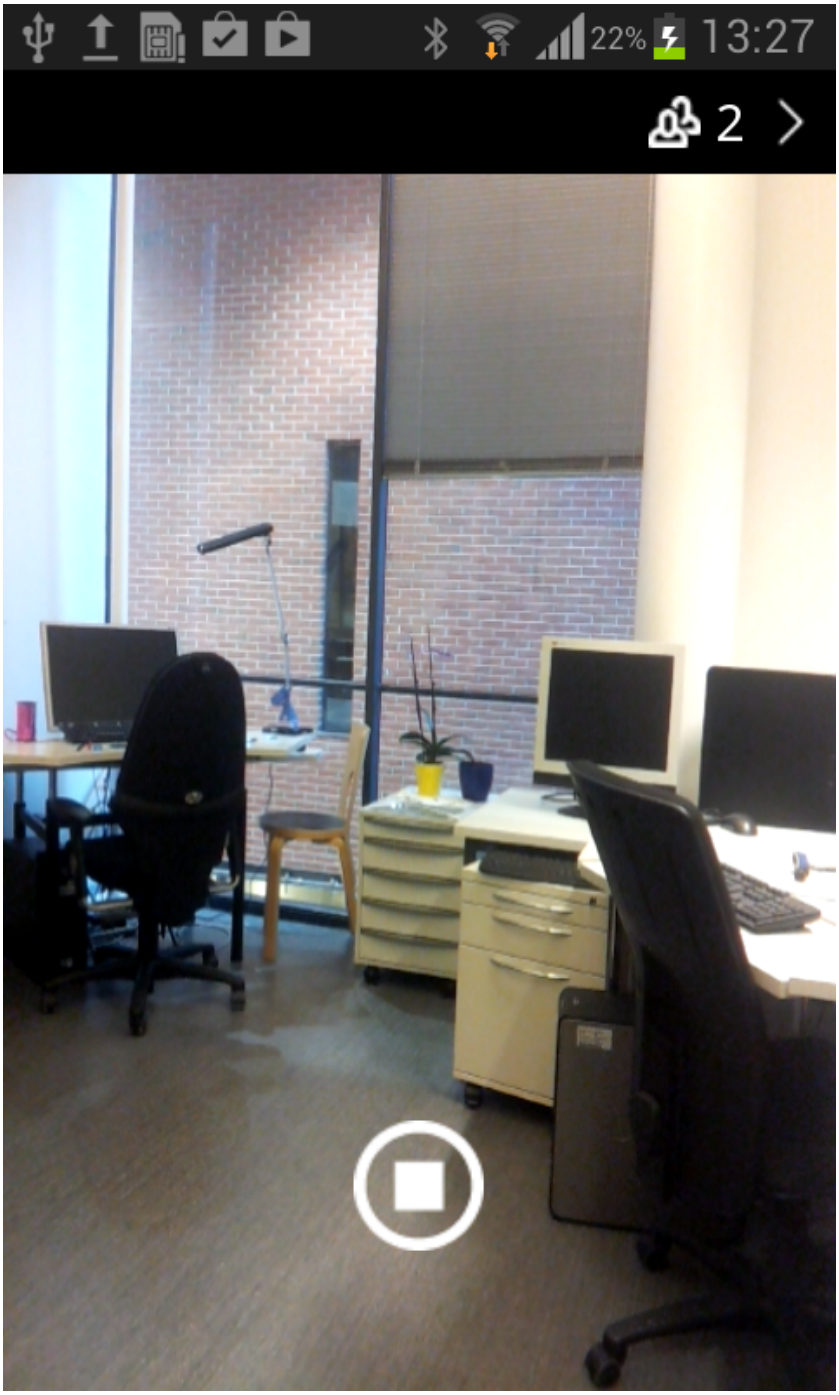

Figure A.4: Sreenshot of video screen in recording mode

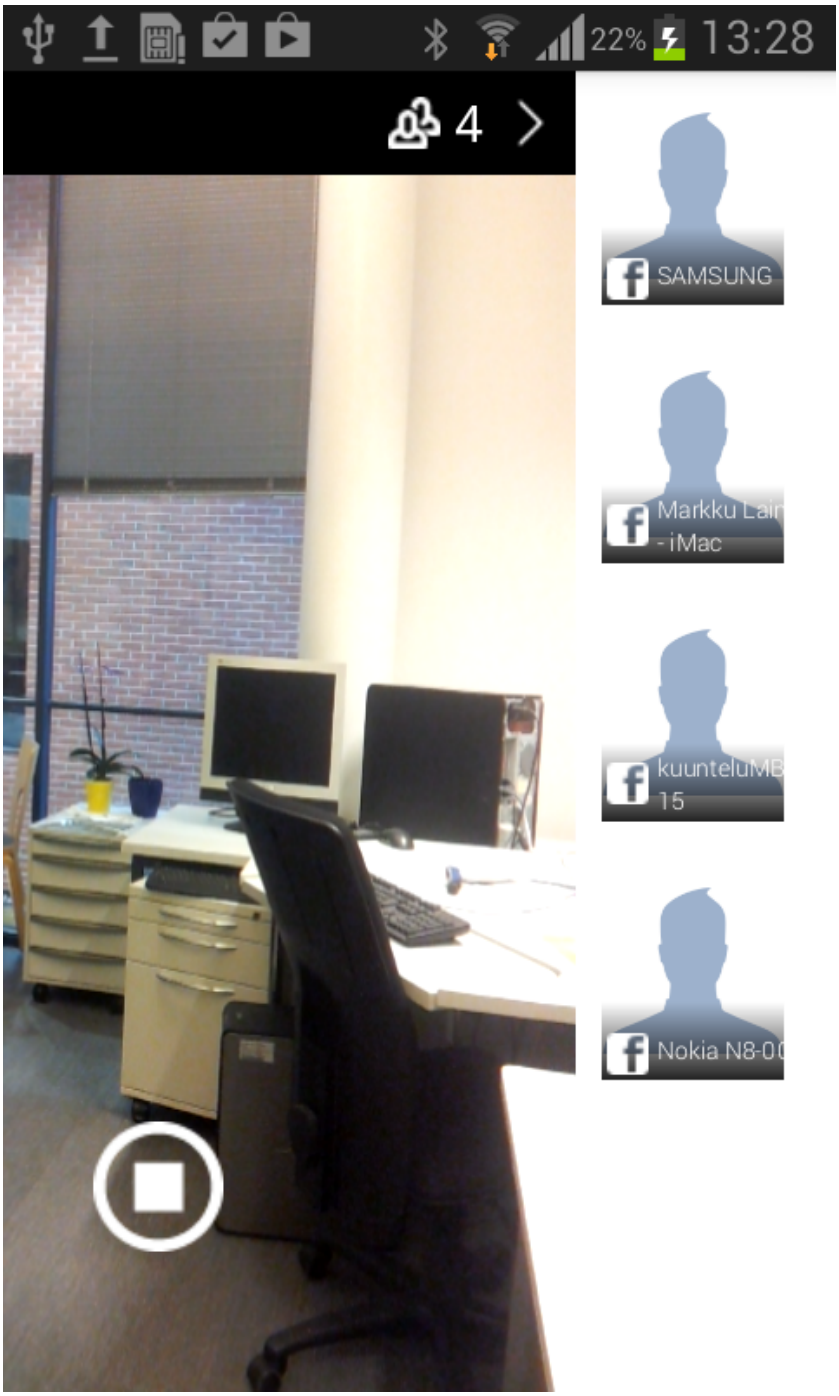

Figure A.5: Sreenshot of video screen in recording mode with user details panel expanded

Bibliography

FourSquare. URL http://foursquare.com.

Amazon Web Store. URL http://amazon.com.

Atooma Application Website. URL http://atooma.com.

Facebook Website. URL http://facebook.com.

Google Maps Mobile Application Website. URL http://google.com/mobile/maps.

Helsinki Sanomat Webpage. URL http://hs.fi.

Ikea Online Store. URL http://ikea.com.

IMDB - Internet Movie Database. URL http://imdb.com.

Last.FM. URL http://last.fm.

OpenStreetMaps - Open Source Map Project Website. URL http://www.openstreetmap.org/.

Situations Application Website. URL http://www.situationsapp.com.

Spotify. URL http://spotify.com.

TouristEye. URL http://touristeye.com.

TripAdvisor. URL http://tripadvisor.com.

Twitter. URL http://twitter.com.

YouTube. URL http://youtube.com.

"Spot in Helsinki" Application Page in the Google Play. URL https://play.google.com/store/apps/details?id=in.spotinhki.

