Understanding and Supporting Mobile Application Usage

Dissertation

zur Erlangung des Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

vorgelegt von Matthias Böhmer
Saarbrücken, den 16. Juli 2013

Dean:
Professor Dr. Mark Groves

Head of Committee:
Professor Dr. Philipp Slusallek

Reviewers:
Professor Dr. Antonio Krüger
Professor Dr. Anind K. Dey
Professor Dr. Albrecht Schmidt

Committee Member:
Dr. Ralf Jung

Day of Defense:
September 6, 2013

Notes on Style
References to web resources (e.g., links to articles published on the Internet) are provided as URLs within footnotes with date of last access; longer URLs have been shortened. References to scholarly published resources (e.g., journal articles and conference proceedings) can be found in the Bibliography at the end of this thesis.

When referring to unknown individuals in third person singular (e.g., users of a system or anonymous participants of a study) we alternate between the female and male pronouns between paragraphs for reasons of gender neutrality.

Acknowledgements

Over the last few years I have been fortunate to get into contact with the following people. They supported me in a variety of ways working on the document which you are — hopefully — just about to start reading.

I am deeply grateful to Antonio Krüger for supervising my dissertation. He has shaped my way of thinking with fruitful discussions, while leaving me enough freedom to set out my own research agenda. I thank my reviewers Anind K. Dey and Albrecht Schmidt, who inspired my overall work early on with their own papers and research in related fields. I am thankful that both eventually agreed to review my thesis. I thank Gernot Bauer for encouraging me to start working on a PhD, helping me with the first steps, and teaching me: Clarity is essential.

I have always had the luck to work with nice people in pleasant and inspiring environments. Therefore I thank my former and current colleagues at DFKI in Saarbrücken, especially Benedict Fehringer, Florian Daiber, Johannes Schöning, Markus Löchtefeld, and Sven Gehring (in alphabetical order) who refreshed my brain cells not only with chitchat at the coffee maker, but also with challenging questions and a lot of fun. I thank Morin Ostkamp, Stefan Jürgens and Sven Luzar for being supportive colleagues during the first years of my work at Münster University of Applied Sciences.

I am thankful to all people whom I had the pleasure to collaborate with over the last few years. Among those the coauthors who worked with me on the papers that this thesis is based on: Thanks for your input and discussion! I am especially thankful for the joint work with Brent Hecht and Luis Leiva — both fruitful collaborations resulted in parts of this thesis. Many thanks also to all students who I had the pleasure to supervise for their own theses, and who worked with me as student assistants. Also many thanks to Margaret De Lap for thoroughly proofreading my thesis.

Finally, I thank my family for their support in all other matters over the years, and Dorothee: without you none of this would have been possible. Thank you for understanding and supporting me!

Für Doro und die Rabauken.

Abstract

In recent years mobile phones have evolved significantly. While the very first cellular phones only provided functionality for conducting phone calls, smartphones nowadays provide a rich variety of functionalities. Additional hardware capabilities like new sensors (e.g. for location) and touch screens as new input devices gave rise to new use cases for mobile phones, such as navigation support, taking pictures or making payments. Mobile phones not only evolved with regard to technology, they also became ubiquitous and pervasive in people's daily lives by becoming capable of supporting them in various tasks. Eventually, the advent of mobile application stores for the distribution of mobile software enabled the end-users themselves to functionally customize their mobile phones for their personal purposes and needs.

So far, little is known about how people make use of the large variety of applications that are available. Thus, little support exists for end-users to make effective and efficient use of their smartphones given the huge numbers of applications that are available. This dissertation is motivated by the evolution of mobile phones from mere communication devices to multi-functional tool sets, and the challenges that have arisen as a result. The goal of this thesis is to contribute systems that support the use of mobile applications and to ground these systems' designs in an understanding of user behavior gained through empirical observations.

The contribution of this dissertation is twofold: First, this work aims to understand how people make use of, organize, discover and multitask between the various functionalities that are available for their smartphones. Findings are based on observations of user behavior by conducting studies in the wild. Second, this work aims to assist people in leveraging their smartphones and the functionality that is available in a more effective and efficient way. This results in tools and improved user interfaces for end-users. Given that the number of available applications for smartphones is rapidly increasing, it is crucial to understand how people make use of such applications to support smartphone use in everyday life with better designs for smartphone user interfaces.

Zusammenfassung

Mobiltelefone haben sich innerhalb der letzten Jahre signifikant weiterentwickelt. Während erste Modelle lediglich Sprachtelefonie zur Verfügung stellten, ermöglichen heutige Smartphones vielseitige Dienste. Technologische Fortschritte, wie beispielsweise GPS-Lokalisierung und berührungsempfindliche Displays, haben neue Einsatzbereiche für Mobiltelefone eröffnet, wie solche als Navigationsgerät oder als Fotoapparat. Doch nicht nur in Bezug auf die Technologie haben sich Mobiltelefone weiterentwickelt, sondern auch in der Verbreitung ist die Anzahl der Geräte enorm gestiegen. Sie werden allgegenwärtig im täglichen Leben genutzt, da sie ihre Anwender bei verschiedensten Aufgaben unterstützen können. Das Aufkommen von Vetriebsplattformen für die Verbreitung mobiler Software erlaubt es dem Anwender selbstständig Modifikationen an der Funktionalität seines Geräts vorzunehmen und dieses an persönliche Zwecke und Ansprüche anzupassen.

Bisher ist wenig darüber bekannt, wie sich Anwender die Vielfalt zu Verfügung stehender Applikationen zu Nutze machen. Als Folge daraus gibt es bisher nur rudimentäre Unterstützung für Anwender, die Vielfalt von Applikationen effektiv und effizient einzusetzen. Diese Dissertation ist durch den Wandel des Mobiltelefons vom reinen Kommunikationsgerät hin zum multifunktionalen Werkzeug motiviert. Das Ziel dieser Arbeit ist es, Systeme für die Unterstützung einer besseren mobilen Applikationsnutzung zu entwickeln, deren Design auf dem neuen Verständnis von Benutzerverhalten beruht, das durch empirische Studien gewonnen wird.

Diese Dissertation hat einen zweiteiligen Beitrag: Zum einen werden theoretische Erkenntnisse dazu erarbeitet, wie Anwender die Applikationsvielfalt nutzen, installierte Applikationen auf ihren Geräten organisieren, neue Applikationen entdecken und zwischen diesen in der Ausführung wechseln. Die Erkenntnisse hierzu beruhen auf der empirischen Beobachtung von Nutzungsverhalten. Zum anderen hat diese Arbeit ingenieurwissenschaftliche Ziele dahingehend, die Anwender von Applikationen dabei zu unterstützen, ihre Smartphones sowie deren Funktionsvielfalt effektiver und effizienter einzusetzen. Dieser Beitrag resultiert in der Beschreibung implementierter Systeme und verbesserter Benutzerschnittstellen für Anwender. Angesichts der rapide wachsenden Zahl zur Verfügung stehender mobiler Applikationen ist es wichtig, zu verstehen wie Endanwender diese nutzen, denn nur so kann die Nutzung von Smartphones gebrauchstauglicher und einfacher gestaltet werden.

List of Publications

Ideas, systems, applications, studies, data, results, conclusions and other text fragments as well as figures in this dissertation have in some cases appeared in the following papers. The chapters of this dissertation are partly based on these publications.

[40] **Full paper:** M. Böhmer and A. Krüger. *A study on icon arrangement by smartphone users*. In: Proceedings of CHI 2013 (appears in Section 4.3).

[38] **Full paper:** M. Böhmer, B. Hecht, J. Schöning, A. Krüger, and G. Bauer. *Falling asleep with Angry Birds, Facebook and Kindle: a large scale study on mobile application usage*. In: Proceedings of MobileHCI 2011 (appears in Sections 3.2 and 3.3).

[36] **Full paper:** M. Böhmer, L. Ganev, and A. Krüger. *AppFunnel: A framework for usage-centric evaluation of recommender systems that suggest mobile applications*. In: Proceedings of IUI 2013 (appears in Section 5.4).

[183] **Full paper:** A. Parate, M. Böhmer, D. Chu, D. Ganesan, and B. M. Marlin. *Practical prediction, prefetch, and prelaunch for faster access to applications on mobile phones*. In: Proceedings of UbiComp 2013 (appears partly in Section 3.4).

[31] **Note:** M. Böhmer and G. Bauer. *Exploiting the icon arrangement on mobile devices as information source for context-awareness*. In: Proceedings of MobileHCI 2010 (appears in Section 4.2).

[157] **Note:** L. Leiva, M. Böhmer, S. Gehring, and A. Krüger. *Back to the app: the costs of mobile application interruptions*. In: Proceedings of MobileHCI 2012 (appears in Section 6.2).

[30] **Poster:** M. Böhmer and G. Bauer. *Improving the recommendation of mobile services by interpreting the user's icon arrangement*. In: Proceedings of MobileHCI 2009 (appears in Section 4.2).

[42] **Poster:** M. Böhmer, M. Prinz, and G. Bauer. *Contextualizing Mobile Applications for Context-aware Recommendation*. In: Adjunct Proceedings of Pervasive 2010 (appears in Chapter 5).

[37] **Poster:** M. Böhmer, S. Gehring, J. Hempel, and A. Krüger. *Revisiting Phone Call UIs for Multipurpose Mobile Phones*. In: Proceedings of Mobile HCI 2013 (appears in Section 6.3).

[140] **Poster:** A. Karatzoglou, L. Baltrunas, K. Church, and M. Böhmer. *Climbing the app wall: Enabling mobile app discovery through context-aware recommendations*. In: Proceedings of CIKM 2012 (appears in Chapter 5).

[29] **Workshop paper:** M. Böhmer and G. Bauer. *The case for Context-Collaborative filtering in pervasive services environments*. In: Proceedings of CAM3SN Workshop at MobileHCI 2009 (appears in Section 5.3).

[32] **Workshop paper:** M. Böhmer, G. Bauer, and A. Krüger. *Exploring the design space of recommender systems that suggest mobile apps*. In: Proceedings of CARS Workshop at RecSys 2010 (appears in Section 5.2).

[41] **Workshop paper:** M. Böhmer, C. Lander, and A. Krüger. *What's in the apps for context? Extending a sensor for studying app usage to informing context-awareness*. In: Proceedings of UbiMI Workshop at UbiComp 2013 (appears in Section 3.3.3).

Further, implementations, studies and results presented in this dissertation also have in part been described and presented in the theses submitted by students the author has supervised. These are:

[—] **Bachelor's thesis:** Jonas Hempel. *Lowering the impact of phone call interruptions on concurrent mobile application usage*. Saarland University, 2013, work in progress.

[—] **Master's thesis:** Lyubomir Ganev. *Evaluation of Context-aware Recommender Engines that Suggest Mobile Applications*. Saarland University, 2012.

[—] **Master's thesis:** Christian Gerdes. *Entwicklung eines Empfehlungssystems für lokationsbezogene Dienste auf Basis von Benutzerbeobachtung*. Münster University of Applied Sciences, 2010.

[—] **Master's thesis:** Thomas Remmersmann. *Entwicklung eines kontextsensitiven Empfehlungssystems auf Basis semantischer Tag-Korrelationen*. Münster University of Applied Sciences, 2009

[—] **Master's thesis:** Daniel Triphaus. *Entwurf und Implementierung einer mobilen Applikation mit kontextsensitivem Icon-Menü*. Münster University of Applied Sciences, 2009

He: “Didn’t you want to keep your phone out of the bedroom?”
She: “It’s not my phone — it’s my alarm clock!”
Author and his wife now and then.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This first chapter motivates our work and provides an introduction into the topic of this dissertation. We will first briefly recap the evolution of mobile phones, and discuss challenges that result from that evolution. Based on those challenges we will derive our research questions. Finally, Chapter 1 will give an overview on the structure of this work, its different pieces and how they form a bigger picture towards understanding and supporting mobile application usage.

1.1 The Evolution of Mobile Phones

The mobile phone recently had its 40th anniversary: On April 3rd, 1973, Motorola employee Martin Cooper made the very first phone call using a handheld mobile phone.[1] From that day until this thesis was written, four decades later, mobile phones have evolved significantly. In particular, we can distinguish two aspects of this evolution:

(1) Mobile phones improved in terms of the technology.
(2) Mobile phones became ubiquitous in daily life and worldwide.

The more functionality mobile phones provide and the better the technology is, the more likely people are to get themselves a new mobile phone [141]; i.e., (1) supports (2). Inversely, the more people have a mobile phone the bigger the market grows, and the more likely it becomes that manufacturers of mobile phones will

[1] The Guardian: *Mobile phone's 40th anniversary: from 'bricks' to clicks*, http://goo.gl/jCtnP, last accessed on 05.06.2013.

improve their devices to increase their market share (e.g. since people will be more likely to choose the model which has a camera built in); i.e., (2) also leads back to (1). The relation between these two aspects has lead to the evolution that finally resulted in the ecosystem of mobile phones as we know it today. Next, we will discuss both aspects — technology and distribution — leading to the momentum of mobile application stores.

1.1.1 Technological Development

Martin Cooper's invention became commercially available as the *Motorola DynaTAC* in 1983, 10 years after the first call was made using Motorola's prototype. The device, shown in Figure 1.1(a), was commonly referred to as the *"brick"* device due to its form factor, which had to contain 30 electronic circuit boards.[2] The first mobile phone referred to as fitting into a pocket became available in 1986: the *Technophone EXCELL PC105T*.[3] This fact was the unique selling point for advertising this model, as Figure 1.1(b) shows. The miniaturization of electronic parts is an ongoing trend mainly supported by Moore's Law, as is the reduction of the devices' weight. Nowadays, the size of a mobile phone is mainly determined by the dimensions of its battery and the display; while the latter cover nearly half of current devices' surfaces as can be seen in Figure 1.1, which shows a comparison between the first mobile phone that was commercially available and one of the latest models of the same company. While the displays on the first cellular phones were tiny, the displays on latest generation phones cover nearly the whole front face of the device. This is also leveraged by the fact that today's displays are touch-reactive and can be used as input devices, e.g. to implement buttons for keyboards.

The *IBM Simon*, which became available in 1994, was the first cellular phone that was referred to as a smartphone.[4] The device, shown in Figure 1.1(c), already provided many features that we would expect from a modern smartphone, including *"an address book, calendar, appointment scheduler, calculator, world time clock, electronic note pad, handwritten annotations and standard and predictive stylus input screen keyboards"*.[5] Such kind of first mobile phones providing a few ad-

[2]The Guardian: *Mobile phone's 40th anniversary: from 'bricks' to clicks*, http://goo.gl/jCtnP, last accessed on 05.06.2013.
[3]GSM History: *Vintage Mobiles*. http://goo.gl/QHDOw, last accessed on 07.06.2013.
[4]Wikipedia: *IBM Simon*. http://goo.gl/AEcJ7, last accessed on 08.06.2013.
[5]Wikipedia: *IBM Simon*. http://goo.gl/AEcJ7, last accessed on 08.06.2013.

(a) DynaTAC (b) Excell Pocketphone (c) IBM Simon (d) RAZRi

Figure 1.1: Milestones of the mobile phone evolution: (a) Motorola DynaTAC was the first mobile handheld phone commercially available, (b) an advertisement of the first mobile phone that would fit into a pocket, (c) the IBM Simon as fist mobile phone that was referred to as a smartphone, and (d) the Motorola RAZRi as a representative of the current generation of smartphones.

ditional services beyond phone calls, however, were mainly used by either businesspersons or very tech-savvy people [57], as such a minority.

Figure 1.2 covers some important milestones of the technical evolution of mobile phones. There were also important advances with regard to connectivity, like the improvement of mobile networks by switching from analog to digital transmission and by increasing coverage and bandwidth, wireless internet services like the *Wireless Application Protocol*, and certain influential devices including those already mentioned and the *Apple iPhone*, which became available in 2007. Also in terms of hardware the variety of sensors integrated into mobile phones has increased, including for instance sensors for positioning (based on the Global Positioning System), for measuring device acceleration, and compasses.

Another important technical milestone in the evolution of mobile phones to smartphones as we know them today in 2013 was the possibility for third party software developers to program their own applications for mobile devices. While at first only the device manufacturers and network carriers were able to add new functionality to devices, now outside developers could also provide software for mobile phones [124, 159]. For a long time a special derivate of the Java Platform, the Micro Edition, was the major technology for building software for mobile devices. This derivate was especially designed for embedded systems such as mobile devices. It was introduced in 1999 and was also known as Java Platform 2 Micro

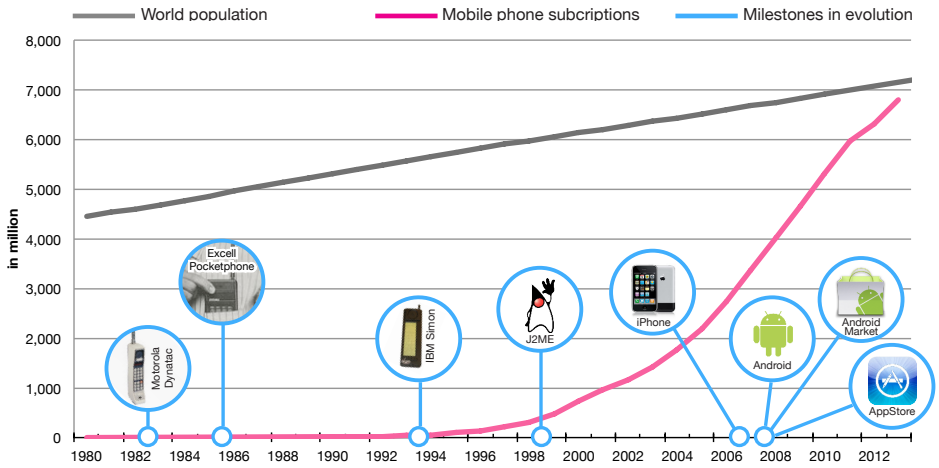


Figure 1.2: Growth of the mobile ecosystem in terms of mobile phone subscriptions in relation to world population. The figure also shows selected milestones in the technological development in recent history of mobile phones.[6]

Edition (J2ME). The J2ME platform provided standardized APIs for access to core functionalities like storage and web access, to multimedia features of devices, and to sensors like these for GPS location and acceleration.

To summarize the technical evolution of mobile phones, it can be said that they evolved from single-purpose communication devices into multi-purpose devices that support their users in a wide variety of tasks, e.g., playing games, listening to music, sightseeing, taking pictures, and cookbooks, as well as GPS-assisted navigating. In this way, the mobile phone has become increasingly analogous to a *"Swiss Army Knife"* [17, 152, 215] in that mobile phones provide a plethora of readily-accessible tools for everyday life in the form of small pieces of assistive software. Want in 2009 even anticipated that the primary computer that people use will eventually be completely integrated into a mobile phone [259]. This thesis aims to understand how people make use of their mobile phones as multi-functional tools — beyond making phone calls — to create a foundation for improving the ways to make use of this functional richness, and to build assistive systems implementing such solutions.

[6]Data sources: World population based on figures from the United Nations, http://goo.gl/ZwbPU; mobile phone subscriptions based on figures from the ITU, http://goo.gl/ojQmX and http://goo.gl/LlMMa; all last accessed on 31.06.2013; see text for milestones.

1.1.2 Ubiquitous Spread

Mobile phones have made an enormous leap from the hand of one single person in 1973 to the hands of billions of people worldwide today. In 2013 more than 6.8 billion mobile cellular phone subscriptions have been existing as reported by the ITU.[7] This enormous spread is reflected in Figure 1.2, which shows the steadily increasing number of worldwide mobile phone users. While one might think that the number of mobile phones will be limited by the size of population, it may instead be anticipated that people will start to use more than one mobile device as reported by the BBC.[8] According to a recent report of the Pew Research Center on smartphone ownership, currently about 91% of the US population owns a mobile phone, an increase from 83% in 2011; of those, the proportion using smartphones increased from 35% in 2011 to 56% in 2013.[9] In fact, Bell and Dourish [22] in 2007 already argued that mobile phones — at that time already ubiquitous computing devices with capabilities for supporting users during everyday life — are an fruition of Weiser's [261] vision of ubiquitous computing.

By looking into the numbers of people who start using specific services on their mobile phone, we can infer that the time people spent on their devices daily is also increasing, as the devices assists them with an increasing variety of applications. For instance, the share of cell phone owners using their devices to check their bank accounts increased from 18% in 2011 to 29% in 2012; using the device for recording videos increased from 18% in 2007 to 44% in 2012.[10] In fact, the time people spend with applications on their phones these days comes close to the time people spend watching television.[11]

The proliferation of mobile phones also had an impact on personal and interpersonal social aspects, resulting from the mobile phones' ubiquitous spread and pervading of daily life. They became an inherent part of their owners' daily life, as discussed by Want [258]. In fact, compared to previous generations of phones, people are more likely to keep smartphones in the same room, and take less care

[7] ITU: *The World in 2013*, http://goo.gl/4Kmwe, last accessed on 26.06.2013.

[8] BBC: *Mobiles 'to outnumber people next year', says UN agency*. http://www.bbc.co.uk/news/technology-22464368, last accessed on 26.06.2013.

[9] Pew Research Center: *Smartphone Ownership — 2013 Update*. http://goo.gl/4y24S, last accessed on 08.06.2013.

[10] Pew Research Center: *Cell Phone Activities 2012*. http://goo.gl/oD3Hq, last accessed on 07.06.2012.

[11] Techcrunch: *Time Spent In Mobile Apps Is Starting To Challenge Television, Flurry Says*. http://goo.gl/3pRH6, last accessed on 07.06.2013.

that their smartphone use will disrupt other people [83]. Researchers also recognize the disorder of people depending on their mobile phones, called *Nomophobia* [145]. And interestingly, the possibility of incessantly being able to check dynamically changing content, like updates from online social networks, has been found to be habit forming [179].

The spread of mobile phones over the whole world, and the large number of people using mobile phones on a daily basis, emphasizes the urgency of the line of research conducted in this thesis: gaining a better understanding of how people use their devices will allow us to improve the support for this vast group of users in making more efficient use of their devices.

1.1.3 The Age of Application Stores

As already briefly mentioned, the invention of a platform for the distribution of mobile software was a technical invention, which made it *"easy, inexpensive and fast to download programs"* to a user's own mobile phone.[12] Jenson [133] coins this change of paradigm as going from a *"one app, one device"* model to a *"zombie apocalypse"*, referring to technologies and applications that invade our lives. The most influential of such mobile application platforms was the *AppStore* release by Apple in 2008 for the iPhone, which was the fist of its kind.[13] However, we shall emphasize this part of the technical evolution in this section since it had a major impact on the usage of the recent generation of smartphones, and thus on this dissertation.

As described, mobile phones have evolved from single- to multi-purpose devices, and today there exist a huge number and great variety of functional add-ons that support users in different activities, e.g. banking, navigating, playing games, taking notes, or sightseeing. The most interesting phenomenon of such application stores is, that people can now easily alter the purpose of their devices by adding new functionalities, so called *apps*. Thereby, a smartphone can easily be transformed from a phone to a camera, sketchbook, bus schedule, musical instrument, or dictionary. This functional customization is supported by application stores like *Apple's AppStore* or *Google Play Market*. They provide new means for software

[12] NBC News: *Apps could be overtaking the Web, says report*. http://goo.gl/NPcvk, last accessed on 08.06.2013.

[13] Chris O'Brien: *Apple's path to the app store wasn't a straight road*, http://goo.gl/ibMYT, last accessed on 08.06.2013.

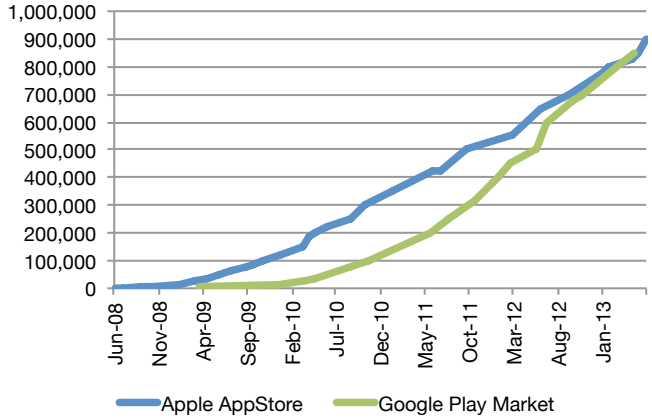


Figure 1.3: Number of applications available per application stores between June 2008 to June 2013.[14]

providers to develop, market and distribute their applications [7], and for end-users such platforms provide a convenient way to access applications since the end-users do not have to handle any technical details [124]. While the customization of a phone's look and feel and audio profiles was a very important feature of first mobile phones [109], being able to also customize phone's functionality in terms of applications also became increasingly important [17]. As such, the most important aspect of application stores that we will focus on in this work is that the end-user himself is able to customize the functionality of his own device. Due to the variety of services available on application stores, e.g. recreational applications and spiritual applications [53], mobile phones were integrated even deeper into people's lives [17].

Resulting from the popularity of mobile application stores, the number of available applications is steadily increasing. At time of writing this thesis there were more than 775,000 applications available for Apple's iPhone and more than 900,000 applications for the Android platform.[15] One can expect these numbers to be outdated soon, and therefore Figure 1.3 shows the recent growth trend of mobile applications stores, based on which a further increase can be anticipated. The number of application downloads, i.e., the number of times people installed applications on

[14]Data source: Wikipedia, http://en.wikipedia.org/wiki/App_Store_(iOS) and http://en.wikipedia.org/wiki/Google_Play, data interpolated, last accessed on 28.06.2013.

[15]Wikipedia: *List of mobile software distribution platforms*. http://is.gd/pzjWb6, last accessed on 07.06.2013.

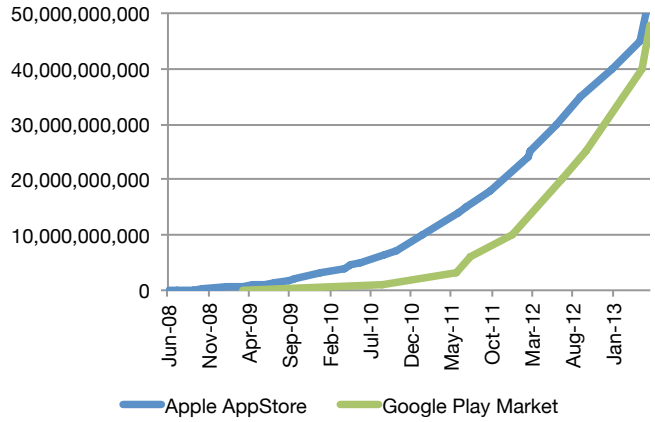


Figure 1.4: Cumulative number of downloads of applications from stores between June 2008 to June 2013.[17]

their phones, is anticipated to surpass 81.4 billion in 2013.[16] Figure 1.4 shows the increasing number of applications that users have downloaded from the application stores.

The described age of application stores has a twofold impact on this dissertation: First, it gave rise to using a large variety of applications on smartphones, and as such created the apparatus that we will study in this work. Second, the possibility of easily distributing mobile applications to a large base of users motivated the research method we are using in this work to study mobile application usage. This dissertation is motivated by questions like: How do people make use of the large variety of applications that are available on application stores? How can we help people to find the right applications to install and launch them? And what are the costs of integrating a large variety of functions beyond telecommunication into mobile phones?

1.2 Challenges and Motivation

If the owner of a first generation mobile phone pulled his device out of his pocket, it was clear what he wanted to do: He wanted to conduct a phone call. Over the

[16] Gartner: *Gartner Says Free Apps Will Account for Nearly 90 Percent of Total Mobile App Store Downloads in 2012*. http://goo.gl/QvCO8, last accessed 07.06.2013.

[17] Data source: Wikipedia, http://en.wikipedia.org/wiki/App_Store_(iOS) and http://en.wikipedia.org/wiki/Google_Play, data interpolated, last accessed on 28.06.2013.

course of the evolution of mobile phones, as described in the previous section, this has changed: If the owner of a mobile phone pulls out his device nowadays, we cannot know in advance if he wants to make a phone call, read news, check the weather forecast or play a game.

A Thought Experiment. With the increased variety of functionalities that mobile phones provide to their users, the uncertainty of anticipating which function a user would use next also increased. However, a theoretical optimum for the design of current smartphones would be a phone that instantly has the correct application open when the owner wants to use it — as old phones always had the "phone call application" immediately available. Whereas the dial pad was directly available for conducting a phone call on old mobile phones, nowadays a user first has to tell her phone that she wants to make a phone call by clicking on a telephone icon, before she can dial a number. An optimal smartphone would remove the additional effort that users need to make, which have resulted from the variety of applications that became available for mobile phones. Reasons for this optimal smartphone being impossible mainly come down to the uncertainty in modeling human behavior and people's changing interests [273]; sensor-based approaches bear an a-priori failure that leads to ambiguity [82].

Challenges. While the optimal smartphone as we sketched it is of theoretical nature and unlikely to be built, we can still draw the motivation of this work from this thought experiment. In particular, from the effort that users need to make due to the integration of more than one application on their device, we can derive four challenges worth investigating to work towards this optimal smartphone design. Thus, the following four challenges arise from considerations of the thought experiment's optimal smartphone.

- *Launching applications:* For the optimal smartphone, which would auto-launch the applications in advance when users needed them, we first need to understand how users launch applications on their own. Patterns we find will provide a foundation for building systems that anticipate which application is most likely to be launched next. In addition, we need means to observe how people make use of the applications that they have available on their devices. As such, understanding people's patterns of application launching and helping them to quickly launch the next application is an intermediate challenge which we need to address in targeting the optimal smartphone design.

- *Housekeeping applications:* Having available the functionality of auto-launching applications will remove the user's need to maintain his own set of applications. While for the reasons mentioned above this is not feasible, the challenge remains how to support people in maintaining the sets of applications they have installed on their smartphones. To help people with housekeeping their own applications we first need to understand how people do maintain their application sets and which concepts they apply to organize their devices.
- *Discovering applications:* The optimal smartphone design should not only consider applications that a user has downloaded to her smartphone, but also those that are available on the mobile application stores. In fact, the optimal smartphone should not require the user to download applications to this device but rather make them instantly available without the user needing to decide which applications to install. In working towards this goal we need to address the challenge of determining which applications out of those that are available are important for a user in her current situation. Also we should provide users with support to find such applications.
- *Multitasking between applications:* With such an automatic launching of applications as described in our thought experiment, a fourth challenge will be to decide what to do if a user already is using an application and a second application should be auto-launched. Either the user will stay in his primary application and the second one will be deferred, or the second application will be auto-launched and will interrupt the usage of the first application. To solve this challenge, we shall first consider how people currently handle this problem, and what the consequences of interrupting mobile application usage are.

The mobile ecosystem, as described in Section 1.1, is scaling at a rate faster than our understanding of how people make use of it. As such, in the context of the situation described in the previous paragraph, we are behind in providing solutions for building better support for smartphone users to leverage the large ecosystem. This work addresses the challenges we derived from the thought experiment described above, in the four fields of launching applications, housekeeping applications, discovering applications and multitasking between applications.

Non-goal of This Work. This work has its background in human-computer interaction and aims to understand and support people with regard to their smartphone

usage. It is not a goal of this thesis to develop new hardware. The technological foundation of this thesis is the state-of-the-art devices and smartphones that are currently widely used and adopted by end-users, as described in the first section. Further, it is explicitly not a goal of this thesis to develop new machine learning algorithms for the prediction of mobile application usage or recommendation of applications, though such algorithms are related to this work and are being used and discussed throughout this thesis. Rather, such challenges have been topics of collaboration apart from the work presented in this thesis, and have been presented in distinct publications (cf. [139, 140, 183]).

Personal Motivation. The author's very personal motivation results from his own experience with mobile phones, as he has witnessed the recent evolution of mobile phones on his own. While his first late 90s device was only capable of doing phone calls, sending text messages and setting up an alarm clock, his latest smartphone allowed him to draft sections of this thesis, take pictures of sketches that made their way as figures into this work, search for resources on the Internet and read papers while on the go, and develop and test applications and systems that will be introduced later in this work. The first time arranging icons on an iPhone motivated the question of *"What's the best way to do this?"* and resulted in the work presented in Chapter 4, and first queries to find new applications on an application store resulted in the research presented in Chapters 3 and 5. The short dialog quoted on page ix at the beginning of this thesis further illustrates this personal experience.

1.3 Research Questions

The overarching research question that we address in this work is:

> *How do people use mobile applications on their smartphones, and how can we build systems to support people in making effective and efficient use of mobile applications?*

This question will be answered more by specifically addressing the challenges of the aforementioned four fields of launching, housekeeping, discovery of and multitasking between mobile applications, from the perspective of mobile human-computer interaction. In particular, we can break down this research question into more achievable questions, which this work sets out to answer, as follows:

(Q1) How do people launch mobile applications, and are there particular patterns?
(Q2) How can we support people to more easily launch applications?
(Q3) How do people organize applications on their devices?
(Q4) How can we support people in housekeeping their applications?
(Q5) How can we build a system to help people discover valuable applications?
(Q6) How can we understand which newly discovered applications are of value?
(Q7) What is the impact of integrating different services into a phone?
(Q8) How can we better support application usage that is concurrent with calls?

The goal of this thesis is to contribute systems that support the use of mobile applications and to ground these systems' designs in an understanding of user behavior gained through empirical observations.

1.4 Thesis Outline

The core contributions of this work are twofold: The first part is on gaining insights into how people interact with applications on their smartphones to inform an *understanding* of mobile application use. Results of these sections are, for instance, descriptions of usage patterns or user behavior. As such, the first part of this thesis makes contributions to theory. Based on this understanding, we present the second part of this thesis, which aims to *support* people during usage of their mobile applications and make handling applications more effective and efficient. The results of this second, more systems-oriented part are, for instance, applications and systems that provide assistive functionality for end-users. As such, this second part of this thesis makes technical contributions to engineering.

These two fields of contributions — understanding and supporting — are addressed throughout this thesis and presented in four interwoven chapters to address the following four fields according to the challenges and research questions stated above:

- *Launching* of mobile applications (Chapter 3)
- *Housekeeping* of mobile applications (Chapter 4)
- *Discovery* of mobile applications (Chapter 5)
- *Multitasking* between mobile applications (Chapter 6)

In each chapter, the initial sections usually contribute to our understanding of how people use mobile applications, while the latter sections usually contribute to supporting people during application usage, building on the findings of the previous

sections. Figure 1.5 provides a graphical representation of the structure of the core parts of this thesis. It shows how chapters and sections relate to each other, especially how the ideas and approaches to support people during daily application usage are based on the knowledge gained from understanding application use on smartphones.

Following on this first chapter, Chapter 2 describes the methodology of the research conducted for this thesis, and frames the work set forth in this document by presenting related work. Finally, Chapter 7 closes by summarizing the key contributions made, sketching ideas for future work and concluding with final remarks.

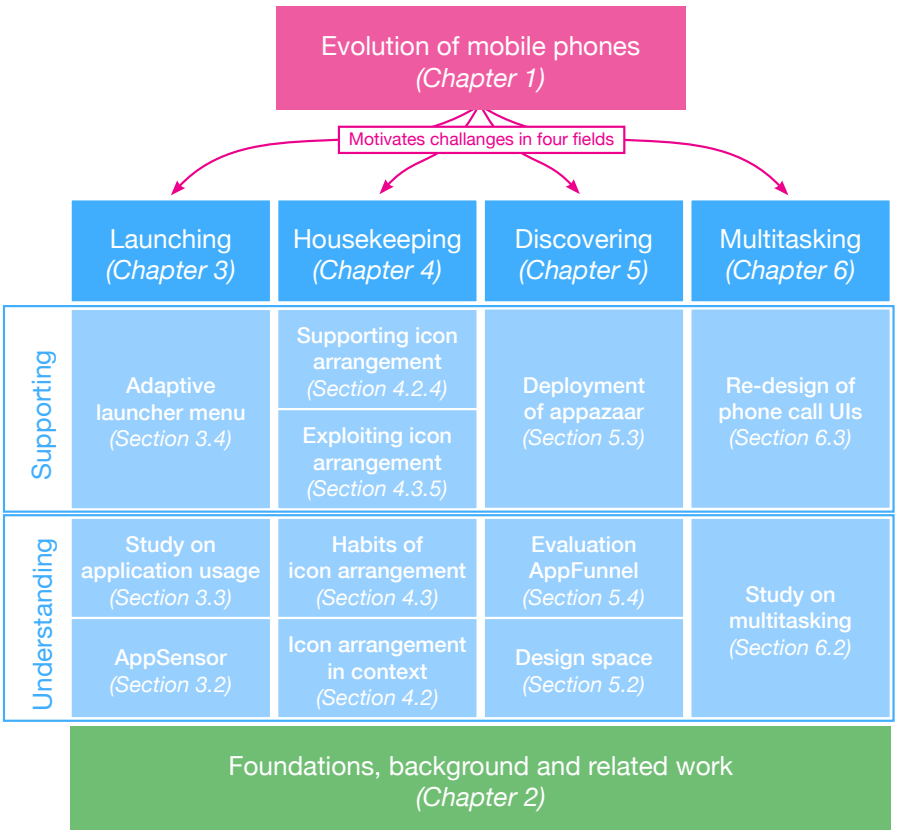


Figure 1.5: Structure of this thesis: Four main chapters on achieving insights into application usage and exploiting the understanding gained to build supportive systems for end-users.

Chapter 2

Foundations, Background and Related Work

This chapter provides the foundations, background and related work for this thesis. We introduce the notion of context as it relates to the field of mobile Human-Computer Interaction (HCI), describe our research methodology, and provide a literature review on works that are related to the four topics of this thesis as well as on fundamental work to frame this thesis.

2.1 Context in Mobile Human-Computer Interaction

Human-Computer Interaction (HCI) *"is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them."*[1] Focusing on the domain of mobile computing for the case of this dissertation, we narrow down such interactive computing systems to smartphones and mobile applications.

It is inherent to the very nature of smartphones as mobile devices that they can be used in various situations of daily life. This is particularly the case since people are carrying around their mobile phones the whole day and their activity changes over the course of the day, e.g. having breakfast, commuting to work, working,

[1] Hewett et al.: *ACM SIGCHI Curricula for Human-Computer Interaction*, http://old.sigchi.org/cdg/cdg2.html, last accessed 09.06.2013.

having lunch, and further day-to-day activities. Works relating general usage of smartphones to their users' contexts will be discussed in Section 2.3.

To characterize such user activities while using a mobile phone, we will use the concept of *context* to characterize the situation of a user's interaction with smartphones and mobile applications throughout the thesis. It attests to the importance of the concept of context that there is more than one definition, and the dispute over what makes context is still an open question [147]. For this dissertation, we refer to the definition of context provided by Dey [80]:

> *"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."*

In this work we refer to Dey's definition of context when describing the situation of a person using her smartphone, especially with regard to a user interacting with the applications her smartphone provides. In the same line of thinking, we use Dey's definition of context-awareness to describe a system's capability to use a user's context as an input in order to adapt to it. The first definition of context-aware computing was made in 1994 by Schilit and Theimer [218] as the *"ability of a mobile user's application to discover and react to changes in the environment they are situated in"*; although their primary focus was on the locations of people and objects for their Active Maps System. In this work, we adopt Dey's [80] definition of context-awareness, which is:

> *"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."*

There are three categories of context-aware systems [80]: those presenting information and services to a user based on context, those automatically executing a service for a user, and those tagging information with user's context to assist in later retrieval. The context-aware systems that we deal with in this work will typically be the smartphone's user interface adapting to the user's context, e.g. by providing varying shortcuts to applications. As such, systems of this work typically fall into the first category: those that present context-aware information to their users.

Apart from different works on defining and understanding context itself (see e.g [177, 88, 1, 147]), a large body of research dealing with context-aware systems in

various domains is also available, e.g. approaches for reasoning on user's activity from context information (see e.g. [61, 203]), platforms and middleware systems to make recognition of and access to context transparent to the applications (see e.g. [211, 192]), and systems making explicit use of certain contextual features (see e.g. [260, 229, 23]).

In addition to Dey's general definition of context given above, for different investigations of this work's research questions we will operationalize the given definition as proposed by Zimmermann et al. [272]. Zimmermann et al. categorize five different categories of information that can be taken into account for context: individuality, activity, location, time and relations. For our different studies we will shift our attention regarding these five categories. For instance, although Schmidt et al. [220] discuss that context is much richer than investigating the mere location where the interaction between a user and a system takes place, in this work we will, in particular, frequently discuss the location of a mobile user when using her device. At other places (cf. Chapter 3) we will in particular discuss time of day and relations between launches of applications as contextual information, or user activities such as shopping (cf. Chapter 4).

Taking the initially-given definitions of HCI and context we can reiterate that this dissertation is concerned with the design, evaluation and implementation of smartphone user interfaces, taking into account the impact of context on mobile application usage.

2.2 Research Methodology

The methodology of this work was inspired by the idea of deploying real systems and leveraging them for research purposes. This was motivated by the work of Müller [174] and Schöning [223], who both rely on the method of *deployment-based research*. This approach serves the twofold purpose of (i) gaining scientific insights and (ii) building systems targeting actual user needs. In deployment-based research, empirical observations are made based on deployed systems, which help reflect on a theoretical understanding of interaction. This in turn helps to further improve the design of the deployed system (cf. [174, 223]). This approach creates an iterative cycle following a user-centered design process.

2.2.1 Leveraging Application Stores

The work of this dissertation started, and first systems that we implemented had been deployed, before the era of application stores became a major driver of the mobile ecosystem (see Chapter 1). Our early idea was to build a software platform for enabling end-users to create their own mobile applications following an end-user programming approach, to deploy these applications in a repository and make them available to other users, who would then be able to use them through a context-aware recommender system (see [34, 35] for early results). However, as the ecosystems of mobile application stores grew and an increasing mass of users as well as applications became available, it felt reasonable to switch to the Android application store for the research questions concerned about the use of applications, rather than waiting to bootstrap our own repository of user-generated applications. As a matter of fact, it was straightforward to conduct studies by leveraging the momentum of the upcoming mobile application stores and study end-users using the deployed systems by collecting data on how people use the systems. The *appazaar* system was a direct result of this idea (see Chapter 5 and [42]).

In this thesis, we also cover the development and deployment of the *AppSensor* framework (see Chapter 3) and the *AppKicker* application (see Chapter 3) to mobile application stores. For all three systems (*appazaar, AppSensor, AppKicker*), we took into consideration user feedback and data logging. We collected data on the usage of the applications to iterate on the design of the technical systems as well as their user interfaces; to improve the technical system (e.g. through users' bug reports) and to inform the design (e.g. through users' feature requests).

Collecting Data in the Wild

According to Rodgers [207] the idea of turning to the wild is to study *"phenomena in the context rather than in isolation"*, and is all about observing how people change, react to, or integrate novel technologies into their everyday lives. The approach of studying new technologies in the wild was used in different fields of HCI and ubiquitous computing to study the use of new or existing systems in situ (see e.g. [45, 181, 196]). Rodgers [207] remarks that isolating specific effects observed in an in-the-wild study is difficult since the participant rather than the researcher is in control of the study, and that effects may be caused by dependencies between various factors.

According to this understanding, implementing a research study into a mobile application and deploying it an application store can be seen as a special case of conducting research studies in the wild.

Quasi-Experimental Design

Oulasvirta [178] urges rethinking experimental design when studying mobile and context-aware systems. His rationale is that assumptions about randomization and control, which can be made for experiments conducted in a controlled laboratory environment, are not necessarily valid in the wild. He proposes grounding the universal practicality of conducting studies in the scientific validity of quasi-experimentation design. Laying out the theory of experimental and quasi-experimental designs, Shadish et al. [225] characterize an *experiment* to be a study where the investigator purposefully applies two or more treatments to parts of a sample to observe the treatments' effects. Common to different forms of experiments is the control of which treatment shall be applied to which units of the sample, though the form of control can differ. Differentiating the researchers' degree of control leads to four different kinds of experiments [225]:

- In *randomized experiments* the control over the experiment is typically applied by randomly assigning treatments to units of the sample. Observed differences in the effects of treatments between groups are likely to be caused by the treatments themselves.
- In *quasi-experiments* the experimenter gives up control of assigning treatments to sample units; most importantly, they are not randomly assigned. Instead, participants in these studies self-select their treatments.
- In *natural experiments* the treatment occurs naturally and comparable conditions are introduced afterwards. As such, neither the experimenter nor the study participants are in control of applying the treatment.
- Finally, in *passive observational studies* there is no treatment of subjects at all; strictly speaking, this is by definition a non-experimental design, rather than an experiment. The aim here is to observe relationships between variables.

Moving on this continuum from controlled randomized experiments to natural experiments raises questions regarding validity of the study, reasons for this being loss of control over study participants and loss of randomization of units to study conditions. Oulasvirta [178] additionally distinguishes *randomized experiments*

into the two types of *laboratory experiments* and *analogue experiments* when applying this theory to studies in the field of HCI.

In this thesis, we are motivated by quasi-experiments (when proposing a way to evaluate recommender engines in Chapter 5), natural experimentation (when studying concepts for arranging icons in Chapter 4 and application multitasking in Chapter 6) and passive observational studies (when investigating application launching in Chapter 3).

Research in the Large

While we reiterated on the early ideas and goals of this work in the light of a changing mobile ecosystem, other researchers also began to leverage the momentum of the growing mass of mobile application users who would install applications through the application store (see Section 2.2.2 for related works). The main motivation was to be able to gain large amounts of data for statistical analysis, run studies with a heterogenous sample of participants, and observe behavior in naturally occurring user contexts [116]. Consequently, this new research approach began to establish itself as a new instrument in mobile HCI research that gained momentum as *research in the large* (coined in particular by the workshop series on *Research in the Large* [189]).

Research through the Application Store

With an increased focus on application stores as the means for running studies we also refer to this instrument as *research through the application store*. In particular, we make this distinction since studies related to HCI can — obviously — also go large in terms of number of participants, geospatial spread and length of time period of observations without being distributed over an application store; e.g. when relying on web technologies (like [121, 68]) or being an inherent part of the operating system itself. Further, while Henze and Pielot [116] in particular praise application stores as a convenient means for the distribution of studies, we also leverage this possibility to get access to low-level APIs on devices which are only available to native applications.

Only a few papers utilizing this method have been published at major conferences and in journals so far (cf. e.g. [165, 117, 173, 119, 71] and Section 2.2.2), four of them comprising parts of this work ([38, 157, 36, 183]). In 2012 McMillan finished

the first PhD dissertation [167] describing the method of *"mass participation user trials"*, as he calls it, and used this method for his studies.

In this work, we will mainly use this research method throughout (Chapters 3, 5 and 6): firstly, because it is well aligned with the field of study itself, which is about understanding and supporting the use of mobile applications; secondly, because it has a strong focus on deployed systems running on the smartphones of end-users; thirdly, because we have chosen an approach that leverages mobile application stores as a means to bring our systems into the natural contexts of our participants for studying their contextual usage; and finally, because collecting data on a large scale allows us to discover findings that would be difficult to derive from conventional studies with smaller samples [116], like phone call interruptions (see Chapter 6).

As described previously, this method of research through the application store is inspired by the approach of *deployment-based research*, is grounded in the method of *quasi-experimental design*, and can be seen as a special case of *collecting data in the wild*. The studies presented in this work follow a quasi-experimental design, e.g. when users self-select whether they download the application or choose between different features of an application, which ought to be studied, they follow a natural experiment, e.g. when we study naturally occurring interruptions of application usage, or else they follow the method of passive observational studies, e.g. when the application store is a means for distributing systems for the goal of logging data.

Reflections on Research Approach

Henze and Pielot [116] argue that upscaling studies to thousands of participants and worldwide distribution of participants makes it possible to claim external validity. However, we argue that the threats to external validity still exist and need to be taken into account and discussed when deriving conclusions from findings of studies in the large.

The main practical disadvantage of research through the application store is that the study apparatus has to be developed under the objectives and requirements of commercial applications, i.e., it needs to be *"ready for prime time"* as Henze[2] calls it, because it is competing against other applications on the application stores,

[2]N. Henze: *How to do MobileHCI Research in the Large? Tutorial at MobileHCI 2011*, http://goo.gl/Yo2EI, last accessed on 08.07.2013.

which are also drumming up user interest. This constitutes a technical challenge [72]. Further, it can take a lot of time before an application serving as a study has reached a reasonable number of installations that would make it possible to collect meaningful data. This and other challenges [72] cannot be addressed in advance, in contrast to a more controlled lab study, and they are also out of control for the researchers. Practices like recruiting people from social networks or advertising applications on famous websites might also bias the sample, and the method of recruiting participants needs to be reported when describing results of such studies conducted through the application store.[3]

When moving form laboratory studies to the application store, contact with the study participants also gets lost, and when collecting data from individuals on a large scale, ethical concerns may arise. Guidelines and best practices on ethical questions are a current topic (cf. e.g. [113, 166, 172, 173, 187]), first studies in the large having been conducted without taking much care of this topic. We shall emphasize that for all studies reported in this dissertation, the users had to explicitly opt-in to data collection, as we implemented the two-button approach proposed by Pielot et al. [187]. As such, we had large numbers of users not contributing to our studies despite using our applications. This underpins the idea of deployment-based research where besides inferring knowledge, the development of the systems is a second goal. More recently, approaches for mitigating some drawbacks of large-scale trials, by adding traditional local user studies in parallel, being discussed [173].

Over the course of this thesis we will present different studies. Some of these studies draw samples from the same deployed systems, but at different points in time. Based on longitudinal observations following the method laid out in this chapter, single studies report on different time series of those systems. The number of users that each study is based on as well as the time spans of investigations may differ. It is worth keeping in mind that this results in different samples, which should not lead to confusion on the part of the reader.

Barnard et al. [18] stated that there is a disconnect between the real use and the evaluation of mobile systems, and they to close the gap between the context of actual use and the context of studies of mobile systems and applications. The authors study how changes in contextual variables (task type, sitting/walking, light-

[3] This is regarded as common sense in the community and was agreed on as a result of a discussion at 3^{rd} Workshop on Research in The Large at MobileHCI 2013.

ning level) impact users' performance and workload in reading and comprehension tasks. Barnard et al. conclude that we need to keep the context of use in mind when designing and evaluating new systems, as it can strongly impact user behavior when interacting with a mobile device. The method of research through the application store, which we make use of in this work, is a way to study systems in their real context. Related studies using the same approaches are presented in the next section.

This research approach — although it is in its infancy since it is inherently coupled to the era of mobile application stores — has been used by different researchers in the last few years. Their works will described in the next section. The studies we present in this thesis are framed by the theory of experimentation and mainly driven by the ideas of deployment-based research, research in the wild and research through the applications store.

2.2.2 Methodically Related Studies

Although the approach of research in the large is relatively young, it has been applied to different research questions in HCI, ubiquitous computing and related research. For a better overview on work related to this method, and additional classification of the method, we adopt categories from Henze's MobileHCI tutorial[4]. We distinguish among five categories of works: (1) using application stores as proofs of concepts and for dissemination of work, (2) leveraging application stores as a research tool to study distinct research questions and learn about new aspects when upscaling studies, (3) for study of dedicated research questions at scale, and (4) for studying the ecosystems of smartphones and application usage itself. We will further add to this classification (5) a group of papers presenting *lessons learned* from using the approach previously presented. While we will present related work as belonging in individual categories, some research actually falls into more than one category.

Proving Concepts and Disseminating Results

The motivation of deploying an application to application stores for some works is to prove the concept behind the research, or collect additional feedback on research

[4]N. Henze: *How to do MobileHCI Research in the Large? Tutorial at MobileHCI 2011*, http://goo.gl/Yo2EI, last accessed on 08.07.2013.

contributions by making the work available to users. These types of works mainly rely on the review and comments function that application stores provide to collect feedback on their concepts [116]. Another motivation also might be to disseminate existing work, and make research results available for end-users, maybe aiming to acquire additional insights by deploying the system.

Wang [257] present the design of *Ocarina*, which is a musical instrument for the *iPhone* that uses its touch display and additional sensors for creation of tones. By making their application available on the Apple AppStore, they were able to reach more than a million users, which allowed them to investigate users' social experiences of their application.

Buddharaju [51] test the concept of an application for measuring physical activity while walking, leveraging proxy measures for metabolic measurements. By deploying the application to the Apple AppStore and collecting data on the body mass index of users and daily patterns of physical activity, they were able to argue for the reasonability of their concept.

Zhai et al. [267] present *ShapeWriter* as a *"transfer of user interface research to end-user practice"*, which is an implementation of research on keyboard interaction. ShapeWriter is an early example of how application stores can be leveraged to get user reviews, and Zhai et al. present the insights they got from analyzing 556 user comments. Similarly, the result of research on an optimized keyboard layout for tablet devices, called *KALQ* [180], was recently deployed to the Android application store.[5] The application already got first user feedback that might be taken into account for subsequent iterations on the design of the keyboard. Riccamboni et al. [204] tested the concept of educational multimedia applications for identification of organisms by releasing applications to the Apple AppStore. Based on their findings, e.g. applications for spatially-limited species are best offered for free, they adapted their marketing strategy.