Microsoft Word 2000 Product Enhancements Guide, 2000.

Android Design Guide, 2014. URL https://developer.android.com/design/index.html.

Amazon Help and Customer Service, 2014. URL http://amazon.com/gp/help/customer/display.html?nodeId=16465251.

Android Developer Guide, 2014. URL https://developer.android.com/guide/index.html.

Firefox OS Guidelines, 2014. URL http://www.mozilla.org/en-US/styleguide/products/firefox-os.

Google Now Presentation, 2014. URL www.google.com/landing/now.

Institute of Electrical and Electronics Engineers, 2014. URL http://www.ieee.org.

iOS Human Interface Guidelines, 2014. URL https://developer.apple.com/library/ios/design/index.html.

The World Wide Web Consortium, 2014. URL http://www.w3.org/Consortium.

WHATWG community, 2014. URL http://www.whatwg.org/.

Wi-Fi Alliance, 2014. URL http://www.wi-fi.org.

User Experience Design Guidelines for Windows Phone, 2014. URL http://msdn.microsoft.com/en-us/library/hh202915%28v=VS.92%29.aspx.

Android Patterns - collection of interaction patterns for Android application design, 2014. URL `http://androidpatterns.com`.

Gregory D. Abowd, Christopher G. Atkeson, Jason Hong, Sue Long, Rob Kooper, and Mike Pinkerton. Cyberguide: A mobile context-aware tour guide. *Wirel. Networks*, 33(5):421–433, 1997.

Adnan Al-bar and Ian Wakeman. A survey of adaptive applications in mobile computing. *Distrib. Comput. Syst. Work. 2001 Int. Conf.*, pages 246–251, 2001.

Victor Alvarez-Cortes, Benjamin E. Zayas-Perez, Victor Huga Zarate-Silva, and Jorge A. Ramirez Uresti. Current Trends in Adaptive User Interfaces: Challenges and Applications. In *Proc. Electron. Robot. Automot. Mech. Conf.*, pages 312–317. IEEE Computer Society, 2007.

J. Anhalt, A. Smailagic, D.P. Siewiorek, F. Gemperle, D. Salber, S. Weber, J. Beck, and J. Jennings. Toward context-aware computing: experiences and lessons. *IEEE Intell. Syst.*, 16(3):38–46, May 2001. ISSN 1541-1672. doi: 10.1109/5254.940025.

D. Avrahami, D. Gergle, S. E. Hudson, and S. Kiesler. Improving the match between callers and receivers: A study on the effect of contextual information on cell phone interruptions. *Behav. Inf. Technol.*, 26(3):247–259, May 2007. ISSN 0144-929X. doi: 10.1080/01449290500402338.

Matthias Baldauf, S Dustdar, and F Rosenberg. A survey on context-aware systems. *Int. J. Ad Hoc Ubiquitous Comput.*, 2(4):263–277, 2007.

Paolo Bellavista and Antonio Corradi. A survey of context data distribution for mobile ubiquitous systems. *ACM Comput. Surv. (...*, 2012.

Gregory Biegel and Vinny Cahill. A framework for developing mobile, context-aware applications. *..., 2004. PerCom 2004. Proc. ...*, 26031, 2004. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1276875`.

Daniel Billsus and MichaelJ. Pazzani. Adaptive news access. In Wolfgang Brusilovsky, Peter and Kobsa, Alfred and Nejdl, editor, *Adapt. Web*, volume 4321, pages 550–570. Springer Berlin Heidelberg, 2007.

Daniel Billsus, CA Brunk, and Craig Evans. Adaptive interfaces for ubiquitous web access. *Commun. ACM*, 45:34–38, 2002.

Szymon Bobek, Krzysztof Porzycki, and GJ Nalepa. Learning sensors usage patterns in mobile context-aware systems. *Comput. Sci. Inf. Syst. (FedCSIS), 2013 Fed. Conf.*, pages 993 – 998, 2013.

C Bolchini, FA Schreiber, and L Tanca. A methodology for a very small data base design. *Inf. Syst.*, (October 2004):1–33, 2007.

PJ Brown, JD Bovey, and X. Chen. Context-aware applications: from the laboratory to the marketplace. *IEEE Pers. Commun.*, 4(5):58–64, 1997.