Testing early versions of applications as proof of concept can precede other ways of leveraging an application store, before continuing a line of work, to get first feedback from users.

[5] Google Play Store: *KALQ Keyboard (Official) Beta*, http://goo.gl/e49Pq, last accessed on 12.06.2013.

Using Application Stores as Research Instruments

Application stores can also be leveraged as a tool to the inform design of a system following a user-centered approach, as the following papers suggest. As such, works of this category focus on improving the design of a system rather than investigating dedicated research questions.

McMillan et al. [165] describe how they collected a maximum number of participants combining qualitative and quantitative feedback for redesign of a game called *Yoshi*. Their approach was to inform the design of the application itself based on a large amount of feedback collected from end users through in-game feedback measures and by contacting some players through social networks and interviewing them over VoIP or telephone.

Karpischek et al. [143] deployed an application for sharing product reviews based on barcode scans. First user comments were analyzed to improve the design of the system. The authors describe their deployment on the application store as a research tool to learn about customer-product interactions.

Henze and Pielot [116] discuss the research tool of studies through the application store as a means for increasing external validity, simply because large samples of world-wide participants, who will use the application in their natural contexts, can be reached.

Discussions about ethical considerations (e.g. [113, 166, 172, 187]) of using mobile application stores also fall into this category of thinking about application stores themselves as a research tool.

Implementing Dedicated Research Questions

Another way to leverage application stores for research studies is to conduct studies which alternatively also could have been conducted within controlled lab studies. As such, this category's goal of using an application store is to study a distinct research question, with special focus on upscaling the size of a sample, reaching a wider range of participants, or bringing the apparatus into the natural context of the participants.

A good example for studies whose research questions are implemented within applications are Henze's studies of off-screen visualizations on small screens [114, 118]. In [114] the study compared stretched arrows, scaled arrows and *Halos*

with a task of finding 10 randomly distributed points on a map. In [118] the focus is more on testing the method. To test such "experiments in the wild [they used] the well-defined off-screen problem", which they already had used previously, but this time with the study task embedded into a game.

Budde et al. [50] study the question of whether a social gaming approach can be used to build a crowdsourced database of product information. During a rather short study period of 17 days, the authors were able to provide evidence that it is possible to motivate users to start submitting information, but the authors could not show that a full database could be created.

In [119] Henze et al. studied the touch performance of smartphones with regard to precision. With a task where users had to tap on elements of different size and position, they replicated a known offset in touch accuracy. Most importantly, they found that this offset is systematically skewed. By releasing an updated version of the game, they showed that the resulting touch error could be compensated for, and touch performance could be improved.

Sharazi et al. [231] investigate whether non-verbal iconic user interfaces are reasonable for real-time opinion sharing in the case of television programs. They released an application for judging football games, and studied its use during a football world championship. Using data from 925 users they revealed that the interface serves its purpose and can even create a sense of connectedness among users in different locations.

Informing an Understanding of Smartphone and Application Usage

Researchers also apply the instrument of research through the application store to learn about more general aspects of smartphones and application usage. As such, in this fourth category we present works which conduct research through the application store because their research questions are inherently bound to this ecosystem. Some of the works presented previously on recommender systems for mobile applications could also be put into this category.

In [255] Watzdorf and Michahelles investigate the accuracy of positioning to understand how accurate such data is, based on data collected through a commercial iPhone application. They find that different positioning technologies (GPS, Wifi-based, Cell-ID) provide different degrees of accuracy. Based on their findings they suggest which positioning technology should be used for which requirements. In

[39] we present an approach to exploit users' smartphones themselves as a study apparatus by making the original smartphone operating system an integral part of the gameplay.

Other works in this category strongly relate to the overall topic of this thesis (cf. [103, 104, 85, 115, 191, 247, 93]), and will therefore be introduced in Section 2.3, which describes work relating to the research questions rather than to the method.

Informing the Research Approach

Lessons learned constitute the main contributions of a few papers conducted using the approach of research in the large. As such, they inform the approach itself and provide guidelines for other researchers using this approach.

Henze et al. [117] report on the experience of five studies they have conducted in the large. One valuable insight is that findings cannot be generalized without knowing much about the sample, e.g. when they people reside only in one country. Secondly, when users have to opt-out from the study instead of opting-in, obviously more data can be collected, though this has to be aligned with legal and ethical aspects. Further, Henze et al. found that it is important to engage users for long-term usage since drop-out rates are high, and qualitative feedback received through market comments is mostly useless. Finally, they note that applications put on application stores will often be used in unforeseen ways.

Kranz et al. [151] study the adoption of near-field communication (NFC) technology (they released a game that motivates people to scan NFC tags they have available) and investigate people's behavior with respect to updating applications. The main lessons are that short development cycles can support fast iterations on user feedback, visual appeal does attract users, marketing and maintenance is required to turn downloads into active use, multiple applications for the same research question could increase the sample size, and studies in the large are conditioned by user-side constraints (e.g. whether NFC is available).

In [81] Dey et al. present lessons learned from a study of people's battery charging habits observed over a 4-week study of more than 4000 users. They describe how they improved the application based on user feedback. The paper discusses ideas for running controlled studies by randomizing conditions based on device identifiers, and argues that maintaining a deployed system would be well supported

because update cycles can be short. A downside that was experienced was that engineering efforts tend to be high when aiming for good reviews, which will impact the visibility of research applications among the large number of applications. Therefore time and effort to be spent for developing and running research in the large should be carefully planned.

Miluzzo et al. [170] describe the *"deploy-use-refine approach"* that they have adopted. Their strategy creates a circular iterative process similar to the idea of deployment-based research introduced earlier. In addition to aspects already mentioned in works previously discussed, they add that users need an incentive for using a research application and software limitations need to be considered, since APIs (especially for sensors) are not necessarily designed for research purposes. Most importantly, the authors add that when conducting research through the application store, the researchers lose the ground truth of traced data; they propose to add elements of experience sampling or to cross-validate collected data with participants' data from status messages on social networks.

Tossel et al. [247] explain nine constraints that need to be taken into account when using the method of smartphone logging for studies in the wild. These are related to the variables that are needed, whether data is potentially sensitive and requires privacy, the degree of obtrusiveness and whether the user will be interrupted for data collection, whether an interface is required for logging, whether participants' tasks will be natural or constructed artificial tasks, the type of technology used, who the participants are, where the study will take place and what the setting will be, and finally how long the study will be. The authors compare three studies they have conducted, and conclude that logging can be more accurate than self-reporting, e.g. about the amount of time people spend in applications.

Last but not least, Coulton and Bamford [71] report on experiences with applications released to the *WidSet*[6] platform, with two deployed applications with a total of more than 1.4 million users. Data collection was begun in October 2007, and — to the best of our knowledge — this constitutes the first and largest conducted study to be reported. Their lessons are that while value-added functionality can amplify popularity and usage, it may also impact usage behavior in a way that impacts the study. To the best of our knowledge, this is the only work reporting on the risk that the evolution of the chosen application store may strongly impact the

[6] An early application store run by Nokia until 2009; see http://en.wikipedia.org/wiki/WidSets, last accessed on 12.06.2013.

research plan, as promotional actions might boost application installs; finally, the *WidSet* application store was taken offline, which put an end to the study — such issues are obviously out of the researchers' control.

We cannot claim that in these five categories we present an exhaustive list of all works relating to the method of conducting research through the application store, but rather we wanted to exemplify the ways the method can be leveraged. As the approach laid out in this chapter is rather young and still evolving, the papers on the lessons learned in particular were published after the studies for this dissertation had been conducted. However, most important practices that are suggested are rather obvious and have been considered nonetheless (e.g. engaging long-term usage, providing value as incentive, iterative design based on immediate feedback). Further readings can be found in the workshop proceedings of related workshops such as *Workshop on Observing the Mobile User Experience*[7], the Workshop series on Research in the Large[8], and *Workshop on Informing Future Designs via Large-scale Research Methods and Big Data*[9].

2.2.3 Methodically Related Frameworks and Datasets

In Chapter 3 we will present our means for studying mobile application usage in the large. Essentially, we will investigate the lifecycle of mobile applications from an end-user's perspective and implement and deploy a logging framework to trace mobile application usage. This is not a new approach as such, and related systems are available to study user behavior on smartphones or — more generally — in instrumented ubiquitous computing environments.

Datasets

Some datasets are available for researchers to investigate mobile application usage, three of which shall be discussed shortly: the *Reality Mining Project*, the *LiveLab traces*[10] from Rice University and the *Lausanne Data Collection Campaign*[11] from

[7] http://omue10.offis.de/files/OMUE10-Proceedings.pdf, last accessed on 12.06.2013.
[8] http://large.mobilelifecentre.org, last accessed on 12.06.2013.
[9] http://informdesign.ubiplayground.com/, last accessed on 12.06.2013.
[10] Rice University: *LiveLab: Measuring wireless networks and smartphone users in the field* , http://livelab.recg.rice.edu/traces.html, last accessed on 13.06.2013.
[11] Nokia Research Center: *Lausanne Data Collection Campaign*, http://research.nokia.com/page/11367, last accessed on 13.06.2013.

Nokia. Analysis and results of this data related to this thesis will be introduced in Section 2.3 (e.g. [226, 85, 86]).

The *Reality Mining Project* [89] aims to investigate social phenomena through empirical data collection. The project instruments mobile phones with rich sensing capabilities and distributes them to 100 participants and collects data over a period of 9 months. Based on the pieces of data the researchers collect from participants' phones, for instance direct Bluetooth information, they can investigate people's usage of devices, along with the relationships between people, and they can also reason about individual as well as group behavior.

The *LiveLab traces* collected by Shepard et al. [226] provide data on smartphone use over one year. The authors had to apply unusual software modifications to get access to the information on iPhone, which prevented them from deploying the system to an application store, and therefore they could only reach 25 participants, recruited from university students, in their study.

For the *Lausanne Data Collection Campaign* a Nokia research lab recruited about 170 participants for over one year [146], later extended to 200 people. The logging focuses on social interaction between people (e.g. phone calls) and spatial behavior related to movement. Participants were virally recruited using a snowball approach, and asked to use an instrumented mobile phone as their primary mobile phone. For a full set of resulting papers we refer the reader to the project's website[12].

In contrast to such datasets that are partly available to the public and the data we will describe in this thesis, besides the focus of the investigation (application usage in our data), publicly available datasets are limited in scale and only reach users from local areas. In addition, providing participants with instrumented devices might bias their natural usage behavior (see e.g. [171]). Further, we cannot release our data for ethical reasons as it provides very sensitive information about individuals. Even though we collect only anonymous data, it is so far unknown whether parts of our data or specific features can be used to deduce the identity of individuals. However, we do provide access to the data in a controlled way.[13]

[12] See http://research.nokia.com/page/11374, last accessed on 13.06.2013.
[13] See [139, 140, 157, 183, 228] for results of such collaborations.

Frameworks

In Chapter 3 we will introduce a framework for tracing mobile application usage. With the improved capabilities of smartphones, different frameworks for different purposes have been developed; these can be used for collecting various sensor information on users' devices.

Raento et al. [192] present *ContextPhone*, which is an extendable framework published as open source. The framework focuses on logging contextual information on smartphones (e.g. related to communication, application usage, and location) and provision and storage of such data. The platform was extended for different purposes and used within different research projects, e.g. *ContextLogger*[14] to investigate the reasons for unavailability for mobile phone calls [212].

The *AWARE* framework[15] is similar to *ContextPhone* in the sense that it provides an open extendable platform. The main difference is the target platform: while *ContextPhone* was built for Symbian OS[16] and the Nokia Series 60 platform[17], *AWARE* is built for the current generation of smartphones running with Android OS. Studies using *AWARE* (cf. [83, 81, 93]) will be described in Section 2.3.

In a similar line of work, also the *funf Open Sensing Platform*[18] was developed by MIT. It resulted out of the Reality Mining project [89]. Like *AWARE*, *funf* provides an open-source extensible approach for collecting data on users' smartphones. *funf* is a little bit more established as it provides support for easy use, e.g. though data collection in the cloud.

Rawassizadeh et al. [199] present the *UbiqLog* framework for life logging applications. *UbiqLog* logs events such as application use or phone calls with their relation to location and time. The authors evaluated their approach with an Android-based implementation for usability and resource efficiency (e.g. CPU usage) with six users over one to fourteen months.

Wagner et al. [256] present *Device Analyzer*: An application they implemented for the Android platform to collect more than 200 different pieces of information from end-users' smartphones; e.g., WiFi connection, installed applications, system characteristics, and information regarding telephony. Over the course of nearly 2

[14] See http://www.contextlogger.org, last accessed on 13.06.2013.
[15] See http://www.awareframework.com/, last accessed on 13.06.2013.
[16] See http://www.symbian.com, last accessed 13.06.2013.
[17] Wikipedia: *S60 (software platform)*, http://goo.gl/8sqv1, last accessed on 13.06.2013.
[18] See http://funf.org/, last accessed on 13.06.2013.

years they were able to collect data from 12,500 users. They present their data processing pipeline and discuss privacy issues.

Related platforms are typically designed to be exhaustive in tracing and measuring basically everything that can possibly accessed on a smartphone. However, it is known that due to software limitations, not every piece of information can be accessed with the same level of accuracy [170]. Therefore, for the framework that we are developing in this thesis, our design goal was instead to focus and trace application usage as accurately as possible.

2.3 Related Work

In Section 1.3 we introduced the research questions of this thesis. Work related to these questions can be found in five different areas: First we will provide an overview on literature on social and communicational use of mobile phones in general (Section 2.3.1), and then explain related studies and systems as they apply to the four areas of this thesis: mobile usage of the Internet and applications (Section 2.3.2), housekeeping of mobile applications (Section 2.3.3), context-aware recommender systems for discovery (Section 2.3.4), and multitasking and task interruptions (Section 2.3.5).

2.3.1 General Mobile Phone Use

The work presented in this thesis contributes to the general understanding of mobile phones and their usage. In the first place, mobile devices are phones, although they became smart as Want [259] describes. Thus, we will give an overview on general aspects of personal and socio-cultural studies dealing with mobile phones, and present works that reflect on how mobile phones shaped our habits and how we customize our devices.

Personal and Socio-cultural Aspects

Want [259] discusses an era when our primary computers that we use for work will be mobile phones. He discusses technological trends like integration of the previously separate communication processor and application processor into one chip. He differentiates smartphones from phones as devices whose processors became computationally powerful enough to execute general purpose applications.

In [258] Want discusses the metaphor of cell phones as *personal proxies* for their owners. Customizations such as like personal ringtones would say something about the owner's personality, as physical sensors would give rich information about the owner's experiences. Further analysis of the phone's sensory data would make it possible to infer information about its user's activity. Want concludes that mobile phones became a multifarious proxy for their owners, enabling logging and tracking of their life.

Dey et al. [83] investigate the proximity of people to their smartphones in daily life. They collected interviews and logging data from 29 people over four weeks, and surveyed additional 367 smartphone users about their smartphone habits. They found that smartphones are kept within arm's reach 53% of the time, and in the same room (including within arm's reach) 88% of the time, although the users themselves perceived their phones to be within arm's reach 91% of the time. Based on their data, they created a model to predict phone proximity. Most interestingly, they found that people keep smartphones within arm's reach at the same rate as they kept previous phone models within the same distance, but they keep smartphones in the same room more often than they did devices of previous phone generations.

In [195] Rahmati et al. elaborate on the social and economic status of smartphone users and their behavior. They find that such status has a significant impact on usage behavior, e.g. with people of lower socio-economic status spending more time on the device and spending more money for applications. Participants of lowest socio-economic status judged iPhone usability to be poorest (mostly due to battery lifetime). The authors conclude that for different socio-economic user groups, different phone designs should be available.

Lang et al. [154] also investigate mobile phone use by people with lower social status. They conducted an ethnographic study in China to look into the phone usage of young migrant workers. The authors found that their 26 study participants used their phones to create, maintain and enhance social ties with both local and remote contacts, e.g. sharing fun content with acquaintances or staying in contact with people from their hometowns.

Harmon and Mazmanian [110] study the cultural discourse of smartphone use in a broader sense of American culture. Their sources of evidence are phone advertisements and news articles collected from magazines through the years 2002–2010. The authors unpack two distinct idealized tropes: integrating smartphones into everyday life vs. dis-integrating smartphones from everyday life. Through ethno-

graphic fieldwork, the authors find that these two overly-simplistic concepts reveal conflicts, tensions and instability related to people's values, technologies and everyday life.

Mehrotra et al. [169] study usage of mobile phones in Rwanda, Africa, with a focus on differences between women and men regarding communication. The main data source for their investigation is a dataset on mobile phone activity from telecommunication providers containing demographic information. The study reveals that while men overall have higher levels of phone usage, women have higher usage during evening hours and at special events. Despite concerns about the external validity of the study due to locality, it still provides interesting insights about the ways that societal phenomena (e.g. Christmas, Valentine's day, politics) can impact mobile phone usage behavior.

Habits of Mobile Phone Usage

Oulasvirta et al. [179] investigate habitual patterns of smartphone usage. They engage with data of three studies and describe a checking habit: a *"brief, repetitive inspection of dynamic content quickly accessible"* on users' smartphones which contributes a large part of overall usage, and which may increase phone use in general. The authors describe that such a habitual form of smartphone use is triggered by contextual cues, e.g. situations. Although they relate such habits to mental disorders, they leave open whether smartphone habits are addictions or enablers of multitasking; participants reported both positive and negative experiences. Relatedly, King et al. [145] investigate the disorder of people being afraid of having lost their phones, called Nomophobia (no-more phone phobia). They present a case study and consider that that Nomophobia might indicate other social disorders.

Butt et al. [54] relate psychological theory about personality to patterns of mobile phone use. Interestingly, they find that disagreeable extraverts spent more time customizing their phones' appearance. Chittaranjan et al. [60] contribute to the same line of research, and investigate the personality traits that can be concluded from smartphone logging data. The authors analyze logging data from 117 users (call, SMS, application usage, Bluetooth, profile settings) and self-reported Big-Five personality traits, and find correlations between these measures. For instance, extraversion positively correlates with use of office and calendar applications, and negatively with game usage, and extraverts also spent more time on incoming

(a) By Häkkilä and Chatfield [109] (b) By Tossel et al. [246]

Figure 2.1: Personalization elements of mobile phones as taken into account in related work.

phone calls. Comparing genders, the paper reports that men are more likely to use applications like games, video and office applications.

Häkkilä and Chatfield [109] investigate the different ways people customize their mobile phones and study different properties that people modify, such as audio settings and look and feel like those shown in Figure 2.1. Their study of 60 participants reveals that customization is a highly relevant way of personalizing devices. Häkkilä and Chatfield were able to distinguish temporal patterns: While modifications of device look and feel are early personalizations, functional settings (e.g. short cuts and quick dial keys) are changed over the long term, and even more complicated features (e.g. access point settings) are often left unchanged. Also Tossel et al. [246] investigate personalization of smartphones, and look for gender differences, and relations between personalization, device use and perceived usability. They develop a score to assess a user's personalization of his device (for iPhone only). This score for instance takes into account the number of applications installed, moves of icons from the quick launch menu and on the first page of the menu, adding an phone case, and setting lock screen image. They collect data from iPhone users and find that people who personalize their phones perceive it as more usable. They also report that women customize their phones differently than men; e.g., females are more concerned about appearance. However, they do not qualitatively investigate the concepts that people use for arranging icons; this is what we do in Chapter 4.

In [196] Rahmati and Zhong present a four-month study on smartphone usage by 14 teenage novice users with little or no experience with mobile phones. The study took place in 2007 and authors traced mobile phone usage (power, display status,

WiFi for determining locations) and qualitative data from semi-structured focus groups. The authors investigated the adoption of the phone and found, for instance, that the novices quickly learned to use games and music players. As such, they describe the evolution of phone use during the study and find that personalization, in terms of storing personal data like pictures, takes time and can have a significant effect; for example, sharing of the device was limited to close friends.

Ferreira et al. [93] study smartphone charging behavior through a study in the large. Based on data from more than 4000 people collected using the *AWARE* framework (see Section 2.2.3) they found that people charge their phones in long periods over night, and short bursts during the day, while in the latter case people use USB charging (connected to a computer via USB) more often than a AC power outlet. On average, people leave their devices plugged in for 4 hours and 39 minutes after the battery is full. However, authors also reveal that individual users have unique battery charging behaviors.

Kujala and Miron-Shatz [153] study the role of emotions and memories with regard to usability and user-experience in mobile phone use. They collect longitudinal data on 22 users of different phone models by surveying them at five different times over a 5-month timeframe. The authors became convinced that positive emotions mostly evoke good user experience and negative ones result in low usability. More interestingly, they found that in first-time device ownership people overestimate their positive emotions and mainly focus on their use experience, whereas the importance of usability increases over time.

Suzuki et al. [239] study the relation of a user's familiarity with phone models on task performance. They found that when users are more familiar with their phones, their perceived usability of the phone correlates more strongly with performance.

2.3.2 Mobile Web and Application Usage

The more ubiquitous and popular smartphones and mobile applications became in recent years (see Chapter 1), the more important it became to understand the principles of mobile application usage. In particular, an understanding with regard to the mobile users' contexts and changing tasks [18] is of interest. However, since people were able to access mobile services through websites on their phones before mobile application stores became popular, we will first present works on

understanding mobile web access that shall support our understanding of mobile application usage and needs.

Studies of Mobile Web Access and Mobile Search

Mobile information needs can be seen as a more general case of mobile application needs, since these also provide information to users (e.g., news and weather forecasts). Due to the special nature of people's mobile information needs, their access to mobile information by leveraging search engines has been studied thoroughly (see e.g. [67, 242, 232]), but we can also build on studies of web use that show that context is a salient concept when describing usage (see e.g. [62, 74, 176]).

Cui and Roto [74] investigated how people use the mobile web based on contextual inquiries of 47 participants and URL access logging data of 547 phone users. They found that web use is influenced by spatial, temporal and social factors as well as means of web access, e.g. the timeframe of web sessions is rather short in general but browser use is longer if users are connected to a WiFi network. Nylander et al. [176] describe a study of 19 people using diaries and interviews. They found that the most frequent location from which people access the Internet with their mobile phones is at home. In [175] Nylander et al. further found that people find Internet access using their mobile phones most convenient at home. But interestingly functionality used outside the home, e.g. messaging or picture taking, would create usage behavior that people then also carry out at home. Lee et al. [155] study the usage of mobile internet services by 75 Korean participants over a 1-month period. In their exploratory study, they investigate the types of services used, the contexts of use, and relations between both. Interestingly they found that usage is mainly limited to only a few services (9 services account for 50% of mobile service usage), but these services differ from those used on stationary computers. Further, the authors also find that mobile service usage is limited to specific contexts (e.g., at public places, when not moving, and when not working). To address a specific context, Sohn et al. [232] also study people's information needs while on the go. They used the experience sampling method to examine the mobile information needs of 20 people over two weeks. Mostly, people were interested in trivia knowledge, directions and points of interest. The authors also found that people multitask to satisfy urgent information needs (e.g., using the phone while driving a car). Some information needs (42%) may be postponed or remain unmet due to attentional costs and contextual factors.

More specifically Church et al. [63, 64, 65, 66] look into mobile information access and how people search for information on their mobile devices: Early work [68] analyzes a large corpus of mobile web search (6 million search requests by 260,000 users over one week) and discovers that mobile search topics differ from stationary search, although adult content is a popular topic. A diary study of 18 users over 4 weeks [65] revealed that use of mobile web search in stationary contexts at home and office is increasing, but when used in a mobile context location, time, activity and social surroundings have a strong impact. Another diary study [66, 67] found that more than 30% of search serves non-informational and geographical needs, and users mostly conduct mobile searches when they are away from familiar contexts. In [64] Church et al. elaborate on the social aspects of mobile search by conducting a 200 person survey and a 20-user diary study over two weeks. Their findings are that social mobile search is more likely to happen in unfamiliar areas and is motivated by curiosity and assisting an activity. Church et al. [63] also present an approach for understanding mobile users' information needs at large scale. Their study collects data from more than 100 participants over 3 months, leveraging experience sampling and online diaries triggered by SMS. They describe their approach and methods: They asked their participants for their daily routines beforehand, and investigated participants' information needs by scheduling SMS surveys that the participants would answer accordingly. A web survey asked for additional information to clarify the SMS responses (e.g. about the participants' companies and their location at the the information was needed).

A similar line of work is done by Teevan et al. [242], who surveyed 929 people about their mobile local search. They also report that time, location and people's social context influence how people search on their mobile. Most interestingly, they found that queries not only relate to the users' actual locations but also to their travel destination and route. Hinze et al. [122] also investigate mobile search queries. Based on a diary study of 12 participants over one week, they find that types of questions people have are influenced by their location (e.g. home, work, with friends), but also a large portion of questions is unrelated to context. Finally, Chua et al. [62] conducted a one-week diary study with 20 participants to learn about mobile search, and among other findings they confirm that mobile information needs are triggered by contextual factors like location, intended activity and social surroundings.

Summarizing studies of mobile information needs and information consumption, we can conclude that mobile information access is governed by the users' contexts

(e.g., the impact of location, time and social context [242, 122]). Mobile access to information on the Internet was available before information access through mobile applications became popular. Therefore, we can learn from the studies of mobile information access to inform our understanding of mobile application usage, especially with regard to context of usage.

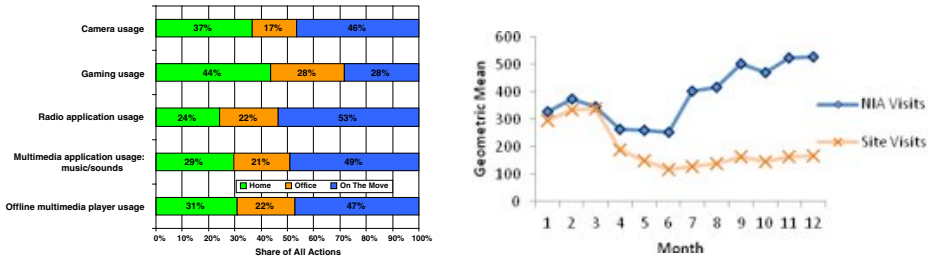
Studies of Mobile Application Usage

Pioneering work by Verkasalo [251], Froehlich et al. [100] and Demieux and Losguin [79] investigated mobile application usage on the previous generations of mobile phones not providing application stores for users to install applications, and was done on a small scale.

Verkasalo [251] showed that people use certain types of mobile services in certain contexts. He analyzes data from several hundred mobile phone users over 2–3 months concerning their locations and smartphone usage. Based on a heuristic, he correlates the location and the time of the application usage in the recorded data. Based on this, for example, Verkasalo finds that people mostly use browsers and multimedia services when they are on the move, but play more games while they are at home; other examples are shown in Figure 2.2(a). Froehlich et al. [100] presented a system that collects real usage data on mobile phones by keeping track of more than 140 types of events. They provide a method for mobile experience sampling and describe a system for gathering in-situ data on a user's device. The goal of Demieux and Losguin [79] was to collect objective data on the usage and interactions with mobile phones to incorporate the findings into the design process. Their framework is capable of tracking the high-level functionality of phones, e.g. calling, playing games, and downloading external programs.

Verkasalo et al. [252] study the intent behind using or not using mobile services; in particular, they look into the three most-used applications of their sample (web browsing, maps and games). The authors collect device logging data from 579 participants and run a web-based survey. They find that perceived technological barriers impact people's confidence in the possibility of task fulfillment with advanced mobile services. Further, the the authors explain that use of games has a hedonic usage intent, while maps and the mobile web have a more utilitarian motivation.

With the growing ecosystem of mobile applications as described in Chapter 1, there also is growing body of research on how people use mobile applications, to which



(a) Verkasalo et al. [251] (b) Tossel et al. [245]

Figure 2.2: Works investigating mobile phone use: (a) Share of application usage by context for different applications; (b) native applications with Internet access (NIAs) are used more that web access through browsers.

the work presented in this thesis has also contributed. One way to study application usage on smartphones on a large scale is to analyze the network communication that those applications cause. This opportunity gives insights into such applications that require network communication — which is an extensive but not exhaustive portion. Xu et al. [263] analyzed web access data that was done through native mobile applications, which was provided by a US network carrier. They found diurnal patterns of application usage (similar to those we will report in Chapter 3; e.g. news applications are popular in the morning and sports applications are more popular in the evening). They also found that a large fraction (20%) of available applications have a rather local use restricted to a few areas. From user traffic they can also see that the opening of one application can predict the usage of another one, and that users have alternative applications for the same purpose, e.g. news sources. Xu et al. discuss ideas for optimization of network traffic rather than for the benefit of the user herself. Tossel et al. [245] study the web use using the LiveLab traces (introduced in Section 2.2.3) and investigate the use of web browsers and native internet applications. Compared to browser usage on the PC, they find that mobile browser usage is not as fundamental as it is on the PC, and on smartphones people instead use native applications to access Internet content, as Figure 2.2(b) reveals. Further, their work replicates some of our results in Chapter 3, e.g. that application use per launch is rather short, with an average duration around one minute. They also report that about 15% of website re-visits are done after an interruption by SMS or phone call. Tossel et al. propose design optimizations like designing launchers for better recommendation of new applications by exploiting web visit history.

Mallat et al. [160] investigate the impact of context on the adoption of mobile services. Their conclusions are based on the case study of a mobile ticketing service.

Results of their study are that the contexts in which mobile services are used impact the service's benefit, and context as such determines a person's intent to use the mobile ticketing service. Mallat et al. conclude that it is important to deduce contexts in which a particular service is likely to be used, and provide functionality that is valuable for users in those particular contexts.

Falaki et al. [91] study the use of smartphones under the four different aspects of user interaction, application use, network traffic and energy drain, with special focus on the latter two. The authors analyze traces of 255 users ranging over periods of 7–28 weeks, which had been collected by instrumenting Android devices and Windows Phones. Among others, their main finding is that smartphone usage differs immensely between different users, though they provide evidence that different user behaviors can be described by a single model. They conclude that mechanisms to improve user experience should be personalized. Further, they argue that demographic factors are only an unreliable predictor of user behavior. In [90] Falaki et al. specifically look into networking aspects of smartphone usage like throughput and packet loss based on low-level networking layers and protocols.

Do et al. [85] investigate mobile application usage trails of 77 users of Nokia N95 phones over periods of 9 months in a European city (see *Lausanne Data Collection* in Section 2.2.3). Essentially, the contextualized data set they collected by means of a background logging service contained application usage logs, location data, and Bluetooth data. Among other results, they find that application usage correlates with users' semantic locations; e.g. at holiday locations people are likely to use their camera and map applications, and at transport-related locations people will likely use their clock and camera applications. With regard to Bluetooth scans as a rough proxy for users' social surroundings, the authors show that the clock application has lower usage in dense Bluetooth environments; that short messaging, voice calls and web usage increases in dense Bluetooth areas; and that usage of email increases at both extremes of Bluetooth surroundings. The authors present implications for contextual design of phones. Similarly, Do and Gatica-Perez [86] analyze patterns of mobile application usage based on a sample of more than 230,000 hours of application usage provided by 111 people (see *Lausanne Data Collection Campaign* in Section 2.2.3). Based on this data, they verify a model for recognizing patterns in daily application usage, and for describing user behavior based on the patterns found.

Rahmati et al. [194] present results of a longitudinal study of 34 iPhone users over one year maximum. To collect data, they instrumented participants phones with logging software and conducted interviews. They find that smartphone usage changes over the year and that users can be clustered by similarities and differences in application usage into classes of users.

Henze and Boll [115] investigate at what times people install games from the Android application market. They analyze data from more than 157,000 users and observe installations of 24,600 other games. In particular, the authors compare the usage of their own game application to deployments of other applications on the market, both with regard to time. Their data reveals that the likelihood a new application on the market will be launched directly is highest between 11pm and 5am. From this the paper derives advice for researchers as to what time to release an application to the market.

Brown et al. [45] present a video-based approach for investigating mobile application usage by leveraging screen capturing and wearable cameras. They collect nearly 24 hours of 15 participants' (5 individuals, 5 couples) activities and four hours of iPhone usage observing the use of a range of applications. Most interestingly, their method also reveals switches between applications, such as a browser with search engine and a maps application. Further, the authors conclude that mobile application usage is threaded with other user activities.

Möller et al. [171] use logging to investigate the accuracy of self-reporting of mobile application usage in a six-week study. They developed a self-reporting tool that would provide three different ways to ask for reports: voluntary (participants are reminded to report once before study), interval-based (participants are reminded to report daily), and event-based (participants are reminded to report directly after each application launch). In particular, they queried about the usage of Facebook and the mail client and found that in general participants underestimate their application usage, down to only 40% of real usage. Further, Möller et al. found that asking for reports biased the actual application usage behavior which they had logged.

Our work on investigation of mobile application usage (see Chapter 3) is unique in that it combines the approach of large-scale, in-the-wild user studies through the application store with the fine-grained measuring of application usage. In this way, we are able to (1) study large numbers of users and (2) large numbers of applications, all over (3) a long time period. Previous work has had to make sacrifices in

at least one of these dimensions. Furthermore, the mobile phones used in related studies have been mostly from the last generation, without application stores, i.e. they could not be customized by the end-users in terms of installing new applications.

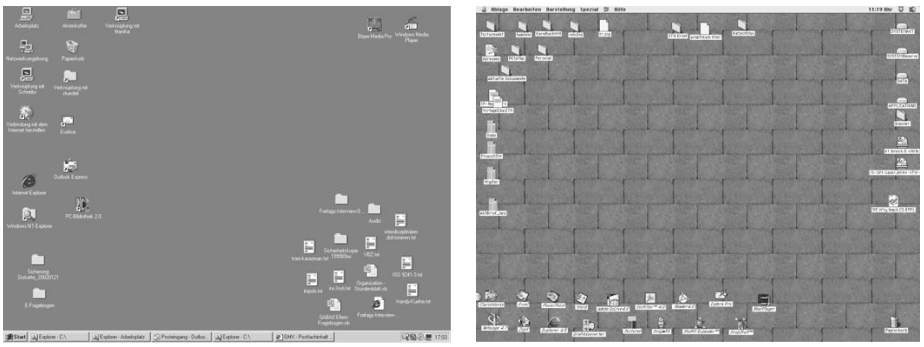
2.3.3 Menus for Launching Mobile Applications

This thesis deals with adaptive and adaptable menus for launching mobile applications. Findlater and McGrenere [94] define both variants of menus in terms of who has control over the adaptation of menus: For adaptable menus, the user is in control of customizing the menu; by contrast, with adaptive menus the system is in control of the adaptation. In Chapter 3 we support users to more efficiently launch their applications on their smartphones by designing an adaptive menu which change the set of icons that they show. In Chapter 4 we investigate the concepts people use for adapting menus on their smartphones. In this section, we will present works related to both understanding how people adapt menus themselves, and adaptive menus for supporting item finding.

Users Adapting Menus

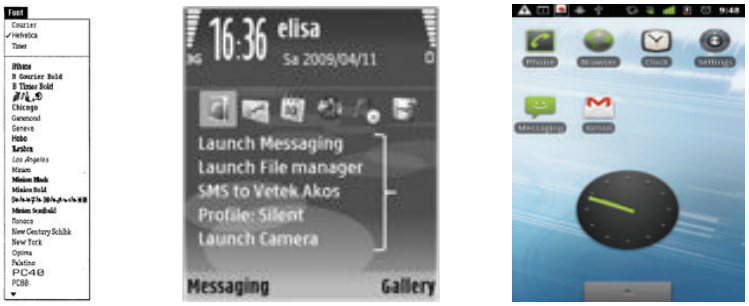
Pioneering work on users' practices for adapting menus in terms of organizing icons and digital information in general was done in the stationary desktop area: Barreau and Nardi [19] summarized their studies of file organization on personal computers. They found that a visual search for files based on location is preferred over text-based search, that people put icons at special places as a reminder, and that information can be categorized as ephemeral, frequently-used, or archived. Ravasio et al. [198] investigated habits and problems during document classification and retrieval. Among other findings, they show that people cluster documents by their types, and that people use different desktop areas for different purposes, as shown in Figure 2.3. Shipman et al. [230] investigated the implicit structure that humans implement in layouts when manipulating icons or other visual objects. They propose to parse and exploit this structure for assistive facilities. In this paper we go beyond desktops and set out to explore people's practices for arranging icons on smartphones.

Current research on the recent generation of mobile applications is mostly focused on how people use applications by means of installing or executing them



(a) Microsoft Windows PC (b) Mac desktop

Figure 2.3: Examples of customized desktop screens [198].



(a) Split menu [224] (b) SmartActions [253] (c) Dynamic home screen [229]

Figure 2.4: Examples of adaptive non-mobile and mobile menus from related work.

(cf. Section 2.3.2). So far, there is little public work available on how people customize their menus and organize applications on their devices, and how they can be supported in doing this — aside from the published results of this work (see [30, 31, 40]).

Systems Adapting and Optimizing Menus

Supporting users with faster access to phone functionality has been a topic of previous research on phone usage (esp. [138, 253, 265, 229, 268]). Inherent to creating adaptive mobile menus is the requirement of modeling the usage of the items in the menu, especially predicting application usage.

In pioneering work Sears and Shneidermann [224] introduced split menus. Their motivation is to improve selection performance in long menus where only a small subset is used frequently, and therefore propose to distinguish an area with fre-

quently used items from an area with infrequently used items. The authors conduct studies to provide evidence for the advantage of split menus, both in terms of performance and user preference They derive guidelines and formulate a model to estimate the benefit of moving items into a split menu. An example of a split menu is shown in Figure 2.4(a).

Bridle and McCreath [44] have investigated shortcuts for mobile phone UIs that can be adaptively injected into users' main menus. They evaluated different approaches to provide users with shortcuts to common tasks on their mobile devices, e.g. calling a specific person or sending a specific text message. The presented simulation-based study is conducted on data on mobile phone usage. They found that a hybrid approach presenting most-frequently-used communication options mixed with naive-Bayes filtered options works best.

Vetek et al. [253] present SmartActions: a context-aware menu that automatically creates shortcuts to phone functionality. The shortcuts appear as textual links directly on the homescreen of a user's phone, as Figure 2.4(b) shows. Vetek et al. implement their system based on unsupervised learning and a clustering algorithm using location (GSM Cell ID) and time and date. The UI provides not only shortcuts to applications, but also shortcuts for user-level actions, like *"(send) SMS to Mary"*. First user feedback for the adaptive UI was positive, though participants had privacy concerns.

Kamisaka et al. [138] investigate the feasibility of prediction operations on a mobile phone based on observable attributes, such as calls, emails, web browsing, other applications, location, signal strength, and battery power. They introduce a priority order as an abstraction layer for all available applications and shortcut icons, which they predict using a machine learning approach. From a study with 19 users over 4 weeks to 8 months, they learn that the time and last action are among the best predictors for application usage, while the location was rather not very useful.

The system for automatic menu customization presented by Fukazawa et al. [101] on the one hand presents content-based recommendations on the user's device, but on the other hand it does not consider the user's context. Interestingly, however, they found that novice users can benefit more from adaptive menus than expert users who have already owned their devices for a longer time.

St. Amant et al. [234] investigate the optimization of hierarchical text-based menus for cell phones that can be traversed by keyboard input. Based on their findings, they propose options for optimizing menu structures that result in reducing traversal time, e.g. by putting commonly used items higher in the hierarchy. Their approach for menu redesign results in time savings of 30% in simulation studies. Also, Matsui and Yamanda [163] presented an algorithm for optimizing the menu structure for hierarchical menus on mobile devices. In their experiment they minimize menu item selection time by changing menu structures. Their approach is based on genetic algorithms and optimizes the menu performance in terms of selection time and for functional similarity and granularity of the menu items.

Do and Gatica-Perez [84] present a model for prediction of a smartphone user's next location and application use based on her current context of location, time, application usage, Bluetooth proximity and communication logs. Based on smartphone logging data of 71 users over 17 months the authors were able to evaluate their model. Interestingly, regarding prediction of application usage, the authors find the preceding application use is the best predictor for the next application.

Yan et al. [265] present an approach for speeding up application launches up to 6 seconds by pre-fetching content from the Internet and pre-launching applications into the operating system. While the authors focus on the optimization of low-level operating system features rather than UI aspects, they also present a model for predicting application launches. Their machine learning driven approach is mainly based on the relation of temporal and spatial variables to application usage and sequences of application launches.

Most recently, Shin et al. [229] presented a model and UI for predicting smartphone applications. In a first study, they collect data in-the-wild to build a model for prediction based on a deployment of an Android application on the application store. Their model, based on naive Bayes, performed best when compared against state-of-the-art algorithms like most-recently-used and frequently-used, and they find that the previous application is the strongest predictor for the next application. In a second, smaller study they found that their study participants used their adaptive application icon menu more often than static launcher menus. The adaptive launcher menu that they are presenting (shown in Figure 2.4(c)), however, only refreshes after the user unlocks her device, and not every time the user goes back to the launcher menu after using an application.

Along the same line of research as Shin et al. [229], Zhang et al. [268] also present an adaptive launcher menu, which they call *Nihao*. They predict application usage based on exploiting a time-, location- and application-aware Bayesian network, and study their approach with a small group of users. They conclude that prediction accuracy is most important for these kinds of menus, since people perform visual search for application icons in fully adaptive menus starting from the top of the screen.

However, none of the recent works on adaptive mobile application menus (cf. e.g. [229, 253, 44, 268, 138]) investigates where (spatially) to place adaptive shortcuts within a mobile menu. This, instead, has been a topic of work on non-mobile menus. For instance, Cockburn et al. [70] present a theoretical model to predict user performance for different desktop menu designs. Based on the Hick-Hyman Law and on Fitts' Law, the model allows for evaluation of different menu layouts before actual implementation. However, we agree with Findlater et al. [95] that findings on classical desktop menus are not necessarily valid for mobile devices. This work will contribute solutions to how adaptive mobile launcher menus should be designed, based on understanding how users themselves customize their mobile launcher menus.

Other Related Aspects on Mobile Menus

Findlater and McGrenere [95] investigate adaptive user interfaces for small screens. In a study with 36 subjects, they compared a small screen with a desktop-sized screen for item selection in a static and two adaptive split menus with different accuracies. During the study, the authors measured performance, awareness of items and subjective measures (difficulty, efficiency, satisfaction, consistency, predictability) for item selection in each menu type. Their findings are that high-accuracy adaptive menus have a larger impact on improvement of selection performance and satisfaction on small screens. On small screens, users are also more likely to use the adaptive parts. However, high adaptive accuracy was also found to negatively impact awareness of menus items. In [94] Findlater and McGrenere report an experiment with 27 participants comparing static, adaptive and adaptable menus within a two-split design. They found that their static menu outperformed their adaptable menu, and their adaptable menu outperformed their adaptive menu in terms of speed. Participants perceived the adaptable one to be most efficient.

The authors conclude that mixed-initiative interfaces, where the system and the user iterate on the optimization of the menu, would be best for user satisfaction.

Ziefle and Bay [270] investigated people's abilities to build mental models of their hierarchical mobile phone menus. They found that younger people have a better mental model of their mobile phones' menus. Further, they also found that awareness of the menu's structure increases navigation performance. In [271] Ziefle et al. particularly investigate young and old users navigating menus on PDAs with or without hyperlinks. From a 20 person study measuring performance when navigating websites, the paper reports that older users were less efficient when hyperlinks were available. Building on mental models of phone menus, Gustafson et al. [108] build an imaginary phone that is operated by mimicking the interaction with a smartphone's interface on the palm of a hand. In this way, they study how well people can transfer learning from a real device to a non-visual interface. Gustafson et al. researched how good people's spatial memory of their phone menus is. They found that their study participants knew the positions of 64% of home-screen icons by heart, and could even more accurately recall the positions of applications used daily at a success rate of 75%.

Kim and Lee [144] investigate the impact of cultural differences on mobile menu interfaces. They found that Koreans preferred a thematically grouped menu and Dutch participants preferred a functionally grouped menu. They argue that this is due to different cognitive styles of the participants, and they propose culturally adapted interfaces.

Ben Lulu and Kuflik [24] present a clustering approach for applications in smartphone launcher menus. Based on the textual descriptions of applications that people have installed, they crawl data from the Internet and perform text processing to extract functional descriptions of applications. Based on these texts, they perform a clustering to create a hierarchical clustered launcher menu of applications. However, they do not discuss how to visually lay out the application clusters.

Bergmann et al. [26] studied the difference in cognitive load between file retrieval by navigation and by search on stationary computers. They conducted a within-subject study with 62 participants on a stationary computer and compared file retrieval by navigation with retrieval by search. The paper reports that retrieval by navigation was faster, mentally less demanding than by search, and less error prone.

Davies and Beeharee [77] study human visual attention on small screen devices. Most interestingly, they study change blindness on mobile grid-based icon menus with different visual disruptions (none, black screen, orientation change, push notification). A study with 29 participants revealed that the more icons a menu shows, the worse the change detection is, and that orientation change has the highest impact on change blindness. The authors propose to reduce simultaneous ongoing activities on mobile screens to decrease disruption.

2.3.4 Context-aware Recommender Systems

Helping people to discover interesting items in a huge set of available items is a problem in different areas. Recommender systems became tools to provide such support. In this work, we will present the design, implementation and evaluation framework of a recommender system that suggests mobile applications. Thus, this section provides related literature on this topic. In particular, we discuss basic work related to context-aware recommendation in general, work related to context-aware recommendation in other domains, and work that relates to our domain of recommendation of mobile applications. We will also introduce the concept of conversion funnels and provide background on user-centric evaluation of recommender systems.

Besides hybrid approaches, *content-based recommender systems* and *collaborative filtering* are the two approaches most often deployed in commercial systems. Ricci gives an introduction to these approaches [206]: The idea of content-based recommendation is to provide a user with recommendations for items which are similar to those she has liked in the past, with similarity being computed based on descriptions of the items. In collaborative-filter-based recommendations, a user is provided with recommendations for items which other similar users have liked in the past, with similarity being computed based on comparisons of the feedback on items between users, and therefore no content description of items is required. Other approaches are based on demographic information about users, domain knowledge, or communities as social networks.

An inherent problem of recommender systems, especially those based on collaborative filtering, is the *cold start problem* [216]: When a new item is available within a recommender system it is difficult for system, to make any substantiated recommendations about the item, because data that would describe the items rela-

tion to users is missing. This problem obviously also arises for new users who join a recommender system and ask for recommendations; in this case the system will not be able to give thorough recommendations.

Adomavicius and Tuzhilin [2] provide an introductory overview on integrating context into recommender systems. In particular, they introduce a multidimensional model for incorporating contextual information into a model that was hitherto only based on the entities of users and items. In particular, they introduce three different paradigms for adding context to algorithms to provide context-aware recommendations. In contextual pre-filtering, the data available for the system is limited to a subset that is relevant for the current context, and then classical techniques are applied to this subset. In contextual post-filtering, the contextual information is added after classical recommender techniques have been applied, and in contextual modeling, the contextual information is directly exploited within the recommender algorithm itself.

Context-aware Recommendation in Related Domains

The requirement of context-awareness within recommender systems has already been used in other domains of mobile computing. Ricci [205] provides an overview on domains where mobile recommendations have been adopted, such as tourist recommendations, news recommendations, and route recommendations. He concludes that eliciting user feedback by leveraging implicit feedback is a reasonable solution, especially when feedback is required related to contexts.

Classic domains of recommender systems (e.g., music [156, 43, 15], movies [210], and consumer goods [254]) are also starting to apply context-aware algorithms to improve the recommendations they provide. For instance, Lee et al. [156] present a music recommender system taking into account contextual information such as the date, region, season, weather conditions, and temperature. The authors' evaluation shows that their system, which recommends music that the most similar users have listened to in the most similar contexts, outperforms approaches based on demographic information and listening patterns alone. Braunhofer et al. [43] present a context-aware music recommender system for tourist guides. The system maps tracks to points of interest based on tags assigned to both sets of music and points of interest, which are adjectives describing emotions. Baltrunas et al. [15] present a context-aware and personalized recommender system for suggesting music in cars. They investigate which context variables (like road type, mood, weather, sleepi-

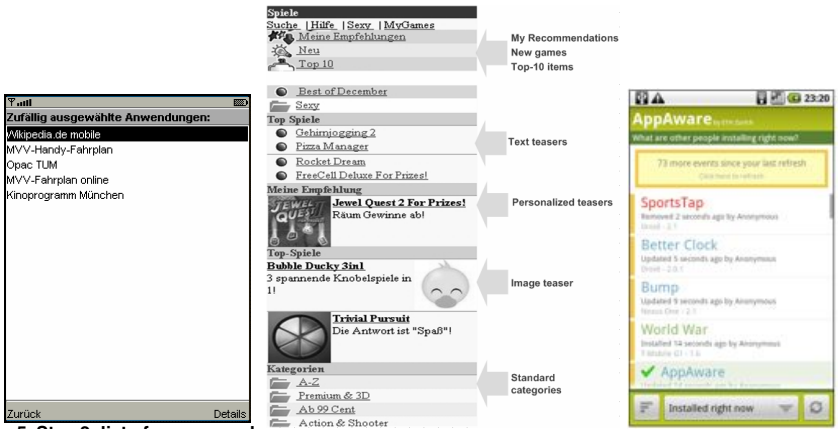
ness) impact the driver's music selection. For instance, they find that for ratings of rock music the traffic condition is an important context factor (e.g. it gets lower ratings in traffic jams, and higher ratings on uncongested roads), and for rating of blues music the driving style is important (e.g. lower ratings in the context of faster and "sportier" driving, higher ratings during more relaxed driving). They take these factors into consideration for building a system to provide recommendations to the car occupants. Said [210] discusses the advantages and opportunities for leveraging contextual information and propose an architecture for building context-aware recommender system for movies. For the domain of shopping, Reischach et al. [254] present a design space for building systems that recommend products. To collect feedback from users about items, they focus on leveraging mobile and ubiquitous computing technologies, especially to facilitate context-sensitive capturing of such data. Gorglione et al. [105] show that leveraging contextual information on the customer's purchase intent and mood can increase customers' trust in the recommendations and increase sales figures.

In the domain of tourist guides, for instance, Belotti et al. [23] present a context-aware recommender system for leisure-time activities on mobile devices. The system design was informed by a field study and takes into account the current time, location, weather, store hours, and users' patterns, and five different kinds of activities (e.g. eating and shopping). The system uses a mix of collaborative filtering, content-based approach, explicitly-stated and learned preferences and other components. Similarly, van Setten et al. [249] present the COMPASS system: a recommender system for tourists that shows points of interest integrating different approaches into a hybrid system.

Also in this line of work, Zhuang et al. [269] present a system that recommends entity types (e.g. local businesses like restaurants and hotels) that users usually would search for on their mobiles by implicitly understanding the user's intent. Their system is informed by the analysis of a large corpus on mobile search queries. The proposed system takes into account contextual information like location and time, and considers also the types of entities. A small user study reveals that the system achieves better user experience than systems asking for explicit feedback.

Recommendation of Mobile Applications

The large variety of mobile applications not only became a practical problem for end-users when they wanted to install new applications; the discovery of mobile



(a) Woerndl et al. [262] (b) Jannach and Hegelich [132] (c) AppAware [103]

Figure 2.5: Screenshots of systems recommending mobile software for download.

applications also became a recent research topic, especially in the field of context-aware recommendation, since mobile application usage is context-dependent (as we will discuss in Chapter 3). However, although the ecosystem of mobile applications is rapidly growing as discussed in Chapter 1, so far there is little research on recommender systems for mobile applications.

Pioneering work before the application store era was done by Woerndl et al. [262] and Jannach and Hegelich [132]. Woerndl et al. [262] present a recommender system for mobile applications that exploits context information and is based on capturing the installations of applications in relation to this context (basically location), though installation times are irrelevant compared to measuring the actual usage.[19] Their recommender engine is based on a hybrid engine following a multi-dimensional approach. Jannach and Hegelich [132] evaluate recommender engines suggesting game applications for downloads on a mobile internet platform. They found that the personalization of recommendations results in an increased number of views and sales. They did not investigate contextual factors at that time.

Girardello et al. [103, 104] present *AppAware*: a recommender system that is based on people's overall application installations, uninstallations and updates. The system recommends applications based on their popularity, i.e. how many times an application was installed but not removed. The system's assumption is that good applications are typically not removed once installed. The *AppAware* system also

[19] Jakob Nielsen's Alertbox: *iPhone Apps Need Low Starting Hurdles*. http://tiny.cc/eyie8, last accessed on 12.06.2013.

exploits social ties by showing which applications have recently been installed by friends. For the end-users the system shows "which applications are hot" by aggregating world-wide occurrences of application installation events.