Dermot Browne, Peter Totterdell, and Mike Norman, editors. *Adaptive User Interfaces*. Academic Press Ltd., London, UK, UK, 1990. ISBN 0-12-137755-5.

Peter Brusilovsky. Adaptive hypermedia. *User Model. User-adapt. Interact.*, pages 87–110, 2001.

Frank Bushmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. Pattern-oriented software architecture: A system of patterns. *John Wiley&Sons*, 1996.

Roberto Casas and David Cuartielles. Hidden issues in deploying an indoor location system. *Pervasive Comput. IEEE*, 6(2):62–69, April 2007. ISSN 1536-1268. doi: 10.1109/MPRV.2007.33.

Federica Cena, Luca Console, Cristina Gena, Anna Goy, and Guido Levi. Integrating heterogeneous adaptation techniques to build a flexible and usable mobile tourist guide. *AI Commun.*, pages 1–17, 2006.

SS Chan, X Fang, and JR Brzezinski. Usability for Mobile Commerce Across Multiple Form Factors. *J. Electron. Commer. Res.*, 3(3):187–199, 2002.

Keith Cheverst, Nigel Davies, Keith Mitchell, and Adrian Friday. Experiences of developing and deploying a context-aware tourist guide: the GUIDE project. *Proc. 6th Annu. Int. Conf. Mob. Comput. Netw.*, pages 20–31, 2000.

Eric S. Chung, Jason I. Hong, James Lin, Madhu K. Prabaker, James a. Landay, and Alan L. Liu. Development and evaluation of emerging design patterns for ubiquitous computing. *Proc. 2004 Conf. Des. Interact. Syst. Process. Pract. methods, Tech. - DIS '04*, page 233, 2004. doi: 10.1145/1013115.1013148.

Belle Beth Cooper. 10 Surprising Social Media Statistics That Might Make You Rethink Your Social Strategy, 2013. URL `http://blog.bufferapp.com/10-surprising-social-media-statistics-that-will-make-you-rethink-your-strategy`.

Joëlle Coutaz, James L. Crowley, Simon Dobson, and David Garlan. Context is key. *Commun. ACM*, 48(3):49–53, March 2005. ISSN 00010782. doi: 10.1145/1047671.1047703.

Lincoln David, Markus Endler, Simone D.J. Barbosa, and Jose Viterbo Filho. Middleware Support for Context-Aware Mobile Applications with Adaptive Multimodal User Interfaces. *2011 Fourth Int. Conf. Ubi-Media Comput.*, pages 106–111, July 2011. doi: 10.1109/U-MEDIA.2011.50.

AK Dey and J Häkkilä. Context-awareness and mobile devices. 2008.

Anind Dey, Gregory Abowd, and Daniel Salber. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interact.*, 16(2):97–166, December 2001. ISSN 0737-0024. doi: 10.1207/S15327051HCI16234_02.

Anind K. Dey, Gregory D. Abowd, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *Proc. 1st Int. Symp. Handheld Ubiquitous Comput.*, pages 304–307. Springer-Verlag, Karlsruhe, Germany, 1999.

Hartmut Dieterich, Uwe Malinowski, Thomas Kühme, and Matthias Schneider-Hufschmidt. State of the art in adaptive user interfaces. *Hum. factors Inf. Technol.*, 10:13, 1993.

Leah Findlater and Joanna McGrenere. Impact of screen size on performance, awareness, and user satisfaction with adaptive graphical user interfaces. *Proceeding twenty-sixth Annu. CHI Conf. Hum. factors Comput. Syst. - CHI '08*, page 1247, 2008. doi: 10.1145/1357054.1357249.

KZ Gajos, Mary Czerwinski, DS Tan, and DS Weld. Exploring the design space for adaptive graphical user interfaces. In *Proc. Work. Conf. Adv. Vis. Interfaces*, pages 201–208. 2006.

KZ Gajos, Katherine Everitt, and DS Tan. Predictability and accuracy in adaptive user interfaces. In *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, pages 1271–1274. 2008.

Jun Gong and Peter Tarasewich. Guidelines for handheld mobile device interface design. *Proc. DSI 2004 Annu. Meet.*, pages 3751–3756, 2004.

L Gorlenko and R Merrick. No Wires Attached: Usability Challenges in the Connected Mobile World. *IBM Syst. J.*, 42(4):639–651, October 2003. ISSN 0018-8670. doi: 10.1147/sj.424.0639.