Yan and Chen [264] present the *AppJoy* system: a recommender system that suggests mobile applications on the *Google Play Market*. It is based on application usage data that is modeled according to the recency, frequency and duration of the application usage. Yan and Chen show that their recommendation approach based on implicit usage data performs well without explicit user input, they evaluated their system by comparing usage times of recommended and not-recommended applications.

Context-aware models for recommendation of mobile applications have recently been presented by Karatzoglou et al. [140] and Shi et al. [228]. Both papers present tensor-factorization-based models that perform better than state of the art context-aware approaches and non-context-aware approaches when using application usage as implicit feedback. The work of Karatzoglou et al. [140] is based on results we present in Chapters 3 and 5, and likewise the work of Shi et al. [228] uses the data corpus we collected in this work for evaluation of their model.

Davidsson and Moritz [76] present the *Applause* system, which is a recommender system that exploits users' locations to recommend applications. In particular, *Applause* tends to exploit context to be able to recommend applications instantly without collecting any additional user data. Shi and Ali [227] present a recommender engine that addresses characteristics of mobile application markets. Their model is based on application-usage as implicit feedback and is specially tailored towards mobile application markets that have heavier heads and longer tails in their distributions. Also Yin et al. [266] present a specialized version of recommender system for mobile applications. Their approach takes into account the satisfaction of a user regarding an application, and how tempted he is to replace an application with a new one providing similar functionality.

Pan et al. [182] present a graph-based approach for predicting the installation of an application on application stores. They extract different networks from smartphone logging of phone calls and Bluetooth proximity, and they collect survey data to construct a social graph. Collecting such data from 55 participants in a study, they show that they can merge different graphs into a graph describing the adoption of mobile applications, which can accurately predict the installation of an application.

More recently, the *Frappe.cc* system (implementing some of the results of [140]) was released as an application to the Google Play Market.[20] The project also deals with context-aware recommender systems that suggest mobile applications and has been deployed to the application store to evaluate the algorithms; however, no results are available yet.

However, none of the related works on recommendation of mobile applications uses a using a user-centric model based on application engagement for evaluation. To the best of our knowledge, so far no comprehensive design space for context-aware recommender systems that suggest mobile applications has been described either. This is what this work will contribute in Chapter 5. Further, we go beyond most other recommender systems for mobile applications by using actual usage of applications instead of download and installation statistics.

User-centric Evaluation

Hayes et al. [112] propose an online evaluation framework for recommender systems that tackles issues related to the dynamically changing conditions in the wild. Instead of measuring absolute engine performance, it performs a comparative measure of how one recommendation strategy performs against another as part of a real world online system used by a community of users. The authors argue that the relative comparison provides both recommender engines with the same resources and that they suffer from the same negative impact of data changes, thus providing fair evaluation in an online scenario. Swearing and Sinha [240] argue that the effectiveness of recommender engines is not solely determined by the quality of the algorithm. Instead they suggest that trust, explanations, serendipity, and user-customizable filters are important factors for effective recommendation. Also, McNee et al. [168] argue that accuracy metrics alone cannot be used to judge usefulness of recommendations and therefore propose new user-centric directions for evaluating recommender systems. They define three aspects of the recommendation process that accuracy metrics cannot measure: similarity of recommendation lists, serendipity, and importance of user needs and expectations. The authors also provide several suggestions for improvement: judging the quality of recommendation lists as a whole, measuring the differences between recommender algorithms beyond ratability, and judging the recommendations for users based on whether or not their needs were met. Pu et al. [190] examine combined criteria for usabil-

[20] *Frappe — a new taste of app discovery*, http://frappe.cc, last accessed 12.06.2013.

ity and satisfaction and conceptualize a unifying recommender system evaluation framework, called *ResQue*. It consists of thirty-two questions and fifteen constructs aimed at measuring qualities of recommended items; systems' usability, usefulness, interface and interaction qualities; users' satisfaction with the systems; and influence of these qualities on users' behavioral intentions. These user-centric questionnaires can be applied to assess different types of recommenders, including rating-based, utility-based, and knowledge-based systems, regardless of the back-end engines used.

In the same line of thinking, Knijnenburg et al. [148] present a framework for user-centric evaluation of recommender systems linking system aspects to user behavior aspects to explain why and how a recommender's user experience evolves. Finally, Konstan and Riedl [150] argue that we need a more diverse set of measures to evaluate the user experience of recommender systems. In this line of thinking, in Chapter 5 of this work we will present a user-centric perspective and its capabilities for the evaluation of recommender engines that suggest mobile applications.

Conversion Funnels

For building up a usage-centric evaluation approach along the lifecycle of mobile applications, we will rely on the concept of *conversion funnels*. A *conversion funnel* describes an action sequence e.g. from showing an advertisement to a user, to the user clicking on the ad, to the user eventually purchasing the advertised product. The paradigm of *conversion funnels* is mostly used in the domains of marketing and advertising. Bagherjeiran et al. [11] present a ranking method for scheduling ads that is optimized for different stages of the conversion funnel. Their model enables a more efficient targeting of ads towards conversions. For online advertising, Becker et al. [20] provide insight into the relationship between landing page types, query classes, and conversion. They conclude that the majority of ad landing pages fall into three distinct classes: home pages, sub pages, and search result pages. Furthermore, the authors correlate these classes with conversion data provided by advertisers to show that the *conversion rate* (i.e., the proportion of certain users who take a predefined, desirable action, such as a purchase, registration, download, etc., as compared to simply page browsing) varies considerably according to these classes. Actions that have some benefit to the advertiser and that happen after the click are called *conversion*; thus, a user is then said to convert from a regular user into a potential business lead or customer [11, 20]. Conversions

representing the final goal of advertisers are also known as *macro conversions*. In order for a conversion to occur, a specific set of events, also known as *micro conversions*, has to occur. These events form the so-called *conversion funnel*. Rosales et al. [209] provide a detailed analysis of conversion rates in the context of non-guaranteed delivery for display advertising. They formalize the problem of predicting the post-click conversion, i.e. conversions after a user clicks on a referring advertisement. The authors further provide fundamental properties of the post-click conversion process based on contextual information, including a comparison between click-through rate and conversion rate. Finally, *conversion rates* measure the proportion of certain users who take a predefined, desirable action, such as a purchase, registration, download, etc., as compared to simply page browsing.

In Chapter 5 we apply the concept of conversion funnels to the domain of mobile application recommendation. The nature of mobile applications — being pieces of software that can be observed during an application's lifecycle (see Chapter 3) — allows us to define an action sequence that relates to engagement with an application.

2.3.5 Multitasking and Task Interruptions

Multitasking can be defined for two different domains: Within the context of systems, multitasking relates to the system's ability to process more than one computational task simultaneously [244]. When talking about humans, multitasking refers *"to the handling of more than one task at the same time by a single person."*[21] In this dissertation, we are interested in HCI-related aspects of multitasking. Related work on multitasking and task interruptions will be explained in five major chunks: multitasking on stationary computers, multitasking while driving a car, multitasking on mobile devices, context-aware telephony and mobile call interruptions. While the former two areas are established as classical fields for studies of multitasking, the latter two are related to mobile phones, which only became ubiquitous in recent years (see Chapter 1). Benbunan-Fich et al. [25] introduce multitasking as people changing between tasks either because of an external interruption or because of a person's internal choice to switch a task.

[21] *"multitasking."* In: New Oxford American Dictionary, edited by Stevenson, Angus, and Christine A. Lindberg.: Oxford University Press, 2010. http://www.oxfordreference.com/view/10.1093/acref/9780195392883.001.0001/m_en_us1269878, last accessed on24.06.2013.

Multitasking on Stationary Computers

Task interruptions have been extensively studied on stationary computers (see, e.g., [25, 128, 214, 134]). We can learn from these studies about the aspects and possible impacts of multitasking and task interruptions when they happen within one system, i.e., the stationary computer in this case.

Iqbal and Bailey [127] are interested in predicting the costs of interruptions. In a Wizard of Oz experiment, they interrupted 12 participants, whose primary tasks were video editing, route planning, and document editing, with the peripheral task of stock trading, and collected the resulting resumption lags. Iqbal and Bailey found that resumption lags can be predicted by taking into account the difficulty of the next task and the amount of information that needs to be maintained during the interruption.

Iqbal and Horvitz [128] conducted a field study by logging the use of applications and the notifications of messengers and emails on stationary computers of 27 people over two weeks. Findings were that users suspended tasks for up to two hours before resuming them, and users justified this by the loss of context due to task switch on interruption. Iqbal and Horvitz derive implications for better design of recovery, like visual cues for finding abandoned windows, thumbnails of suspended applications or playback of the last actions taken before the interruption.

Jin and Dabbish [134] study the phenomenon of self-interruptions while working on a computer, which they also refer to as *internal interruptions*. They observed 13 people doing usual office work on their computers for one hour each, and logged application interaction and switching. Afterwards they interviewed their participants to get insights into the motivations for application interactions. Using a grounded theory approach, Jin and Dabbish were able to find 7 categories of self-interruptions, among them (for instance) waiting times for a primary task to continue, routine as a habit of time and sequence, and switching to a more desirable task. Further, they categorize self-interruptions to be either stimulus-based or goal-oriented, internally or situationally initiated.

Dabbish et al. [75] investigate the phenomenon of self-interruptions on stationary computers. Based on 889 hours of study data from 36 office workers, they found that self-interruption is strongly influenced by companies' organizational environments (e.g., open office spaces), and individual differences. They also discuss a positive aspect of self-interruptions, as workers self-interrupt to complete work

they are accountable for. Most interestingly, they reveal that external interruptions increase the occurrence of self-interruptions in the following hour, which suggests that interruptions can be habit-forming.

Salvucci [213] argues that task resumption after interruptions is a reconstruction process rather than a memory-based process. A discussion of empirical results show that the time people take is too long for memory retrieval. Thus, he argues that for common HCI tasks, reconstruction of a previous task, e.g. re-reading previously written text, is a central process.

Multitasking While Driving

As an extension to the literature on desktop multitasking, we present research on multitasking in the car, since it introduces phone calls as the cause of interruptions. The effect of phone call interruptions has been extensively studied in this domain (see e.g. [200, 130, 48]) due to the high cognitive demands on the primary task of driving and the severity of interruptions.

Early work by Redelmeier et al. [200] suggests that the risk of an accident when using a cellular phone is four times higher. The authors analyzed phone call behavior of about 700 drivers based on their billing records.

Iqbal et al. [130] conducted a car driving simulation study with 18 participants where drivers had to conduct a phone call while driving (setup shown in Figure 2.6(a)). They found that drivers performed worse on more complex routes (more cars, speed and lane changes), and that the more cognitive demanding the phone call is, the more problems with driving arise. In [129] Iqbal et al. studied how displaying pre-alerts of critical driving segments to driver and caller and pausing a conversation by putting the caller on hold would impact simulated driving (setup shown in Figure 2.6(b)). They found that pre-alerts could reduce driving errors and collisions, in particular when they provide details on the drivers' situations. While drivers also preferred to put the conversation on hold, callers found this mode annoying.

Brumby et al. [48] studied how participants would interleave tasks while driving. They conclude that participants would adapt their strategies for switching between driving and the secondary task based on the objective of the secondary task. In their simulation study, participants had to dial a phone number while driving a car,

(a) Iqbal et al. in 2010 [130] (b) Iqbal et al. in 2011 [129]

Figure 2.6: Setups for studying call interruption while driving.

and participants entered chunks of numbers and made quick glances to the road, rather than making a longer interruption.

Multitasking on Mobile Devices

Although multitasking and task interruptions have extensively been studied on stationary computers, so far little has been reported for mobile users and their unique set of difficulties. Studying multitasking on mobile devices became popular when mobile phones became ubiquitous and provided functionality to support users with everyday tasks. Pioneering research, strongly related to the previously mentioned work on driving, was about usage of mobile phone calls. That being said, this is a rather young field of research, so pioneering work will be presented next.

Oulasvirta et al. [181] investigated the issue that people have to split their cognitive resources in mobile scenarios. They carried out a study (28 subjects) on mobile Web search tasks while moving, and observed that continuous attention to the mobile device is fragmented, mostly due to environmental distraction, and broke down to short time spans of 4–8 seconds.

Karlson et al. [142] investigate task flows on stationary desktop computers, smartphones and on combinations of both. As such, they analyze tasks as a whole, including switching to a stationary computer to complete a mobile task, if necessary. They carried out a 2-week diary study mostly focused on email management (24 subjects), characterizing how problematic interruptions are to mobile users and identifying primary sources of frustration. They derive implications for supporting multi-device task flows, and conclude that a variety of challenges regarding multitasking on smartphones do not exist on the desktop.

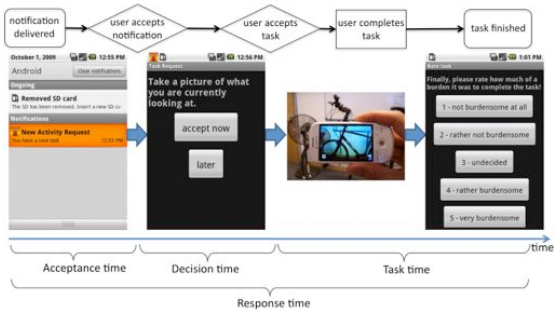


Figure 2.7: The study design of Fischer et al. [96] and the durations they measures.

Ames [6] studied mobile multitasking of student iPhone owners on a university campus. She investigated the techno-social lives of colleague students through three studies, and finds that her study participants — despite being rather tech-savvy and *"digital natives"* — actively set limits to their own smartphone use and disconnect from their devices due to stress and the increased cognitive load of constantly being connected and multitasking. The paper suggests improvements of smartphone designs with regard to social and cognitive aspects.

Fischer et al. [97] investigate how the content and the time of interruptions impacts the receptivity of mobile interruptions, e.g., pushed pieces of information such as notifications. Through an experience sampling based study, they find that content (respectively interests, relevance, and actionability) has a major impact on a person's receptivity to an interruption. The authors draw the conclusion that for design of new systems, one needs to consider factors such as whether the content is interesting, urgent and of high priority for the interrupted person. Regarding the timing of an interruption, the authors conclude that this strongly relates to the interrupted person's current activity and social surroundings. In [96] Fischer et al. study whether there are opportune moments in mobile device usage to deliver notifications, since they lead to interruptive tasks. They conduct an experience sampling study with 20 participants who they interrupt with different tasks (do multiple choice test, compose free text, take a picture) with different timings (random, after phone call, after SMS), as shown in Figure 2.7. During the study Fischer et al. tracked how participants would interact with the notifications and when they would conduct the tasks. Authors find that users act on notifications faster if they have just finished an episode of device usage (calling or reading and SMS).

Figure 2.8: The setup of Brumby and Seyedu [46] for study of auto-locks on smartphones while driving.

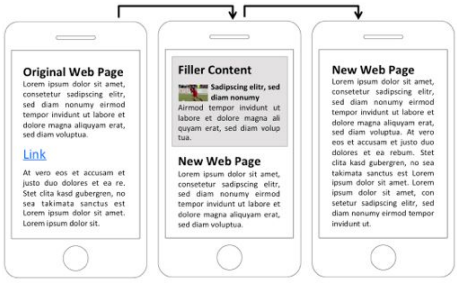


Figure 2.9: The idea of multiplexing a smartphones' display for multitasking [5].

In [46] Brumby and Seyedu were interested in the auto-lock feature of modern mobile phones and found that people feel pressured to return their attention to the phone sooner than usual to prevent the lock. In a simulated car driving study as shown in Figure 2.8, they found that lane-keeping improves with longer auto-lock times, since people took longer pauses and had more time to concentrate on curved road sections. The authors call attention to a design trade-off: While longer lockout times might make devices more useable in mobile scenarios, they also make them less secure and increase energy consumption — which counteracts the reason why auto-locks were introduced.

Alt et al. [5] introduce two different modes for displaying additional web content when waiting for websites to load on a smartphone. A prototype implements a space-multiplexed and a time-multiplexed approach for interweaving content, as shown in 2.9. Although their study did not reveal any differences in usability or other negative impacts, they argue that space-multiplexing allows for more control, as the user can still view the primary website.

Cauchard et al. [56] also investigate the problem of multitasking on a mobile device, and introduce spatially-aware screens for mobile environments. Their study reveals that with spatially-aligned workspaces, users are faster, make better deci-

sions and have a lower physical and mental workload. Further, they introduce a projection-based approach to provide auxiliary displays when switching through applications.

Context-aware Telephony

Another large branch of research related to the work on multitasking between smartphone tasks that we present in Chapter 6 is about phone call interruptions and how they can be mitigated. Much work (see e.g. [221, 241, 106, 78]) investigates how this problem can be solved by conveying the context of the callee to the caller, and vice versa. This topic is also referred to as context-aware telephony [217, 221].

Pioneering work was done by Schmidt et al. [221]. They presented the *Context-Call* system which was the first system for sharing contextual information between communication peers on mobile phones. The system leverages the Wireless Application Protocol (WAP) to exchange information between the peers. Users were able to set their context (either as free text input or chosen from a pre-defined list of free, meeting, working, home, and busy), which was available to callers through a shared database implemented in a client-server architecture.

Knittel et al. [149] also present a mobile application that augments the personal address book with contextual information. Their design is informed by a study suggesting that people's abstract location and movement, recent phone usage, scheduled appointments and manual cues (like setting the phone to vibrate) are most valuable for making decisions on placing a call.

Schneider and Kielser have adapted the idea of displaying remote context to the phone calls during driving [222]: Conveying the driver's context and traffic information to the caller reduced accidents in their study.

Teevan and Hehmeyer [241] investigate the relation of the caller's interruption decisions when placing phone calls to the callee's decisions whether to accept the call. They find that a callee is, interestingly, more likely to accept a call when signaling *"busy"* or *"do not disturb"* to the caller, since the caller then might have decided to call for an urgent reason.

In [106] Grandhi et al. present and study their implementation of *TellingCalls*: a mobile phone application that enables the caller and the callee to convey information about a phone call, such as its topic. A 36-user qualitative and 30-user quantitative study revealed that callees used the information to inform their deci-

sion whether to answer the call, and in particular information provided by callers on call length, urgency and importance helped to convince somebody to pickup a call. Further they found that dynamic information is more useful than information that does not change very often. Provided information was also used to ground the conversation; however, callers disliked the effort of entering information before the call.

Guzma et al. [78] investigate what information cues callers and receivers actually would require to make better decisions rather than what is technically possible. They conducted a diary study with 13 users over 4 weeks. They found that callers could benefit from knowing task status and physical activity of the callee, and the callees considered their social availability to be important. As implications, Guzman et al. propose to make physical activity (e.g. sleeping or driving a car) and the callee's distance from the phone transparent to the caller.

In their study on unavailability in mobile phone communication, Salovaara et al. [212] revealed that 31.1% of phone calls relate to missed calls and follow-up actions. They found that reasons for being unavailable fall into four categories: unavoidable unavailability (e.g. ring tone not audible), enforced unavailability (e.g. rejecting calls due to social context), intentional unavailability (user's own decision not to respond to others), or negligent unavailability (users are unconcerned about being available).

Mobile Call Interruptions

Regarding specific research on call interruptions in a mobile phone context, we found only a few works that studied the costs of responding to a call and discuss auto-reject of calls.

Ho and Intille [123] presented a sensor-based strategy for delaying call interruptions that are not time-sensitive until a physical activity transition; this was successfully tested with 25 subjects. Stamm et al. [235] develop a system that aims to calculate the costs of interruptions in order to schedule interruptions on mobile phones. The system collects context information and implements a decision tree to decide for which incoming calls the phone shall ring and for which not.[22] However, these works do not take into account phone calls with concurrent application usage on the smartphone, as is possible on the latest generation of mobile phones.

[22] Stamm et al. do not report what exactly they are using.

Ter Hofte [243] studied when people are interruptible for mobile phone calls, and which pieces of information about their contexts they are on the one hand willing to share and are on the other hand predictive for their availability. The paper presents an experience sampling study (10 participants) and concludes that a predictor based on a naive Bayesian network and incorporating information that people are willing to share (social relation between caller and callee, whether they are in a conversation or at home) performs better than guessing about the callee's availability by chance. Still, the presented method is too inaccurate for automatically rejecting calls.

It is clear that mobility imposes cognitive restrictions and continuous interruptions on application usage. However, to the best of our knowledge, there is no previous research that focuses exclusively on the application level. While Benbunan-Fich et al. [25] introduce metrics for measuring multitasking on stationary computers (e.g. based on the number of windows used during a task), since on smartphones only one window can be open at a time, focusing on the application-level approach is reasonable for smartphones as we provide first studies on this fine-grained level of multitasking. Moreover, other studies in a similar vein introduced above have been performed in carefully controlled settings. Our main aim with this work is to investigate the costs of mobile application interruptions in the wild, i.e., in a natural, general environment and at scale that makes it possible to find subtle effects.

2.4 Summary

In this chapter we have introduced the foundations of this thesis, presented the research approach we have taken for the work we conducted, and discussed works relating to the four fields of this thesis as well as mobile phone use in general.

First, we introduced the notion of the term *context* and highlighted its meaning for the field of this work, i.e. mobile Human-Computer Interaction. The concept of context can be used to make the ever-changing circumstances of mobile phone usage accessible.

In Section 2.2.1 we explained the research approach that we are following to carry out the work we will present in this dissertation. We introduced the impetus for our approach based on deployment-based research, and presented the method of research through the application store and how it is motivated through research in

the wild and grounded in theory of quasi-experimental design. Based on that, we presented frameworks and datasets which follow a similar idea as our work, and we presented the results of studies using the same approach. We will refer to the methods we introduced when reporting on the studies that we conducted.

Next, we presented works relating to the four fields of launching, housekeeping, discovering and multitasking between mobile applications as well as mobile phone usage in general in Section 2.3. The link between related work and these four fields is as follows:

- *Launching:* From related work on web and application usage we have most importantly learned that user context influences mobile device usage and information needs. Based on that, we extend existing literature, in particular by investigating mobile application usage through combining a fine-grained tracing approach with running a study through the mobile application store in Chapter 3 (contribution of paper [38]). We have also taken into account related work on adaptive menus for designing an adaptive mobile application launcher (part of the contribution in [183]), such as the idea of two-split menus.
- *Housekeeping:* In this chapter we also presented related studies of information management on stationary computers looking into how people arrange icons on their desktops, as well as people's mental models of their smartphone menus. We contribute to this line of work (Chapter 4, published in [40]) by reaching out to the mobile domain and studying how people arrange their icons in their smartphone launchers, which concepts and other factors influence their arrangements, and how this can be exploited for improving systems and deriving implications for smartphone design.
- *Discovery:* We also presented related work on context-aware recommender systems, which we took into account when designing the mobile application recommender system in Chapter 5. In particular, we extend existing literature by presenting an approach for evaluating different recommender algorithmns based on the engagement of users with a recommended application (this led to paper [36]). In addition we will provide a design space and a system design for recommender systems that suggest mobile applications.
- *Multitasking:* Our work on multitasking is based on studies of application switching on stationary computers, phone call interruptions while driving a car, multitasking on mobile phones and call-related studies. We extend existing literature by focusing on interruptions that do not happen in the

environment of a mobile phone user, but on the device itself through another application. Our study replicates the phenomenon in the mobile domain and discusses findings on mobile application switching and call interruptions (contributed in [157]), and leading to the proposal of a new UI design for phone call applications.

In the following four chapters, we present our work based on the foundation we presented in this chapter. Our work uses the research approach we introduced and delivers contributions going beyond what is known in current literature as we have sketched.

Chapter 3

Launching Mobile Applications

This chapter deals with the launch of mobile applications on smartphones. We will first describe our way of tracing mobile application usage, then look into how people use their applications, and finally propose a solution to help people launch their mobile applications quickly.

The results of this chapter have been presented in three publications [38, 41, 183]. Work related to this chapter can be found in Sections 2.2.3 on logging frameworks and 2.3.2 on mobile application usage.

3.1 Introduction

While mobile applications for smartphones have become extremely popular in the last few years — as we described in Chapter 1 — little public information exists on how people make effective use of the mobile applications that are available and that they have installed on their devices. In this chapter, we first develop a means to trace mobile application usage, then go on to consider how people utilize mobile applications on their smartphones through collecting data on application launches, and finally provide support for launching mobile applications easily.

Launching is the act of starting something or putting something into motion.[1] In common parlance the term is often used to describe, e.g., putting a boat put into the water, or sending a space shuttle into orbit. In this work, we refer to launching as the act of starting mobile applications, which are represented by icons and which are activated when a user selects them from a menu and clicks on them — as occurs in classical WIMP interfaces [208, p. 160].

This chapter is structured as follows. First, in Section 3.2, we describe and discuss the notion of a virtual sensor that we designed to trace mobile application usage directly on end-users' smartphones. In particular, this sensor focuses on the end-user's perspective on utilizing an application rather than on a operating system's perspective on when an application is actually being executed as a software process. Second, in Section 3.3, we describe a large-scale deployment-based study that logged detailed application usage information from over 4,100 users of Android-powered mobile devices using the sensor we introduced. As results of this study we present two types of findings obtained by analyzing the data that we have collected: On the one hand, we provide basic descriptive statistics, and on the other hand we present contextual descriptive statistics. In the case of the former, we find for instance that the average session a user spends within an application lasts less than a minute per launch, even though users spend almost one hour per day using their smartphones. Our contextual findings include those related to people's time of day and location. For instance, we show that applications providing news and weather information are most popular in the morning and games at night, but communication applications dominate through most of the day. We also find that despite the variety of mobile applications that are available, communication applications are almost always the first applications used when a user starts using the device again after it has been in standby mode. Finally, in Section 3.4, we present an adaptive menu that supports smartphone users when launching applications on their smartphones. This system uses the framework for tracing mobile application usage that we will introduce and is informed by the results we gained from the study of mobile application usage. The system's adaptive UI provides a changing set of icons so that the user can start the next application more easily.

Contributions of this chapter are a framework for tracing mobile application usage on end-users' devices along with contextual information, findings on patterns and habits of mobile application usage resulting from a large-scale deployment and a

[1] Oxford Dictionaries: *"launch"*, Oxford University Press, http://oxforddictionaries.com/definition/english/launch, last accessed on 02.07.2013

study of the collected data, and a implementation of an adaptive menu to support people utilizing their installed applications. Related work on understanding and supporting the utilization of mobile applications can be found in Section 2.3.2.