RN Griffiths and L Pemberton. Don't write guidelines write patterns. *Retrieved August*, 2005.

Jonna Häkkilä and J Mäntyjärvi. Developing design guidelines for context-aware mobile applications. In *Proc. 3rd Int. Conf. Mob. Technol. Appl. Syst.*, pages 1–7. 2006. ISBN 1595935193.

RG Hanumansetty. Model based approach for context aware and adaptive user interface generation. 2004.

John Heggustuen. One In Every 5 People In The World Own A Smartphone, One In Every 17 Own A Tablet. *Bus. Insid.*, 15, 2013. URL `http://www.businessinsider.com/smartphone-and-tablet-penetration-2013-10`.

K. Henricksen and J. Indulska. A software engineering framework for context-aware pervasive computing. *Second IEEE Annu. Conf. Pervasive Comput. Commun.*, pages 77–86, 2004. doi: 10.1109/PERCOM.2004.1276847.

Ken Hinckley, Jeff Pierce, Mike Sinclair, and Eric Horvitz. Sensing Techniques for Mobile Interaction. In *Proc. 13th Annu. ACM Symp. User Interface Softw. Technol.*, UIST '00, pages 91–100, New York, NY, USA, 2000. ACM. ISBN 1-58113-212-3. doi: 10.1145/354401.354417.

Thomas Hofer and Wieland Schwinger. Context-awareness on mobile devices-the hydrogen approach. *Proc. 36th Hawaii Int. Conf. Syst. Sci.*, 43(7236), 2003.

Andreas Holzinger, Michael Geier, and Panagiotis Germanakos. On The Development Of Smart Adaptive User Interfaces For Mobile E-buisness Applications. 2012.

JI Hong and JA Landay. An infrastructure approach to context-aware computing. *Human-Computer Interact.*, 16:287–303, 2001.

Jong-yi Hong, Eui-ho Suh, and Sung-Jin Kim. Context-aware systems: A literature review and classification. *Expert Syst. Appl.*, 36(4):8509–8522, May 2009. ISSN 09574174. doi: 10.1016/j.eswa.2008.10.071.

Kristina Höök. Evaluating the utility and usability of an adaptive hypermedia system. *Knowledge-Based Syst.*, 10(5):311–319, March 1997. ISSN 09507051. doi: 10.1016/S0950-7051(97)00034-8. URL `http://linkinghub.elsevier.com/retrieve/pii/S0950705197000348`.

Kristina Höök. Designing and evaluating intelligent user interfaces. *Proc. 4th Int. Conf. Intell. User Interfaces*, pages 5–6, 1998.

Jadwiga Indulska and Peter Sutton. Location Managment in Pervasive Systems. *Proc. Australaisian Inf. Secur. Work.*, pages 143–151, 2003.

Anthony Jameson. Adaptive interfaces and agents. *Comput. Interact. Des. Issues, Solut.*, 2009.

Kristiina Jokinen and Jyrki Rissanen. Learning interaction patterns for adaptive user interfaces. *Work. User Interfaces*, 2002.

Christine Julien, Jamie Payton, and GC Roman. Reasoning about context-awareness in the presence of mobility. *Electron. Notes Theor. Comput. Sci.*, 97:259–276, 2004.

SK Kane. Improving Mobile Phone Accessibility with Adaptive User Interfaces. *ideals.illinois.edu*, pages 1–3, 2010.

Alfred Kobsa, Jürgen Koenemann, and Wolfgang Pohl. Personalised hypermedia presentation techniques for improving online customer relationships. *Knowl. Eng. Rev.*, 16(02):111, December 2001. ISSN 0269-8889. doi: 10.1017/S0269888901000108.

Kazunori Komatani, Shinichi Ueno, Tatsuya Kawahara, and Hiroshi G Okuno. User Modeling in Spoken Dialogue Systems to Generate Flexible Guidance. *User Model. User-adapt. Interact.*, 15(1-2):169–183, March 2005. ISSN 0924-1868. doi: 10.1007/s11257-004-5659-0.

P. Korpipaa, J. Mantyjarvi, J. Kela, H. Keranen, and E.-J. Malm. Managing context information in mobile devices. *IEEE Pervasive Comput.*, 2(3): 42–51, July 2003. ISSN 1536-1268. doi: 10.1109/MPRV.2003.1228526.