3.2 Framework for Tracing Mobile Application Usage

Despite the large numbers of mobile applications and users of mobile application as described in Chapter 1, public research has only recently begun to understand how people use their applications (cf. Chapter 2, esp. [251, 100, 79, 103]). Very basic and fundamental questions on how people utilize the mobile applications that they have installed has remained unanswered for a long time; for instance, how long does each interaction with an application last, and does this vary between different application categories? If so, which categories and types of applications inspire the longest interactions with their users? So far, data on the context's effect on application usage is equally sparse, leading to additional interesting questions. How does the user's context — e.g., location and time of day — affect her choices of which application to use? What type of application is opened first when users take devices out of their pockets, and does opening one application predict opening another?

In this section, we provide a basis for providing answers to these questions by introducing a framework for tracing mobile application usage. We describe a virtual sensor for measuring mobile application usage right on the users' smartphones. Since this framework basically provides a means to measure mobile application usage, we call this virtual sensor *AppSensor*.

3.2.1 A Sensor for Measuring Mobile Application Usage

In this section, we describe our data collection tool: *AppSensor*. Because context is a known important predictor of the utility of an application [17], *AppSensor* has been designed from the ground up to provide context attached to each sample of application usage that it takes.

Lifecycle of a Mobile Application

The goal of the *AppSensor* is to trace application usage as the user experiences it. Therefore, we need to gain an understanding of how users can interact with

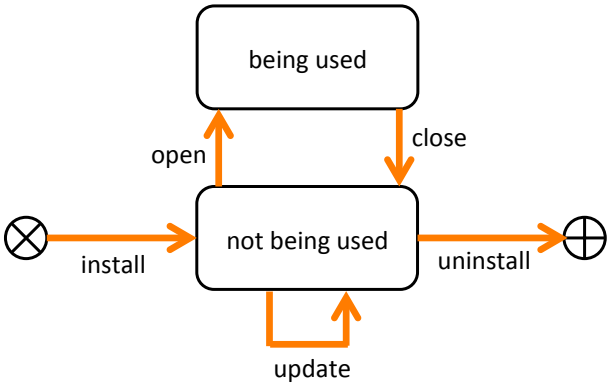


Figure 3.1: The lifecycle of a mobile application on a user's device according to different states and events.

their applications in general — abstracting from all the functional aspects that each application has, and reducing the usage of general common aspects. The design of the *AppSensor* is based on the lifecycle of a mobile application, as shown in Figure 3.1. The *AppSensor* understands five different events in an application's lifecycle:

- installing
- updating,
- uninstalling,
- opening,
- and closing the application.

The first event that we can observe is an application's installation. It reveals that the user has downloaded an application, e.g., from an application market. Another event that is observable is an update to an application, which might be interpreted as a sign of enduring interest in the application. However, since updates are sometimes done automatically by the system and the update frequency strongly depends on the release strategy of the developer, the insights into usage behavior that can be gained from update events is relatively low, simply because the event does not happen as a result of the user's intent or action. The last event that we can capture of an application on a user's smartphone is the uninstall event, which expresses the opposite of the installation event: a user does not want the application anymore.

However, these maintenance events (i.e., installation, update, uninstallation) usually only occur a few times per application. For some applications, there might even be only a single installation event (e.g. when the user has found a good application) or even no event at all (e.g. for preinstalled applications like the phone

application). For other applications these events might repeat and result in a loop over these events, e.g., when a user re-installs an application that he has deleted once. Further, such maintenance events are also of limited utility for understanding the relationship between a user's context and his application usage. Because, such events might be decoupled from the user's context, for instance, a user might install an application at one location then use it elsewhere (e.g., an application for sightseeing might be installed at home in preparation for traveling).

Some systems, e.g., the *AppAware* system presented by Girardello et al. [103, 104], rely solely on these events. Instead, *AppSensor* is designed to continuously sample a user's application *usage*. In other words, we are especially interested in the two application states of *being used* and *not being used*, which can both be inferred from the open and close events, as Figure 3.1 suggests. For most applications, these events naturally appear much more often and in a much shorter period of time than the maintenance events (install, update, uninstall). Tracing the two states of *being used* and *not being used* enables us to observe application usage on a more fine-grained level and provide a much more accurate understanding of context's effects on application usage. In this work, we are particularly interested in the open events, which we also refer to as the user's act of launching of the application. The act of launching an application on a smartphone implies that the user navigates to the icon of the application within her launcher menu and then clicks on the icon to open the application.

In order to gather data on the two states *being used* and *not being used*, our implementation of the *AppSensor* takes advantage of the fact that the Android operating system can report the most recently started application through its API[2]. Because of this feature, we know with which application the user is currently interacting. We are thus able to infer which single application is in the state of *being used* owing to the fact that the Android operating system only shows only one application to the user at once (as does the iPhone OS and most other mobile operating systems, due to limited screen real estate). Therefore, we can presume that all other applications are consequently in the state of *not being used* in terms of not showing their graphical user interface to the user. It is worth mentioning that this is contrary to using applications on stationary computers which have larger screens and commonly provide better capabilities for multitasking when multiple application

[2]In particular, this can be done through an instance of the class `ActivityManager`.

windows are open (a deeper discussion on multitasking between mobile applications will appear in Chapter 6).

Our implementation of the *AppSensor* does not consider background applications that are not interacted with through the smartphone's graphical user interface, e.g., background applications like music players that can be controlled through gestures or other modalities.

A Formal Description of the AppSensor

As noted above, the *AppSensor* is meant to be a sensor that indicates the specific application that is currently being used at a given time t. Speaking more formally, the sensor can be described as follows: Let $A = \{a_1, \ldots, a_n\}$ be the set of applications that are available for a smartphone and let $A^* = A \cup \{\epsilon\}$ be the set of applications with which a user can interact. The symbol ϵ means that the user is currently not using any application at all, e.g., when the smartphone is turned off or in stand-by mode. For most current platforms (e.g., Google's Android and iOS), this set A is usually defined by the applications available on the corresponding application stores. As we discussed in Chapter 1, the number of applications is growing, and therefore this set is not static. However, this set always has a defined number n of elements, i.e., the number of available mobile applications. With time given as t, the *AppSensor* shall provide the following values:

$$as(t) = \begin{cases} a_i & \text{if application } a_i \text{ is being used,} \\ \epsilon & \text{if no application is being used.} \end{cases}$$

With respect to the lifecycle of mobile applications the value $as(t)$ describes the application that a user has launched most recently and is currently using. The value is distributed on the nominal scale given by the set A^* of available applications. Therefore, the simple conclusion that can be drawn on the mere sensor data of two measures at times t_1 and t_2 is a comparison of whether the application a user is using is the same as before (if $as(t_1) = as(t_2)$) or whether it has changed (if $as(t_1) \neq as(t_2)$).

3.2.2 Implementation and Deployment

We implemented the virtual sensor that we have designed previously for the Android operating system, where it was possible to implement it as a background service[3]. This background service is always running on a user's smartphone and traces context information that is available directly on the user's device (e.g. location, local time, previous application interaction) and application usage at the same time. As such, the sensor is able to collect very rich data as it collects not only information on application usage, but also data on various other pieces of context information.

We implemented the main *AppSensor* loop for tracing which application is currently being used with a sampling rate of 2 Hz, which means that the background service checks for the two states of applications of being used or being used every $500ms$ and collects this data accordingly. This loop starts automatically as soon as the device's screen is turned on and stops when the screen is turned off again. When the device goes into standby mode, we note which application was left open and omit the standby time from the application's usage time. We were able to determine when the device was in standby by listening to screen-off and screen-on events that are provided by the operating system. One special case is the phone application, since the implementation of most mobile phone applications turns off the screen as soon as the user holds the phone against her ear, using the proximity sensory to prevent unwanted button presses.

Running in the loop, the *AppSensor* keeps track of which application is used every time the value of the sensor changes, i.e., $as(t_{i-1}) \neq as(t_i)$, which means that the user has switched to another application. Besides the value of the sensor, i.e., the identifier of the current application, all sensor data and a local timestamp given in milliseconds is recorded.

The data measured by the *AppSensor* is recorded in a local database on the user's device and only periodically uploaded to a central server. In the first place, only uploading it to a central server will allow us to study mobile application usage by analyzing the data. In case of connectivity failure, the data is kept in the smartphone's database and will be included in the next transmission.

[3]This is made possible on the Android platform through background threads. Those can be started and continuously executed in the background, which is not commonly possible on other mobile operating systems.

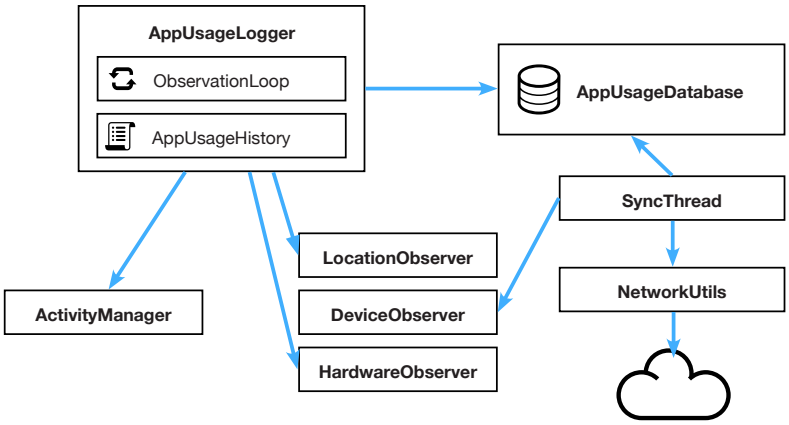
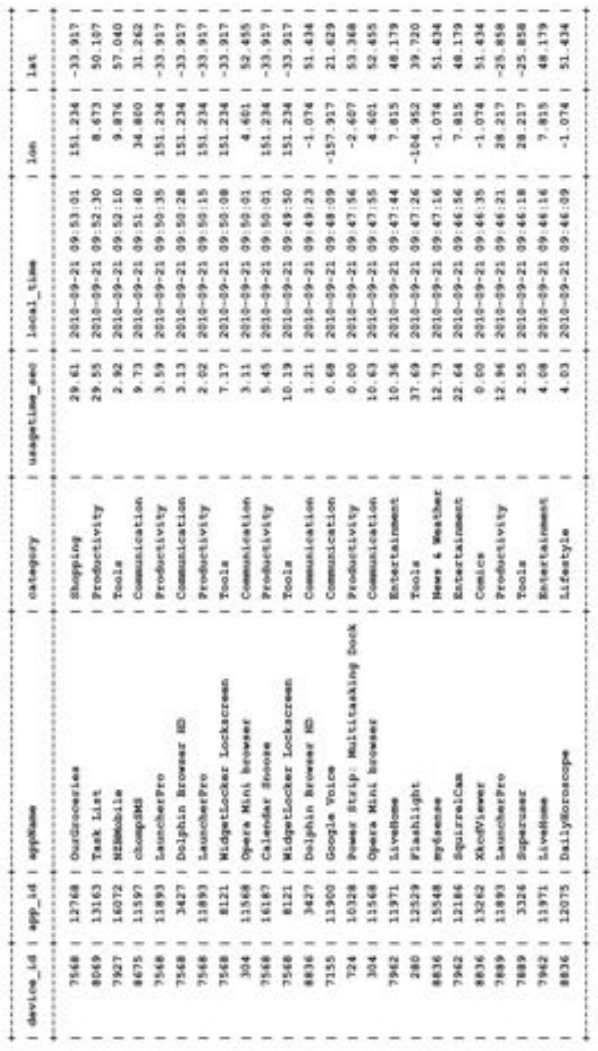


Figure 3.2: Conceptual architecture of the *AppSensor* framework.

Figure 3.2 shows the architecture of our implementation of the *AppSensor* framework. The `AppUsageLogger` constitutes the core of the *AppSensor* framework. This class implements the main thread that executes the main observation loop that traces which application is currently being used by the smartphone user. To observe the state of applications, it queries the currently running tasks by using the `ActivityManager`, which is part of the API provided by the Android OS. For reasons of performance and power consumption, the `AppUsageLogger` keeps the traces of application usage in memory, and only persists them into a database running on the smartphone whenever the device goes into standby mode. Additional contextual information is provided by components for observing location-related sensors (e.g. speed, geo-location) and hardware-related information (e.g., state of battery, wireless connections). To upload the data from the mobile database to a server accessible via the Internet, a dedicated component for synchronizing this data is available (cf. `SyncThread`). When the data is uploaded to the Internet, i.e., when it leaves the smartphone, additional pieces of information related to the device are attached (e.g. device identifier, device model name, resolution). Since the values of these variables are inherent to the device, i.e., they will not change, this information is added only when data is uploaded.

Figure 3.3 shows an example database excerpt of the information that is saved into the central database. At the bottom line — and most importantly — the data tells us who (column: `device_id`) has used which application (columns: `app_id`, `category`) for how long (column: `usagetime_sec`) at what time (column: `local_time`) where (columns: `lon`, `lat`); additional context information



device_id	app_id	appName	category	usagetime_sec	local_time	lon	lat
7568	12768	OurGroceries	Shopping	29.61	2010-09-21 09:53:01	151.234	-33.917
8049	13163	Task List	Productivity	29.55	2010-09-21 09:52:30	8.673	50.107
7927	16072	NIMMobile	Tools	2.92	2010-09-21 09:52:10	9.876	57.040
8675	11597	chompSMS	Communication	9.73	2010-09-21 09:51:40	34.860	31.262
7568	11893	LauncherPro	Productivity	3.59	2010-09-21 09:50:35	151.234	-33.917
7568	3427	Dolphin Browser HD	Communication	3.13	2010-09-21 09:50:28	151.234	-33.917
7568	11893	LauncherPro	Productivity	2.02	2010-09-21 09:50:15	151.234	-33.917
7568	8121	WidgetLocker Lockscreen	Tools	7.17	2010-09-21 09:50:08	151.234	-33.917
304	11568	Opera Mini browser	Communication	3.11	2010-09-21 09:50:01	4.401	52.455
7568	16187	Calendar Snooze	Productivity	5.45	2010-09-21 09:50:01	151.234	-33.917
7568	8121	WidgetLocker Lockscreen	Tools	10.19	2010-09-21 09:49:50	151.234	-33.917
8836	3427	Dolphin Browser HD	Communication	1.21	2010-09-21 09:49:23	-1.074	51.434
7155	11900	Google Voice	Communication	0.68	2010-09-21 09:48:09	-157.917	21.629
724	10328	Power Strip: Multitasking Dock	Productivity	0.00	2010-09-21 09:47:56	-2.607	53.368
304	11568	Opera Mini browser	Communication	10.63	2010-09-21 09:47:55	4.401	52.455
7942	11971	Livehome	Entertainment	10.36	2010-09-21 09:47:44	7.615	48.179
280	12529	Flashlight	Tools	37.69	2010-09-21 09:47:26	-104.952	39.720
8836	15548	mydsense	News & Weather	12.73	2010-09-21 09:47:16	-1.074	51.434
7942	12186	SquirrelCam	Entertainment	22.64	2010-09-21 09:46:56	7.615	48.179
8836	13262	XkcdViewer	Comics	0.00	2010-09-21 09:46:35	-1.074	51.434
7889	11893	LauncherPro	Productivity	12.96	2010-09-21 09:46:21	28.217	-25.858
7889	3324	Superuser	Tools	2.55	2010-09-21 09:46:18	28.217	-25.858
7942	11971	Livehome	Entertainment	4.68	2010-09-21 09:46:16	7.615	48.179
8836	12075	DailyHoroscope	Lifestyle	4.03	2010-09-21 09:46:09	-1.074	51.434

Figure 3.3: Excerpt of database showing data collected by *AppSensor*.

like battery status or the device's connectivity status to wireless networks was not shown in the table for space reasons.

For security and privacy reasons our implementation uses one-way hash functions to anonymize all personal identifiers before the data is collected and sent to the server (columns `app_id` and `device_id` of Figure 3.3 show technical identifiers). Furthermore, we do not gather any additional personal information like the name, age or gender of the user. Such demographic information has been shown to be an unreliable predictor of smartphone usage anyway [91]. We removed data coming from our own test and development devices from the collected data contributed

by end-users by using special identifiers for test devices, which we could easily remove from the data we analyzed.

3.3 Large-scale Study of Mobile Application Usage

In this section, we describe results of a large-scale study on mobile application usage, where we made use of the framework for tracing mobile application usage that we presented previously.

3.3.1 Method and Setup of Study

We followed the method of research through the application store that we described in Chapter 2. Our study can be understood as a passive observational study as we do not apply any treatment to our participants and rather look for relations between the observed variables.

We decided to study data collected by means of logging since self-reports on application usage are very likely to be error-prone [171], and only this approach allowed us to collect data on a large scale — both in terms of participants and applications — for being able to find subtle effects. Therefore, we used the Android application store as a means to leverage our study and distribute our deployment of the *AppSensor* as described in Chapter 2.

For this study, we were in particular motivated by the question of how people make active use of the mass of mobile applications that are available on their smartphones and how they use mobile applications in natural contexts. This study is meant to answer the basic questions on application usage that we introduced earlier, as well as those related to contextual usage of mobile applications.

The results reported in this section are based on data from 4,125 users, who used an application containing an implementation of the aforementioned framework *AppSensor*. The application itself containing the sensor for tracing mobile application usage was a mobile application recommender system called *Appazaar*, which will be described in detail in Chapter 5. The system has been available on the Google Play Store and end-users were able to download the application from this application store. It was used by people for varying amounts of time. Users installed the *AppSensor* as part of the recommender system which also used the

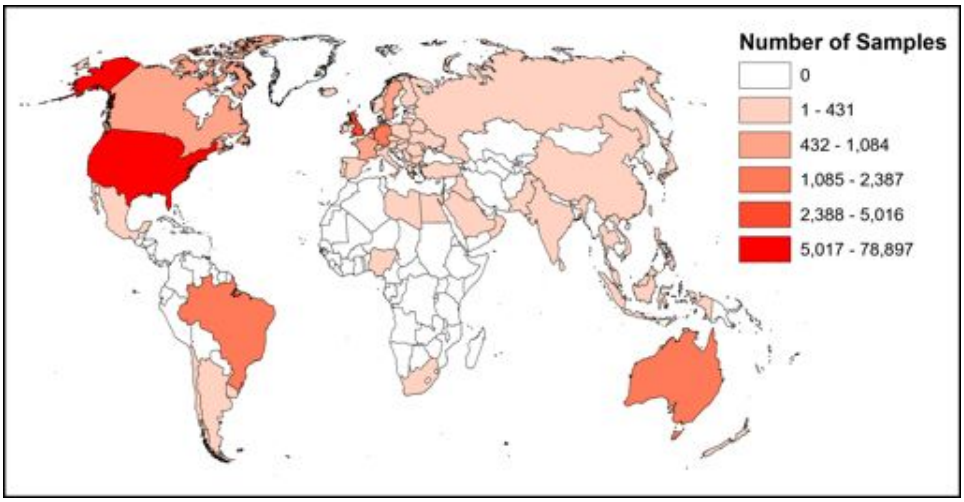


Figure 3.4: The geographic distribution of our users

collected data to inform its algorithms for the recommendation of other mobile applications.

The data that we describe in this section was collected between August 16th, 2010 and January 25th, 2011. The participants of our study were spread out geographically, although most stayed in the United States or Europe during our study (see Figure 3.4). Within the timeframe of 163 days, they generated usage events for 22,626 different applications and the deployment of our *AppSensor* was able to measure 4.92 million distinct instances of mobile application launches. To get people to install and use the application that we used to collect data for this study, we advertised the application on Facebook and Twitter, and two news posts about the system were published on two well-known technology websites (*Gizmodo*[4] and *ReadWriteWeb*[5]), helping us reach a growing number of users.

To achieve a higher-level understanding of our data, we felt it was necessary to add the categories to the applications that we had very fine-grained data on. To do so, we mined the Google Play Market for each application's category. Table 3.1 shows the resulting categorization with examples of famous applications in each category. As such, the categories are largely defined by the applications' developers: they — as domain experts — assign their applications to the categories when uploading

[4]Gizmodo: *Appazaar Recommends Android Apps Through Usage, Location*, http://goo.gl/ExWbQ, last accessed on 04.07.2013.

[5]ReadWriteWeb: *This App Recommender Would Like to Use Your Location*, http://goo.gl/Lk7XP, last accessed on 04.07.2013.

them to the Google Play Store. The only exception to this rule occurred in some minor manual modifications. For instance, we merged all games in the categories *Arcade & Action*, *Brain & Puzzle*, *Cards & Casino*, and *Comics* into one *Games* category. Due to the special nature of browsers — i.e., they do not have a clear-cut domain scope like dedicated applications — we have separated them into their own dedicated *Browsers* category. For some applications, no categories were available on the Google Play Store. These applications are either test applications by developers that appear only on a few devices, applications that are distributed via other channels (e.g., pre-installed by device manufacturers or distributed by companies for their employees), default Android applications (e.g. settings), or applications that have been taken off the market and whose categories were not available at the time when we queried the application categories. We crawled the Android Market on February 3^{rd}, 2011. We manually added categories for some applications where possible. For instance, for the branded Twitter clients of some vendors (e.g. HTC), we added the category of the original Twitter application (i.e. *Social*). To the default applications responsible for handling phone calls we added the *Communication* category. As we did with the browser, we also put the settings application into its own category (*Settings*) due to the special nature of these two applications. Since on devices running the Android operating system the main launcher menu itself also is an application and it is treated as such from the system's perspective, we additionally removed such launcher applications from the results since they give few insights into application usage behavior. Finally, it is important to note that each application can have only one category.

3.3.2 Results of Study

We analyzed the data that we were able to collect to understand how people use mobile applications. Our results are twofold, and therefore this section is divided into two parts: First, we provide basic descriptive statistics on application usage behavior, and second we provide context-sensitive statistics. In the second part, we look at several different forms of context, including an application's place in an "application use chain", as well as more standard contextual variables such as time and location. In both sections, our primary resolution of analysis is the "application category" as defined above, but in the second part we do highlight some interesting temporal patterns for specific single applications.

Basic Descriptive Statistics

On average, the participants of our study spent 59.23 minutes per day on their devices. However, the average application session — from opening an application to closing it — lasted only 71.56 seconds.

Table 3.1 shows the average usage time of applications by category, which ranged from 36.47 seconds for applications of the *unknown* category and 31.91 seconds for applications of category *Finance* to 274.23 seconds for category *Libraries & Demos*. The most-used *Libraries & Demos* applications as measured by total usage time are inherent applications of the operating system (*Google Services Framework*, *default Updater*, *Motorola Updater*). This suggests that maintaining an Android smartphone can be quite a time-consuming task.

It was interesting to see that the *Libraries & Demos* category has a much longer average session than the games category, whose most used applications are *Angry Birds*, *Wordfeud FREE*, and *Solitaire*. Despite its name, *Wordfeud FREE* is a full game and not a demo version, since it provides the same full functionality like the non-free version of the application. The only difference is that it is free and contains advertisements for the developer's revenue. On the low end of the session length spectrum of applications with known categories, we found the *Finance* category. The most-used applications of this category are for personal money management (*Mint.com Personal Finance*), stock market (*Google Finance app*), and mobile banking (*Bank of America*). The briefness of the average session in this category does not speak well for the success rate of financial applications on the Android platform. Another explanation might be that people do not keep their financial applications open for a very long time since they do not want others to see their personal data.

Application Usage over Time

The *AppSensor* further allows us to record temporal information about application usage. Figure 3.5 shows the total number of application launches in our sample according to hour of the day. It can be seen that total application usage (in terms of launches) is at its maximum in the afternoon and evening, peaking around 6pm. It is not surprising that our participants generally start using applications in the morning between 6am and 7am, and their activity grows approximately linearly until 1pm. Activity then increases slowly to a peak around at 6pm. The minimum

Category	Apps	Avg. usage	Examplary Apps
unknown	4,823	36.37 sec	-
Finance	307	37.01 sec	Mint.com Personal Finance, Bank of America, Google Finance, iStockManager
Travel	782	44.72 sec	Google Maps, Yelp, Waze
Communication	881	46.92 sec	Google Mail, Handcent SMS, K-9 Mail
Productivity	1,062	61.49 sec	Calendar, Evernote, GTasks
Shopping	326	61.71 sec	Market, Barcode Scanner, Craigslist
Social	538	62.69 sec	Facebook for Android, Twitter, TweetDeck
Sports	385	65.98 sec	Yahoo! Fantasy Football '10, ESPN ScoreCenter, NFL Mobile
News	784	68.11 sec	NewsRob, reddit is fun, BBC News
Settings	1	68.71 sec	Default Settings App
Browser	10	74.01 sec	Default Browser, Skyfire Browser, Dolphin Browser
Entertainment	84	76.90 sec	IMDb Movies & TV, TV Guide Mobile, PhotoFunia
Multimedia	130	82.79 sec	Pandora Radio, Music, Camera
Comics	3,242	91.33 sec	DailyStrip, XkcdViewer, Dilbert Mobile
Games	2,822	114.25 sec	Angry Birds, Wordfeud FREE, Solitaire
Health	424	153.80 sec	CardioTrainer, Sleep Bot Tracker Log, Baby ESP
Lifestyle	956	167.77 sec	DailyHoroscope, Gentle Alarm, Epicurious Recipe
Reference	764	176.28 sec	Kindle for Android, Aldiko Book Reader, Audible
Tools	3,004	206.26 sec	AppBrain App Market, Apps Organizer, Google Goggles
Themes	1,061	258.28 sec	Zune Home, Fingerprint Screensaver, HomeChange
Libraries & Demos	240	274.23 sec	Google Services Framework, default Updater, Motorola Updater, Bubbles Demo, Ride Logger Demo, ES Task Manager

Table 3.1: Categories of applications investigated in our study. The table shows the number of applications per category, the average usage time of every categories' applications per launch, and examples of popular applications in each category.

application usage occurs is around 5am, although it never falls below 16% of the maximum.