Paula Kotzé, Karen Renaud, and Kostas Koukouletsos. Patterns, anti-patterns and guidelines-effective aids to teaching HCI principles. *Proc. First Jt. BCS/IFIP WG13. 1/ICS/EU CONVIVIO HCI Educ. Work.*,

2006. URL `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.97.218&rep=rep1&type=pdf`.

JA Landay and G Borriello. Design patterns for ubiquitous computing. *Computer (Long. Beach. Calif).*, 36(8), 2003.

Talia Lavie and Joachim Meyer. Benefits and costs of adaptive user interfaces. *Int. J. Hum. Comput. Stud.*, 68(8):508–524, August 2010. ISSN 10715819. doi: 10.1016/j.ijhcs.2010.01.004.

Rosemarijn Looije, GM te Brake, and MA Neerincx. Usability engineering for mobile maps. *Proc. 4th Int. Conf. Mob. Technol. Appl. Syst. 1st Int. Symp. Comput. Hum. Interact. Mob. Technol.*, 2007.

V López-Jaquero and Francisco Montero. Model-based design of adaptive user interfaces through connectors. *... Syst. Des. ...*, pages 245–257, 2003. URL `http://link.springer.com/chapter/10.1007/978-3-540-39929-2_17`.

Wai Yip Lum and F.C.M. Lau. A context-aware decision engine for content adaptation. *IEEE Pervasive Comput.*, 1(3):41–49, July 2002. ISSN 1536-1268. doi: 10.1109/MPRV.2002.1037721.

Jani Mäntyjärvi. Sensor-based context recognition for mobile applications. *VTT Publ.*, 2003.

Jani Mäntyjärvi and Tapio Seppänen. Adapting applications in handheld devices using fuzzy context information. *Interact. Comput.*, 15(4):521–538, August 2003. ISSN 09535438. doi: 10.1016/S0953-5438(03)00038-9.

Joanna McGrenere, Ronald M. Baecker, and Kellogg S. Booth. An evaluation of a multiple interface design solution for bloated software. *Proc. SIGCHI Conf. Hum. factors Comput. Syst. Chang. our world, Chang. ourselves - CHI '02*, page 164, 2002. doi: 10.1145/503403.503406.

Rob van der; Janessa Rivera Meulen. Gartner Says Worldwide PC, Tablet and Mobile Phone Combined Shipments to Reach 2.4 Billion Units in 2013, April 2013. URL `http://www.gartner.com/newsroom/id/2408515`.

Vivian Genaro Motti and Jean Vanderdonckt. A computational framework for context-aware adaptation of user interfaces. *IEEE 7th Int. Conf. Res. Challenges Inf. Sci.*, pages 1–12, May 2013. doi: 10.1109/RCIS.2013.6577709.

Miguel A Muñoz, Marcela Rodríguez, Jesus Favela, Ana I Martinez-Garcia, and Victor M González. Context-Aware Mobile Communication in Hospitals. *Computer (Long. Beach. Calif).*, 36(9):38–46, September 2003. ISSN 0018-9162. doi: 10.1109/MC.2003.1231193.

CA Müller, F Wittig, and J Baus. Exploiting speech for recognizing elderly users to respond to their special needs. *INTERSPEECH*, 2003.

Erik G. Nilsson. Design patterns for user interface for mobile applications. *Adv. Eng. Softw.*, 40(12):1318–1328, December 2009. ISSN 09659978. doi: 10.1016/j.advengsoft.2009.01.017.

Reinhard Oppermann, editor. *Adaptive User Support: Ergonomic Design of Manually and Automatically Adaptable Software*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1994. ISBN 0-8058-1655-0.

Julien Pauty, Davy Preuveneers, Peter Rigole, and Yolande Berbers. Research challenges in mobile and context-aware service development. *Futur. Res. Challenges Softw. Serv. Conf.*, 2006.

Siani Pearson and Yun Shen. Context-Aware Privacy Design Pattern Selection. *Trust. Priv. Secur. Digit. Bus.*, 2010.

Ivan Poupyrev, Shigeaki Maruyama, and Jun Rekimoto. Ambient touch: designing tactile interfaces for handheld devices. *Proc. 15th Annu. ACM Symp. User Interface Softw. Technol.*, 2002.

Paul Prekop and Mark Burnett. Activities, context and ubiquitous computing. *Comput. Commun.*, 26(11):1168–1176, July 2003. ISSN 01403664. doi: 10.1016/S0140-3664(02)00251-7.

O. Riva, C. Di Flora, S. Russo, and K. Raatikainen. Unearthing Design Patterns to Support Context-Awareness. *Fourth Annu. IEEE Int. Conf. Pervasive Comput. Commun. Work.*, pages 383–387, 2006. doi: 10.1109/PERCOMW.2006.138.

Gustavo Rossi, Silvia Gordillo, and Fernando Lyardet. Design patterns for context-aware adaptation. *Appl. Internet Work. 2005. Saint Work. 2005. 2005 Symp. on. IEEE*, pages 5–8, 2005.

J Roth. Patterns of mobile interaction. *Pers. Ubiquitous Comput.*, pages 1–6, 2002.

Daniel Salber, AK Dey, and GD Abowd. The context toolkit: aiding the development of context-enabled applications. *Proc. SIGCHI Conf. Hum. factors Comput. Syst. ACM*, 1999.

AC Santos and JMP Cardoso. Challenges in the Development of Context-Inference Systems for Mobile Applications. *Int. Work. Program. Methods Mob. Pervasive Syst. (PMMPS), Coloca. with Pervasive.*, pages 0–3, 2010.

Bill Schilit, N Adams, and R Want. Context-aware computing applications. *Mob. Comput. Syst. Appl. 1994. WMCSA 1994. First Work. on. IEEE*, pages 85–90, 1994.

BN Schilit and MM Theimer. Disseminating active map information to mobile hosts. *Network, IEEE*, 8(5):22–32, 1994.

WN Schilit. *A system architecture for context-aware mobile computing*. PhD thesis, 1995.

Albrecht Schmidt. Implicit human computer interaction through context. *Pers. Technol.*, 4(2-3):191–199, June 2000. ISSN 0949-2054. doi: 10.1007/BF01324126.

Albrecht Schmidt. *Context-Aware Computing: Context-Awareness, Context-Aware User Interfaces, and Implicit Interaction.* The Interaction Design Foundation, Aarhus, Denmark, 2 edition, 2013. URL `http://www.interaction-design.org/encyclopedia/context-aware_computing.html`.

Albrecht Schmidt and Kristof Van Laerhoven. How to Build Smart Appliances? *IEEE Pers. Commun.*, (August):66–71, 2001.

Albrecht Schmidt, Michael Beigl, and Hans-W Gellersen. There is more to context than location. *Comput. Graph.*, 23(6):893–901, December 1999. ISSN 00978493. doi: 10.1016/S0097-8493(99)00120-X.

Matthias Schneider-Hufschmidt, Thomas Kuhme, and Malinowski Uwe. *Adaptive User Interfaces: Principles and Practice.* Elsevier Science Inc., New York, NY, USA, 1993. ISBN 0444815457.

Ben Shneiderman. *Designing the user interface: strategies for effective human-computer interaction*, volume 2. Addison-Wesley Reading, MA, 1992.

Aaron Smith. Nearly half of American adults are smartphone owners, 2012.

SL Smith and JN Mosier. *Guidelines for designing user interface software.* Number August 1986. 1986.

CA Thompson and MH Goker. Learning to suggest: The adaptive place advisor. *AAAI Spring Symp. Adapt. User Interfaces.*, 2000.

Theophanis Tsandilas and M. C. Schraefel. An empirical assessment of adaptation techniques. *CHI '05 Ext. Abstr. Hum. factors Comput. Syst. - CHI '05*, page 2009, 2005. doi: 10.1145/1056808.1057079.

Roy Want, Andy Hopper, V Falcao, and Jonathan Gibbons. The active badge location system. *ACM Trans. Inf. Syst.*, 10(1):91–102, 1992.

Mark Weiser. The computer for the 21st century. *Sci. Am.*, pages 18–25, 1991.

Scott Weiss. *Handheld Usability*. John Wiley & Sons, Inc., New York, NY, USA, 2002. ISBN 0470844469.

JL Wesson, Akash Singh, and Bradley Van Tonder. Can Adaptive Interfaces Improve the Usability of Mobile Applications? *Human-Computer Interact.*, pages 187–198, 2010.

JS Wilson. *Sensor technology handbook*. 2004. ISBN 0750677295.