Figure 3.6 shows the average time people spent with an application per launch, after opening it, versus time of the day. There is a peak around 5am with 6.26 minutes of average application usage time. The average application session is less than a minute, however, reaching a minimum of around 40 seconds at 5pm. Interestingly, the graph in Figure 3.6 is nearly opposite that in Figure 3.5. This means that when people actively start to use their devices, they spend less time with each application. This might be due to applications that people explicitly leave active while sleeping

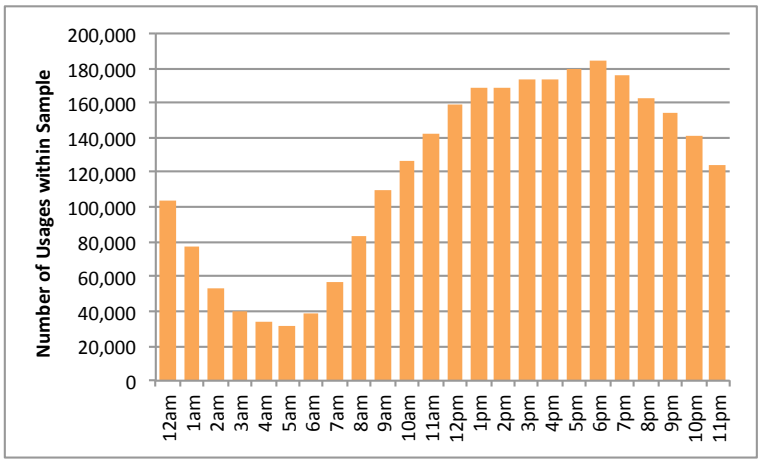


Figure 3.5: Total number of recorded application utilizations during a day.

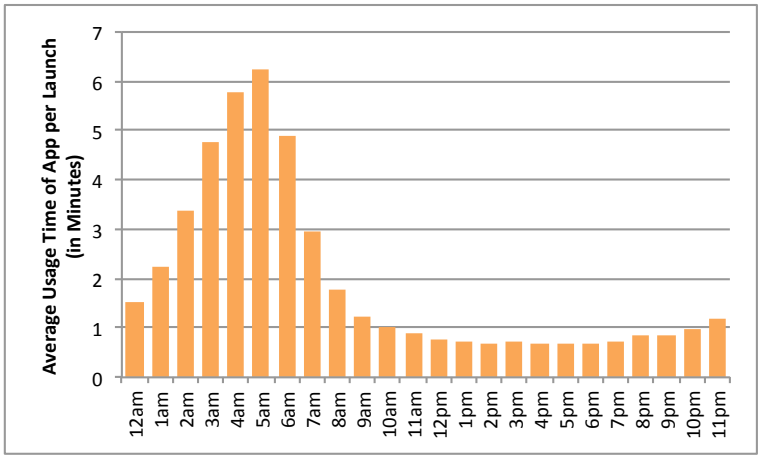


Figure 3.6: Daily average usage duration of opened applications per launch in minutes.

with standby mode prevented, like for instance nightstand clock applications, but there are other possible explanations.

Figure 3.7 shows the change in the relative usage of the application categories over the course of the day in terms of number of application launches. Smartphones are most likely to be used for communication at every hour of the day, especially in the afternoon and evening (11am till 10pm) with a probability of more than 50% that a launch will open a communication application. News applications have the highest probability of being used in the morning (from 7am to 9am). Around 11am, finance applications briefly become quite prominent. After communication winds down late in the evening, games have their highest probability of use. Social

applications also have their highest probability of use in the late evening (from 9pm to 1am). Sports applications seem to play their most important role in the afternoon (between 2pm and 5pm) and in the evening (between 8pm and 10pm). During the early morning, when total application usage is at its lowest, people spend their time with applications of various categories. This is also the time when communication application use share is minimal.

Chains of Application Launches

An important contextual variable in usage behavior is the set of zero or more applications used before an application is launched and the zero or more applications used afterwards. We defined an *"application chain"* as a sequence of applications that are used without the smartphone being in standby mode for longer than 30 seconds. In total, we can distinguish 1,841,226 of such sessions in our data set. Examples include one in which a user started with *Grocery iQ (Shopping)*, switched to *GrubHub Food Delivery (Lifestyle)*, and ended with *Epicurious Recipe App (Lifestyle)*. Another user started with the *AroundMe (Lifestyle)* application and then continued with *Find A Starbucks (Shopping)*, *Google Maps (Travel)*, *Find A Starbucks*, *Google Maps*, *Find A Starbucks*, *Dolphin Browser HD (Browser)*, *Find A Starbucks*, *Google Maps*, *Find A Starbucks*, and *Google Maps*. While the former user seems to be in a shopping context, checking recipes and his shopping list, the latter user seems to be looking for a place to get a coffee using different applications to reach his goal. Another example for such application chains using a different study method was found by Brown et al. [45]: Users switching between a search engine and a map application to lookup locations.

Figure 3.8 demonstrates the distribution of application chains by the number of applications that occur in each particular chain. As the y-axis of Figure 3.8 is on a log-scale, it can be seen that the majority of sessions (68.2%) only contain one single application. In other words, people turn on their phone, use a single application, and put their phone back into standby. This tendency towards the use of a small number of applications during an interaction with the mobile device is further evidenced by the fact that only 19.5% of application chains contain two applications, and only 6.6% contain three. This phenomenon of people frequently using their smartphone to launch only a few applications can be described by the *checking habit* described by Oulasvirta et al. [179], which is a brief but repeated

	12am	1am	2am	3am	4am	5am	6am	7am	8am	9am	10am	11am	12pm	1pm	2pm	3pm	4pm	5pm	6pm	7pm	8pm	9pm	10pm	11pm	% of Total Launches	Users	Apps
Browser	7.9%	7.7%	7.8%	7.6%	7.3%	7.4%	7.0%	7.9%	8.1%	8.0%	7.7%	7.3%	7.0%	6.9%	6.8%	6.4%	6.6%	6.6%	6.4%	6.6%	7.0%	7.4%	7.5%	7.4%	6.83%	2,398	9
Comics	4.5%	5.2%	5.4%	5.8%	5.8%	5.6%	5.5%	5.2%	5.4%	5.1%	4.7%	4.3%	4.3%	4.2%	4.2%	4.3%	4.4%	4.0%	4.4%	4.2%	4.1%	4.1%	4.1%	4.4%	4.31%	2,151	1,810
Communication	44.9%	41.1%	38.3%	35.4%	31.6%	31.8%	32.7%	34.7%	39.4%	44.8%	49.0%	52.6%	54.8%	55.2%	55.2%	56.1%	55.7%	56.8%	57.1%	56.1%	54.8%	53.3%	52.0%	49.0%	49.50%	2,769	550
Entertainment	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.02%	126	43
Finance	0.2%	0.3%	0.3%	0.2%	0.1%	0.1%	0.1%	0.2%	0.3%	0.3%	0.4%	0.5%	0.3%	0.3%	0.4%	0.3%	0.3%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%	0.25%	604	164
Games	3.2%	3.0%	3.0%	2.7%	2.5%	2.3%	2.2%	1.7%	1.9%	1.9%	2.0%	2.1%	2.2%	2.2%	2.2%	2.3%	2.3%	2.2%	2.2%	2.4%	2.7%	3.0%	3.0%	3.2%	2.30%	1,716	1,702
Health	0.3%	0.4%	0.4%	0.4%	0.6%	0.6%	0.7%	0.6%	0.4%	0.3%	0.3%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%	0.3%	0.2%	0.3%	0.2%	0.3%	0.26%	540	227
Libraries & Demo	0.4%	0.5%	0.6%	0.7%	0.9%	0.8%	0.7%	0.6%	0.5%	0.4%	0.3%	0.3%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%	0.3%	0.3%	0.3%	0.3%	0.30%	1,267	117
Lifestyle	0.8%	0.9%	1.0%	1.4%	1.3%	1.5%	1.4%	1.4%	1.1%	0.9%	0.6%	0.6%	0.5%	0.5%	0.5%	0.5%	0.6%	0.5%	0.3%	0.4%	0.4%	0.5%	0.5%	0.5%	0.60%	2,132	451
Multimedia	2.1%	2.1%	2.4%	2.4%	2.7%	2.4%	1.8%	1.8%	1.9%	1.7%	1.8%	2.0%	2.0%	2.0%	2.2%	2.1%	2.2%	2.4%	2.3%	2.3%	2.2%	2.1%	1.9%	2.0%	2.03%	1,713	76
News	2.6%	2.5%	2.6%	2.5%	2.5%	2.7%	3.3%	3.7%	4.1%	3.6%	3.0%	2.6%	2.5%	2.7%	2.5%	2.4%	2.2%	2.1%	2.3%	2.2%	2.3%	2.2%	2.3%	2.3%	2.46%	1,777	440
Productivity	3.6%	5.0%	5.0%	5.8%	6.3%	6.5%	6.0%	5.4%	4.8%	5.1%	4.9%	4.3%	4.2%	4.0%	4.0%	3.7%	3.4%	3.4%	3.0%	3.1%	3.1%	3.0%	2.9%	3.2%	3.76%	2,190	648
Reference	0.7%	0.7%	0.7%	0.7%	0.7%	0.7%	0.6%	0.6%	0.7%	0.5%	0.5%	0.5%	0.4%	0.4%	0.4%	0.4%	0.3%	0.4%	0.4%	0.4%	0.5%	0.5%	0.5%	0.6%	0.47%	903	346
Settings	1.3%	1.6%	1.5%	1.3%	1.6%	1.2%	1.2%	1.1%	1.3%	1.4%	1.4%	1.4%	1.2%	1.3%	1.2%	1.2%	1.3%	1.1%	1.1%	1.2%	1.2%	1.3%	1.3%	1.4%	1.23%	2,178	1
Shopping	3.9%	4.5%	3.7%	3.4%	3.2%	3.2%	3.1%	3.0%	3.1%	3.3%	3.2%	3.2%	3.2%	2.8%	2.9%	2.9%	2.7%	2.7%	2.7%	2.7%	2.8%	3.1%	3.6%	3.5%	2.96%	2,556	198
Social	5.7%	5.0%	4.9%	4.3%	4.2%	4.0%	4.4%	5.1%	5.3%	5.4%	5.2%	5.0%	4.7%	4.8%	4.9%	4.5%	4.5%	4.6%	4.6%	4.9%	5.2%	5.4%	5.8%	5.7%	4.77%	1,902	342
Sports	0.5%	0.3%	0.3%	0.2%	0.3%	0.3%	0.2%	0.3%	0.3%	0.3%	0.3%	0.4%	0.4%	0.6%	0.7%	0.8%	0.9%	0.8%	0.6%	0.6%	0.7%	0.8%	0.7%	0.7%	0.56%	571	215
Themes	0.2%	0.1%	0.2%	0.3%	0.4%	0.4%	0.4%	0.2%	0.2%	0.2%	0.1%	0.1%	0.1%	0.2%	0.1%	0.1%	0.1%	0.2%	0.1%	0.1%	0.2%	0.1%	0.1%	0.1%	0.14%	249	231
Tools	10.9%	12.2%	14.6%	17.6%	20.3%	21.5%	21.4%	18.6%	14.7%	10.4%	8.4%	6.8%	6.1%	5.9%	5.9%	5.9%	6.0%	6.1%	5.8%	6.0%	6.3%	6.8%	7.4%	9.1%	7.89%	2,512	1,688
Travel	1.4%	1.6%	2.1%	2.2%	2.4%	2.6%	2.2%	1.9%	2.0%	2.1%	2.0%	1.8%	1.9%	1.9%	1.9%	1.8%	2.0%	1.9%	2.2%	2.2%	1.9%	1.7%	1.6%	1.4%	1.86%	1,752	407
Unknown	4.7%	5.3%	5.1%	5.0%	5.3%	4.4%	5.0%	5.9%	4.6%	4.4%	4.1%	3.8%	3.5%	3.8%	3.7%	3.7%	4.0%	3.6%	3.7%	3.7%	3.7%	3.9%	4.1%	4.5%	3.88%	2,284	1,796
Total Launches per Hour	103,604	77,053	53,633	40,332	33,438	30,949	38,161	56,895	83,488	109,550	127,069	142,642	158,876	168,082	169,018	172,935	173,963	179,801	184,012	176,050	163,080	153,835	141,303	123,639			

Figure 3.7: Hourly relative application usage by category in terms of launches. Each cell value refers to the percentage of application launches done by our users within each hour for each category. Colors are normalized by row, with green indicating each category's maximum percentage of application time, and white indicating each category's minimum. For example, games reach their peak in the evening (green) and trough in the morning (white).

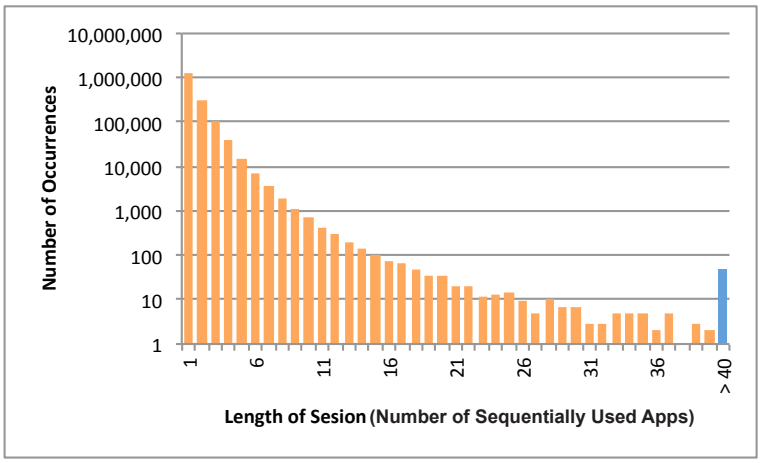


Figure 3.8: Number of applications used in a session. We aggregated sessions longer than 40 applications since the graph flattens out and scarcity increases. The maximum number of applications in one session is 237.

interaction with some applications to quickly inspect the applications' changing contents.

We also looked into the number of *unique* applications used within a session, as can be seen in Figure 3.9. The first bar in this figure is of course identical to the first bar in the preceding Figure 3.8. We found a maximum of 14 unique applications in a single application chain. A vast majority of users use a very small number of unique applications during an interaction with their device. Thus — according to our analysis of sessions — people who use more than 14 applications in sequence tend to re-use applications that they already have launched previously within the same session.

Examining the amount of time our participants spent in each application chain, we found that 49.8% of all recorded sessions are shorter than 5 seconds, which appears to be rather short. The longest session we observed took 59 minutes and 48 seconds. Between these upper and lower limits, the curve has a similar exponential decay to that in Figure 3.8 and 3.9.

Probably the most revealing statistic in our analysis of application chains is that for nearly half of all chains (49.60%) the first application belongs to the category *Communication* (as Figure 3.10 shows). Digging deeper, we found that 15.7% of the chains within our sample were initiated with short messaging (SMS) application (9.5% default pre-installed SMS application, 6.2% a application called *Handcent*

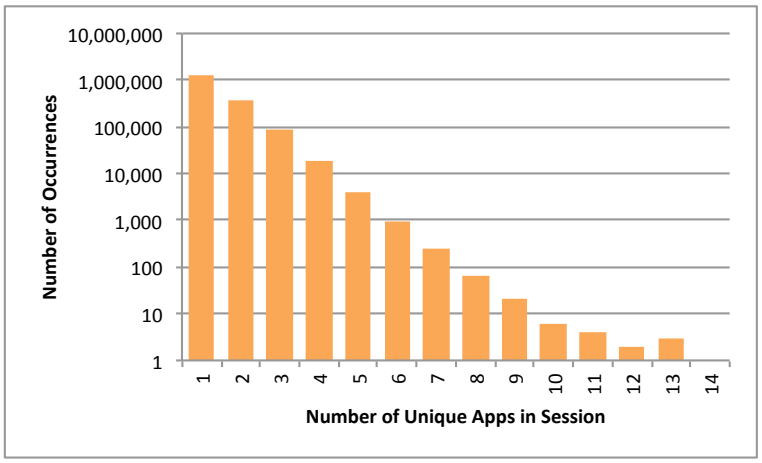


Figure 3.9: Occurrences of sessions versus number of unique applications used within a session.

SMS), 9.6% with the phone application, and 5.9% with the standard mail application. Interestingly, a browser was only used first in 5.9% of the application chains.

Apart form length of the sessions, we also looked into the cohesion of applications used within single sessions. Figure 3.11 shows the transition probabilities between application categories in an application chain. Accordingly, the diagonal of the figure indicates transitions from one application to another in the same category. As such, the values along the diagonal can be non-zero. It is worth noting that this graph obviously considers only those sessions where two or more applications have been used. For each application, it is very likely that the application used next is a communication application, except for *News* and *Lifestyle* applications. Apart from these two categories, the probability that the next application is a communication application is at least 23.2% for all categories. For communication applications, there is a 66.5% chance that the next application will be a communication application again. This is the highest probability for users to stay within one category. Next there are *Tools*, with a probability of 15.7% of staying within *Tools*, and *Games* with a probability of 15.1% of staying within *Games*. It can also be seen that applications from category *Tools* are entered relatively frequently from applications of any other category.

There are also some important unique connections between application categories. Most notably, a browser is opened quite frequently following the use of a news application. The connection between *Lifestyle* and *Shopping* applications is also

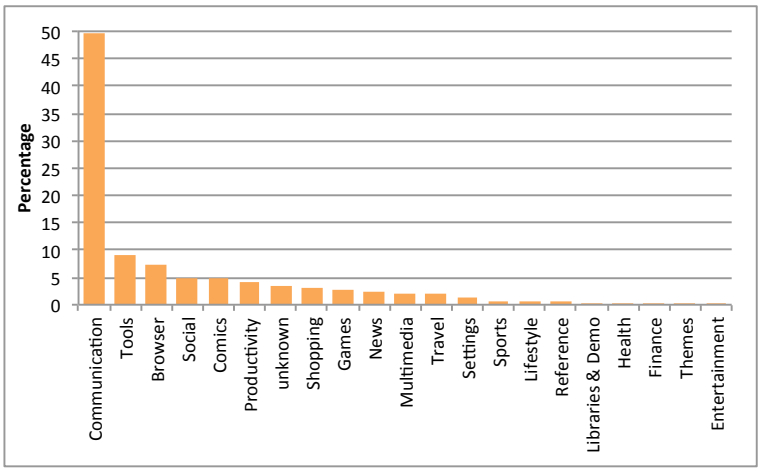


Figure 3.10: Categories of first-used application within a session.

quite strong, with *Lifestyle* applications frequently leading the user to enter into a *Shopping* application. The reverse is also true, but to a lesser extent.

Application Usage by Location

Besides local time of our participants and preceding applications, we also analyzed spatial information that the *AppSensor* provided. We also found clear empirical evidence for location as a covariate of application usage behavior. This occurs across changes in both administrative regions (e.g. USA vs. Europe) and functional regions (e.g. inside airports vs. outside of airports). Some initial findings from our spatial analysis are as follows.

We examined 13,190 samples recorded by the *AppSensor* that occurred while a user was located within the spatial footprint of a known airport in the United States. We found that while in the airport, users were 2.78 times more likely to be using a browser (by usage time) than users located in the United States as a whole. This may suggest that certain functions related to air travel may not be sufficiently migrated into native applications (e.g. looking up flight status). On the other hand, users were less likely to be using games, tool applications, or reference applications while in airports. This was somewhat surprising, especially given that the *Kindle* application belongs to the reference category.

The location API also provided us with speed information related to application usage as traces by *AppSensor*, and in contrast to Do et al. [85] we also were able to relate application usage to participants' movements. We found that when traveling

	Browser	Comics	Communication	Entertainment	Finance	Games	Health	Libraries & Demo	Lifestyle	Multimedia	News	Productivity	Reference	Settings	Shopping	Social	Sports	Themes	Tools	Travel	Unknown	Samples	Users	Apps
Browser	2.4%	3.6%	33.8%	0.0%	0.3%	3.5%	0.2%	0.2%	0.4%	1.5%	11.8%	3.8%	0.6%	1.7%	3.6%	15.6%	0.5%	0.3%	8.1%	2.2%	6.1%	48,379	2,193	9
Comics	6.5%	9.4%	36.1%	0.0%	0.2%	4.8%	0.6%	0.2%	0.6%	5.2%	2.7%	4.1%	0.6%	2.2%	5.2%	4.3%	0.6%	0.4%	8.4%	2.7%	5.0%	31,258	1,754	1,220
Communication	5.7%	2.7%	65.5%	0.0%	0.2%	1.5%	0.1%	0.1%	0.2%	1.3%	2.1%	2.5%	0.3%	1.0%	1.7%	4.8%	0.4%	0.1%	5.0%	1.4%	3.2%	434,974	2,839	449
Entertainment	6.7%	6.1%	26.1%	0.0%	0.0%	3.3%	0.6%	0.0%	0.6%	5.6%	0.6%	2.8%	0.0%	3.3%	7.2%	3.3%	3.3%	0.0%	8.3%	5.6%	16.7%	180	65	28
Finance	10.3%	3.7%	37.3%	0.0%	1.8%	2.9%	0.2%	0.3%	0.2%	1.5%	8.6%	3.5%	0.1%	1.5%	5.5%	6.1%	0.7%	0.1%	10.6%	1.9%	3.1%	1,496	347	117
Games	11.8%	5.9%	30.4%	0.0%	0.3%	15.1%	0.3%	0.4%	0.7%	1.0%	2.1%	4.2%	0.7%	1.5%	6.5%	4.0%	0.8%	0.1%	8.3%	1.7%	4.2%	8,620	1,077	995
Health	3.8%	4.8%	34.3%	0.0%	0.3%	2.5%	6.1%	0.6%	1.2%	6.1%	2.9%	3.1%	1.6%	2.3%	6.0%	4.9%	0.8%	0.0%	12.4%	2.3%	3.9%	1,466	328	130
Libraries & Demo	6.0%	3.7%	23.3%	0.0%	0.2%	2.3%	0.3%	2.6%	0.8%	1.3%	1.7%	3.2%	0.3%	16.2%	11.9%	3.7%	0.3%	0.1%	13.4%	3.2%	5.5%	3,936	1,082	90
Lifestyle	8.2%	5.3%	17.3%	0.0%	0.1%	4.0%	0.5%	0.6%	3.0%	0.9%	2.3%	4.3%	0.7%	2.3%	28.7%	3.1%	0.2%	0.4%	10.2%	2.2%	5.5%	4,673	1,383	303
Multimedia	6.2%	10.5%	38.2%	0.0%	0.2%	1.4%	0.6%	0.2%	0.4%	2.5%	2.5%	6.2%	0.3%	2.0%	1.8%	4.4%	0.3%	0.4%	9.5%	3.2%	9.1%	12,451	1,376	53
News	33.6%	3.3%	33.3%	0.0%	0.5%	1.6%	0.2%	0.1%	0.2%	1.4%	3.9%	2.9%	0.4%	1.4%	3.0%	3.7%	0.4%	0.0%	6.5%	1.0%	2.4%	25,131	1,440	312
Productivity	7.4%	5.0%	38.5%	0.0%	0.4%	2.6%	0.4%	0.2%	0.6%	2.8%	2.8%	7.2%	1.1%	3.8%	4.8%	5.1%	0.6%	0.3%	9.7%	2.4%	4.4%	31,113	1,954	498
Reference	13.1%	4.5%	34.3%	0.0%	0.2%	7.5%	0.6%	0.3%	1.0%	1.0%	2.5%	4.6%	2.9%	1.7%	5.2%	4.1%	0.4%	0.2%	9.8%	1.7%	4.4%	2,611	552	199
Settings	8.9%	5.6%	26.3%	0.1%	0.2%	1.8%	0.4%	5.2%	0.7%	2.0%	2.6%	6.9%	0.5%	0.0%	5.6%	4.7%	0.6%	0.5%	11.6%	4.8%	11.1%	13,576	1,863	1
Shopping	8.5%	7.8%	23.2%	0.0%	0.4%	4.8%	0.4%	0.9%	9.6%	0.9%	2.8%	5.2%	0.7%	3.0%	4.7%	4.3%	0.5%	0.5%	16.6%	1.6%	3.8%	21,788	2,207	132
Social	24.1%	3.0%	35.3%	0.0%	0.3%	2.3%	0.2%	0.2%	0.3%	1.2%	2.9%	2.8%	0.3%	1.5%	2.7%	12.4%	0.7%	0.1%	5.3%	1.2%	3.3%	35,086	1,593	239
Sports	7.4%	4.3%	43.3%	0.1%	0.4%	2.5%	0.4%	0.2%	0.3%	1.3%	3.0%	4.8%	0.5%	2.4%	3.8%	5.4%	7.6%	0.0%	7.0%	1.5%	3.9%	2,793	387	135
Themes	8.5%	10.2%	37.2%	0.0%	0.2%	2.4%	0.1%	0.2%	1.4%	3.2%	0.4%	4.7%	0.4%	3.3%	6.5%	3.6%	0.1%	1.2%	8.6%	3.3%	4.6%	1,929	175	175
Tools	11.0%	5.1%	36.1%	0.0%	0.2%	2.7%	0.3%	0.4%	0.6%	2.1%	2.4%	4.2%	0.6%	2.1%	5.5%	4.1%	0.4%	0.2%	15.7%	2.8%	3.5%	88,911	2,384	1,310
Travel	6.7%	9.1%	36.2%	0.1%	0.2%	2.3%	0.3%	0.5%	0.7%	1.9%	1.6%	6.7%	0.4%	5.0%	2.9%	4.4%	0.3%	0.2%	10.2%	6.6%	3.6%	12,556	1,403	281
Unknown	10.7%	4.4%	40.8%	0.1%	0.2%	2.1%	0.2%	0.3%	0.6%	3.9%	1.8%	3.2%	0.3%	3.9%	2.9%	4.7%	0.3%	0.2%	6.4%	1.5%	11.6%	48,379	1,972	1,277

Figure 3.11: Transition probabilities in application chains. The transitions are from categories in a row to categories in a column. The diagonal indicates transitions between applications in the same category. The probability ranges from yellow (low) to green (high).

at speeds greater than 25kph, not surprisingly that users were more than 2.26 times more likely (by usage time) to be using an application of the *Multimedia* category, to which most music-related applications belong. Interestingly, we found that they were less likely (0.83 times) to be using applications in the *Travel* category.

Looking into differences between groups of our participants, we found some interesting differences between users in the United States and in Europe. European users are 1.21 times more likely to be found using a browser (by usage time). Americans, however, spent relatively much more time with sports, health, and reference applications. Social and news applications were the most equally used.

Specific Application Usage

Although we focused our analysis at the application category level, we did analyze several important and/or well-known individual applications to get a deeper understanding of how people make use of their applications. Figure 3.12 shows the usage times of specific applications with regard to time of the day. In contrast to Figure 3.7, the numbers in Figure 3.12 are not normalized by total usage over all applications within those hours, but by each application's total usage per day.

Previously, we saw that social applications in general have their highest probability of use in the evening. This is somewhat true for Facebook, but its usage time is

	12am	1am	2am	3am	4am	5am	6am	7am	8am	9am	10am	11am	12pm	1pm	2pm	3pm	4pm	5pm	6pm	7pm	8pm	9pm	10pm	11pm	% of Total Usage Time	Users
Facebook	4.8%	3.9%	4.0%	3.4%	3.2%	3.3%	3.8%	4.1%	4.1%	4.1%	3.9%	4.7%	4.0%	4.2%	4.1%	3.5%	4.1%	4.0%	4.3%	4.5%	4.7%	4.8%	5.6%	4.8%	1.91%	1,467
Google Maps	2.9%	1.7%	2.0%	1.8%	1.8%	1.8%	1.9%	2.2%	3.2%	4.0%	4.0%	5.0%	5.7%	5.6%	5.7%	5.8%	6.8%	6.4%	7.3%	6.6%	5.0%	4.8%	4.6%	3.4%	0.81%	1,584
Alarmclock 1	6.6%	8.2%	8.8%	10.0%	10.4%	9.7%	8.7%	6.8%	4.8%	3.3%	2.2%	1.4%	1.0%	1.0%	1.0%	1.0%	0.9%	0.6%	1.0%	1.2%	1.4%	2.0%	3.0%	5.0%	4.55%	341
Alarmclock 2	3.9%	4.3%	7.5%	9.2%	10.8%	10.7%	9.7%	9.2%	8.2%	7.7%	5.3%	3.3%	1.8%	0.5%	0.4%	0.3%	0.5%	0.6%	0.7%	0.4%	1.0%	0.6%	1.1%	2.0%	0.32%	169
Weather App	2.1%	0.9%	0.6%	0.3%	0.5%	2.0%	3.8%	2.2%	4.0%	10.1%	11.2%	9.8%	8.1%	3.2%	2.9%	5.9%	6.5%	4.3%	2.0%	4.9%	3.3%	4.0%	4.7%	2.5%	0.06%	309
Twitter	3.6%	3.3%	3.6%	4.4%	3.3%	3.0%	3.8%	4.0%	4.3%	4.3%	4.8%	4.7%	4.3%	4.4%	4.7%	4.6%	4.0%	4.1%	4.0%	4.6%	4.9%	4.8%	4.3%	4.1%	0.56%	457
Phone	2.6%	2.2%	1.9%	1.9%	1.8%	1.9%	1.9%	1.7%	2.4%	3.3%	4.1%	4.8%	5.2%	5.8%	5.7%	6.5%	6.4%	7.4%	7.6%	6.8%	5.8%	4.8%	4.1%	3.5%	1.94%	2,409
Angry Birds	5.3%	4.3%	3.2%	2.4%	1.6%	1.8%	1.5%	2.0%	1.8%	2.8%	3.5%	3.5%	4.6%	5.6%	4.9%	6.5%	4.7%	6.2%	5.6%	6.0%	5.1%	6.0%	5.4%	5.7%	0.64%	727
Kindle	9.1%	7.7%	6.9%	5.5%	4.0%	3.2%	2.9%	2.3%	1.7%	2.0%	2.3%	1.9%	2.8%	3.6%	2.6%	4.3%	2.5%	3.8%	3.1%	2.5%	4.9%	5.3%	7.3%	7.7%	0.47%	209
Calculator	2.6%	2.3%	2.0%	0.8%	0.7%	0.8%	1.1%	1.7%	2.8%	1.6%	6.6%	6.8%	7.1%	6.2%	5.6%	7.6%	7.7%	6.9%	5.8%	5.3%	6.8%	6.2%	3.3%	1.8%	0.19%	650
Calendar	5.1%	3.6%	0.7%	0.4%	0.2%	0.4%	3.8%	2.2%	3.9%	6.1%	7.6%	7.8%	6.3%	5.3%	3.5%	5.5%	5.8%	5.3%	3.8%	4.4%	5.6%	4.3%	3.1%	5.1%	0.14%	615
Camera	4.2%	4.3%	2.7%	3.5%	2.7%	2.0%	3.0%	2.2%	1.3%	2.6%	3.6%	3.2%	4.4%	5.7%	5.0%	5.1%	6.2%	6.7%	6.0%	4.7%	5.0%	5.8%	4.5%	5.7%	0.19%	781
Music	2.0%	3.6%	4.5%	5.1%	5.3%	5.4%	6.2%	6.1%	5.8%	4.5%	5.3%	3.7%	4.2%	3.8%	3.6%	3.6%	3.4%	3.6%	3.1%	4.1%	3.6%	3.8%	2.9%	2.8%	0.41%	483

Figure 3.12: Application usage time throughout the day. Within each row (i.e., for each app) low usage is indicated by white, increasing through yellow and reaching a peak at red. Percentages indicate the usage time of each application and are normalized within each row.

spread out throughout the whole day. The same goes for Twitter, although it is not as much of a late-night activity.

A somewhat surprising finding can be gleaned by looking into the usage of the *Google Maps* application (*Travel*), which has a relatively strong peak in the early evening hours. Traffic checking is perhaps one possible cause, although one would expect this pattern to be repeated during the morning commute. Another interesting result concerns the use of the built-in music applications, which is somewhat focused in the morning hours. This also might be a result of participants listening to music when commuting to work or to school.

Weather checking is, not surprisingly, largely a morning activity, as is checking one's appointments for the day on the calendar. On the other hand, users' desire to play *Angry Birds*[6] is absent in the morning, and only picks up in the early afternoon and in the evening. *Kindle* usage behavior is even more focused in the late evening.

Another interesting phenomenon emerged from the study of two different alarm clock applications. It seems that alarm clock applications are mostly used — i.e., being the only active application on the device presenting its user interface — during the night (from 2am until 9am). One reason for this might be that people "use" this type of application while sleeping, e.g., as a desk clock, which prevents the device from going into standby mode.

More generally speaking, some applications have spikes in usage and are more destined for special contexts of use — as Figure 3.12 shows — whereas other

[6] A game where one has to shoot pigs by flinging birds at them; see http://market.android.com/details?id=com.rovio.angrybirds, last accessed on 04.07.2013.

applications are more broadly employed throughout the whole day, and also are used at different locations.

3.3.3 Implications and Discussion

The results that we describe above give rise to new implications for improving the design of smartphones, and more particularly for smartphone launchers. However, our study also has some limitations that we will discuss in this section. Further, we discuss which other fields the *AppSensor* can be applied to when systems can benefit from tracing application usage data.

Implications for Design

The results reported in this chapter can be used to improve the design of mobile applications and mobile operating systems. For instance, designers of *"launcher"* applications (like the home screen on the iPhone and Android) could vary application icon position and/or size based on the time of day and/or the user's location. This same idea could apply with regard to an application chain with the last application opened providing the context rather than time/location. Similarly, application developers could design smart links between applications that are used frequently in sequence, which could further benefit from adding functionality like transferring pieces of data from one application into another. Since people often navigate from lifestyle applications to shopping applications, application designers of the former might implement links to shopping applications. Additionally, the *AppSensor* gives insights into the applications' contexts of use. For instance, the design of applications can be optimized if the developers know whether an application is used only while commuting in the morning, or solely in the evening.

Our results show that smartphones are — despite their evolution towards multipurpose devices that we sketched in Chapter 1 — still first and foremost communication devices. This is not only due to phone calls, as smartphones provide a variety of new ways to communicate (e.g., instant messengers, email, voice over IP, video chat). Nevertheless, our findings certainly qualify the mobile phones as multifunctional tools in the *"Swiss Army Knives"* [215] line of thinking. That being said, when people are not sleeping during the late hours of the night and early morning hours they make more use of the non-communication functionality provided by different kinds of applications. Additionally, they spend more time

within an application once they have opened it in the night. One reason might be that fewer people are awake for active synchronous communications at that time.

Our results also suggest that users frequently switch between applications they have already used in application chains of single sessions, rather than only opening new applications. This suggests that there is a functional cohesion between the particular utilizations of single applications. As such, mobile phone operating systems should better support navigation between very recently used applications.

Making Use of the AppSensor

The *AppSensor*, which we introduced earlier, makes it possible to examine the ecosystem of applications residing on a user's device. This has potential to inform the design and customization of novel applications, or even new devices themselves.

One may apply the *AppSensor* to infer a user's context based on her actually used applications. According to Dey [80], context awareness involves adapting services according to a user's context. For instance, the users' needs for mobile services — i.e., applications in our case — depend on their locations [137]. Reasoning about the context of a user based on sensor data while she is interacting with a device usually involves uncertainty and leads to ambiguity [82]. We propose that by adding the *AppSensor* to context reasoning, one can decrease the uncertainty and ambiguity of context recognition. An example is sketched in Figure 3.13: Even though two people may be walking through the same pedestrian mall in a famous city (i.e, they have the same location), if they use different applications (e.g., one a shopping list application and the other a sightseeing app) we would be able to distinguish between the shopper and the tourist, even though other sensors like location, time, and acceleration might provide similar values. Even without any meta-information on the in-use application itself, it would be possible to compare the contexts of two or more people. For instance, if two users are constantly swapping between a map application and a restaurant guide application, they might be doing the same activity — probably looking for a restaurant, as we saw in the example of application chains where the participant was searching for a spot to get a coffee. This idea follows Want's [258] line of thinking that mobile phones are becoming a proxy for their owners' activities, and the Watson system presented by Budzik and Hammond [52] which leverages interactions with desktop applications as contextual information.

Figure 3.13: Incorporating *AppSensor* into context-reasoning. This allows one to distinguish different activities in cases where other sensors provide the same data. In this example one user is using a tourist application (left), and the other one is using a shopping list application (right). Based on this *AppSensor* information, we are better able to distinguish the two activities of sightseeing and shopping near the mall (composed with screenshots showing applications *Stockholm City Guide* and *Out of Milk Shopping List*, background image showing the Saluhall shopping mall in Stockholm, Sweden; photo provided by Holger Ellgaard under Creative Commons).

Context-aware recommender systems that suggest mobile applications can also be made more efficient by exploiting an *AppSensor*, as we will discuss in Chapter 5. For instance, recommender systems that follow a post-filtering approach — i.e., applying knowledge on context-aware dependencies after using basic techniques like collaborative filtering [2, 120] — can exploit the time-dependent usage share as a factor in the estimated ranking of applications. We will present a context-aware recommender system using the *AppSensor* in Chapter 5.

Limitations of Our Study

Some applications have a more general purpose that is not well understood by *AppSensor*. For instance, a web browser can be used for everything from public transportation route planning to looking up a word in a dictionary. The meaning that can be deduced from such applications can be regarded as limited or imprecise. For these cases, the insight that the *AppSensor* provides on the user's context might be limited. However, most services that are provided via a browser are also

available within dedicated applications. Since many users seem to prefer to employ native applications instead of websites on mobile devices [7], this should not have a large negative impact. Thus, when applying the *AppSensor* one should be aware of this issue.

The current design of the *AppSensor* is not capable of determining when users are using applications in parallel. For instance, if a user is listening to music — with the player running in the background from the operating system's perspective — and is browsing the Internet at the same time, the *AppSensor* will return the browser as the single open application that is being used. Similarly, on the Android platform we have the problem that applications' widgets are part of the UI of the home screen application. Therefore we cannot measure the widget-related usage of applications. However, most widgets are simply entry points into applications, i.e., when users want to use the functionality provided through the widget, they would open the corresponding application automatically by clicking on an element of the widget.

While we have no detailed information on the participants due to the domain of the underlying platform *appazaar* — i.e., helping people to discover new applications — we may assume that some of our users are early adopters with a high affinity for trying out new applications. Thus, our participants in general may have a slightly higher affinity towards mobile application usage and be a little more tech-savvy than the general population.

Like every sensor, the *AppSensor* is not error-free. For instance, our sensor might return values that do not relate to the user's current activity. A user might leave and put away the device with an application still running. The uncertainty of the reasoned context will increase with the time that the user has not used her device. However, most devices go into standby after some time of non-usage, as long as the user does not intentionally use an application that prevents standby. Moreover, application usage that occurs when standby mode is purposefully disabled can also be assessed as a valid value of application usage as returned by *AppSensor*.

Furthermore, the *AppSensor* cannot be used to reason about a user's context when no application is used at all, i.e. when the device is in standby or turned off. In the formal definition of the *AppSensor* we dedicated a special symbol to this case. Secondly, the sensor is obviously only available during active usage of the device.

[7] AppsFire.com: *Infographic: iOS Apps vs. Web Apps*. http://goo.gl/1O2rf, last accessed on 13.06.2013.

Otherwise it can only be deduced that the user is currently not using his device. Thirdly, the accuracy of the *AppSensor* also depends on its sample rate. This impacts the quality of the measured data. The sample rate needs to be chosen depending on how often a user is switching between different applications. If the swapping frequency is higher than the sample rate, the accuracy will decrease. However, at a high frequency the system load might increase and impact power consumption. We believe that our sample rate is correctly set given these constraints, as we conducted an informal pre-study on how fast one can switch between and start a new application.

Of course, our findings cannot be transferred to general usage of the underlying types of services and use cases. For instance, it might be the case that people use Facebook during the day on their stationary computer or laptop, and use their smartphones when they are lying in bed in the evening.

Technically, whether or not an *AppSensor* is widely deployable within a system strongly depends on the underlying operating system and the policies of the device's vendor. The *AppSensor* used in this work was implemented based on the Android platform because the Android operating system provides the required openness and APIs to trace system-internal information related to the execution of applications. The sensor itself needed to be implemented as a background service, which would not be possible on every device. For these and other reasons the implementation of the *AppSensor* would not possible on Apple's iPhone, or at least cannot be deployed in the wild.

3.4 Adaptive Menu to Support Launching of Applications

Current smartphone usage can be characterized by devices that have many applications installed. In Chapter 5 we will investigate people's habits for organizing their applications so that they can easily find the right application that they want to start. The complement to people's strategies for coping with the problem of finding applications on their devices — as we will discuss in Chapter 4 — is to help them find the application that they want to start. This form of support results in application launcher menus that adapt to the user's application usage habits. This idea of adaptive launcher widgets for smartphones has existed for some time (see for instance [44, 253, 31, 268]).

The time span from deciding to use an application to actually being able to utilize the application can be separated into two parts: (1) The time to find the application before launching it, and (2) the time it takes for the application to load before the user can start using it (i.e., the application has to be loaded into the operating system's memory to be executable, and many applications need to download up-to-date content from the Internet before they can be used, e.g., news feeds) [268]. In this work, we only focus on the first point, since the second point is related more to a operating system's perspective rather than to an HCI perspective (related approaches for pre-loading and pre-fetching applications and content can be found in [183, 265, 14]).

Based on the *AppSensor*, which we introduced and designed previously, we have built an adaptive menu that provides people with shortcuts to applications that they might want to start next, which we called *AppKicker*. Further, our approaches for building strategies to adapt the menu to the user's application usage behavior have also been informed by the results of the study we presented. In this section, we will describe the design of the *AppKicker* system, and explain the approaches that we implemented to retrieve sets of applications that we would provide shortcuts for.

3.4.1 Design of Adaptive Menu

Zhang et al. [268] show that the time people spend searching for an icon to launch the corresponding application positively correlates with the distance of the icon from the top of the screen. This suggests that people would have to perform a visual search in a fully adaptive menu starting at the top, with search time being linear to the number of icons in the menu, if they are unfamiliar with the items [224]. This is the case for adaptive menus, since the user cannot know where the application that he wants to launch would show up as he cannot build a mental model of the menu [270], and cannot apply any means to benefit from motor memory [108] by arranging the launcher layout for usability reasons (cf. Chapter 4). The main reason for this deficiency lies in the very nature of an adaptive menu: Every icon may change its position whenever the menu is updated; in fact it may also be the case that icons disappear and new icons show up, even all of them. To provide benefit to the user, such a full-screen adaptive menu would require a highly accurate prediction algorithm — especially since smartphones are small-screen devices [95] — that ideally presents the next application's icon in the top row, to keep visual search time low. One cannot require the user to always scan through the whole list

of icons on their menus if the full screen presents an adaptive menu. In fact, Shin et al. [229], who have built and studied a full-screen adaptive menu, found that too many changes in the launcher menu leaves users feeling confused and out of control — despite the benefits provided to the user by adaptive launchers. Further, many users have very specific habits for arranging their smartphone launchers as well as very subtle habits for customizing their launchers, as we will discuss in Chapter 4.

Therefore, we conclude that when providing support for finding icons to launch the applications, designing the launcher as a full-screen adaptive menu is not the optimal approach — though it might seem to be a holistic one. Therefore, the design rationale of the adaptive launcher menu *AppKicker* that we present is twofold: On the one hand there is a benefit from an adaptive menu that anticipates the next application to provide support for launching of applications to more quickly and easily find applications within launcher menus, but on the other hand occupying the full screen with an adaptive menu is not the best approach and we would prefer to leave space for the user to customize parts of her launcher menu on her own.

As a result, we designed a solution that instead follows the approach of two-split menus [224], with one fixed part and one adaptive part. Figure 3.14 shows the design of the split menu that we have built for launching mobile applications. At the top it shows one row of icons that will adapt to the user's current application usage. This part is customizable by the user in that he can add it to his launcher wherever he prefers to have it. Below it are shown stock Android controls and other elements that users are able to customize in their own way, e.g., add other widgets and shortcuts to various applications as we will discuss in Chapter 4.

3.4.2 Implementation

We implemented the approach of adaptive smartphone launchers as designed in the previous section for the Android operating system. For the end-users of our system we only provide the adaptive part of the designed two-split menu, and they can add this part as an extension to their launchers. The UI of the *AppKicker* system itself, i.e., the adaptive launcher menu providing shortcuts to the user's applications, was implemented as a widget on the Android platform, which is shown in Figure 3.14. The widget has space for icons of five applications that the user can use to launch these applications. To be adaptive to a user's actual use of mobile applications the

Figure 3.14: *AppKicker's* menu placed on the homescreen as a widget.

AppKicker application is based on the implementation of the *AppSensor* presented earlier in Section 3.2. Figure 3.15 shows the overall architecture of our approach. *AppSensor* serves as a vital core for *AppKicker*, and the traces of application usage are made available to different subsystems. In our study of mobile application usage we found that people have characteristic behaviors to open specific applications right after unlocking their devices (cf. also reported as the checking habit by [179]). Therefore, we extended the implementation of *AppSensor* for the realization of the *AppKicker* application to also recognize such events in addition to mobile applications' lifecycles.

To assign applications to *AppKicker's* shortcut menu, we implemented six different models, which all use an instance of the *AppSensor* introduced previously to trace the user's application usage. As Figure 3.15 also shows, the traces of a user's application usage are being persisted for the different approaches that we implemented to assign icons to the *AppKicker* launcher menu. Each model keeps track of those pieces of information about the user's application usage and contextual information that it requires to anticipate which applications might be launched next.

- *Most-recently used (MRU)*: The MRU-based model assigns the top 5 most recently used applications to the set of icons that will be presented to the user. These last 5 applications are ordered by time: the most recently used application will be shown at the first position. This approach is state-of-the art and available on many devices for switching through applications (e.g. as

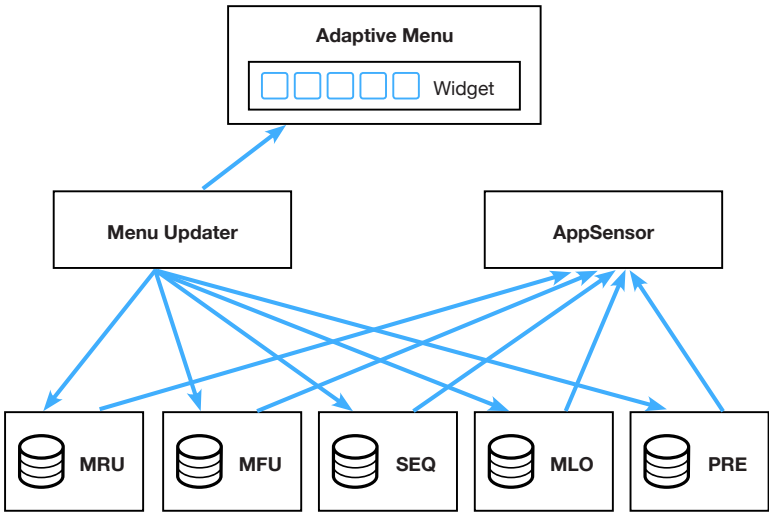
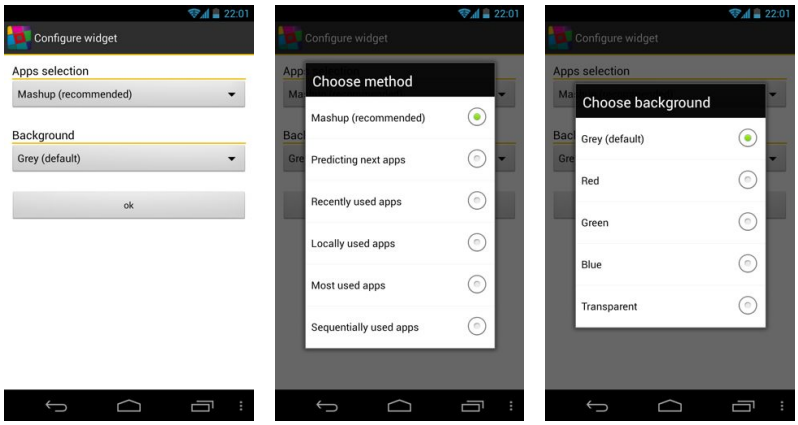


Figure 3.15: Conceptual architecture of the *AppKicker* application.

a kind of task bar under iOS and in a dedicated recent applications menu on the latest version of the Android OS). This model is informed by the finding that people typically only use a few applications within one session, although they may switch between those applications. Therefore, when using this MRU-based model the menu will always show applications which are very likely to be used next, if they already have been used previously — within the recent five applications.

- *Most-frequently used (MFU)*: The MFU-based model assigns the icons of those applications to the widget which in total have been used most; i.e., those applications which the *AppSensor* has logged the most launches of. As such, this model is based on a simple unconditioned probability of which applications are most likely to be used next.
- *Sequentially used (SQU)*: The SQU-based model is informed by the finding that there is a certain cohesion between launches of applications, as we found in the previous study. Therefore we have implemented a model based on simple conditional probability for predicting the usage of an application if another one was used previously.
- *Most-used at location (MLO)*: The MLO-based model aims on supporting users that have strong location-correlated application usage behavior. Therefore, we implemented this simple model that is conditioned by the current location of the user. For instance, when at her office the user will have those applications in the shortcut launcher that she mostly uses at the office; while



(a) Customization dialog (b) Choosing method (c) Customizing widget

Figure 3.16: Customization of the adaptive launcher menu provided by the *AppKicker* application. The user has the possibility to customize *AppKicker* widget when adding it to the homescreen

at home she will have those applications in the shortcut launcher that she has used at home most.

- *Predicted next application (PRE)*: We also used a model that is based on a more sophisticated approach for mobile application prediction, which is a follow-up model of the one presented in [265]. This model for prediction of mobile application usage is based on an algorithm for text compression, which has been adopted for working on sequences of mobile application usage [183].[8]

By design, these five different approaches for assigning applications to the adaptive menu have certain characteristics. While for instance the MRU- and MFU-based models will be able to provide applications right after a single first application was used, the SQU- and MLO-based models will only return applications when other applications already have been used after a particular application or at the current location. That being said, the latter two models require some training time and some time to learn how people use their applications. The PRE-based model, by contrast, requires zero training time [183].

Since it is important for end-users to customize their personal smartphone (cf. [109] and Chapter 4), we added simple capabilities to change the background color

[8]This model's formulation, implementation and evaluation is not part of the work presented in this thesis. It resulted from a collaboration of the author with researchers from University of Massachusetts Amherst and Microsoft Research, and results are published in [183].

of the widget to increase user adoption, as shown in Figure 3.16(c). In fact, this was a feature that users of the *AppKicker* application requested on the application store by leaving comments on earlier versions of our application that we released to the Google Play Store.

As Figure 3.16(b) shows, there is one additional mode that the user is able to choose from to configure a new installation of the widget, which is called *"Mashup"*. This actually is not an additional model on its own, but rather is based on the aforementioned five models for counterbalancing. If a user decides to use this mode for her adaptive menu, the system will randomly choose one of the presented models for assigning application to the launcher menu. This mode allows us to study the different models for application prediction against each other; therefore it is also marked as "recommended" to the user (see Figure 3.16(b)), for no particular reason, except to get as many users as possible into this condition. The list for selecting the other models, as shown in Figure 3.16(b), is always in random order, so that there is no tendency for users to choose one model over another biased by the ranking of items in the list.

3.4.3 Case Study

For a proof of concept we were interested in how people make use of the adaptive menu that we provided in general, and how they would use the different models that we implemented. Therefore, we made the *AppKicker* application available on the Google Play Market for end-users to download, install and use.

Study Design

The design of our study follows the approach of research through the application store, which we describe in Chapter 2 and which we already used for the deployment of the *AppSensor* in the Android market. The *AppKicker* application was made available for download on the Google Play Market.[9]

We compared three of the algorithms that we implemented within the "Mashup" mode that we dedicated to conducting a counterbalanced study. For a fair comparison, we decided to only use those algorithms which are informed by tracing only application usage without additional context information: We have chosen to compare the most-recently-used model (MRU), which is used by all major mobile

[9] See http://goo.gl/mSSN6, last accessed on 06.07.2013.

platforms to show application shortcuts; the model based on sequentially-used applications (SEQ), which uses only the previously used applications to predict the next application; and the predictive model that we implemented based on related work [183] (PRE).

We used a within-subject A/B/C design to test and compare the different prediction strategies. As described above for the "Mashup" mode, we randomized the conditions: Whenever a new set of icons to be shown in the adaptive menu is requested, we randomly choose one of MRU-, SEQ- or PRE-based models for estimating a set of applications. As such, for this study we discarded the data from the modes that participants configured for themselves.

Within the widget, we tracked how users interacted with the icon menu for each of the models. However, our evaluation is limited in that we cannot know what other icons users have placed on their home screens. For example, we might schedule applications into the adaptive launcher widget that the user has already pinned to her home screen (Chapter 4 discusses how people customize their homescreens and launcher menus), or we might not catch application launches that the user initiates outside of our launcher widget.

Findings

At the time of this study, the widget had been installed more than 43,600 times and had more than 7,630 active users.[10] Note that both the installation of *AppKicker* and the participation in the research study are voluntary, and as such participants self-select their study role. As soon as a user starts the *AppKicker* application for the first time, we present a research study disclaimer and ask for an opt-in consent. Following the *Two Buttons Approach* [187], users can decline from contributing data but still use the application.

For a fair comparison of the different prediction algorithms, we report results for the top 100 users as ranked by the number of application launches participants made using the widget. These users had a median of 112.5 clicks on icons for application launches (min 62, max 1,807). This resulted in 16,991 clicks in total.

Figure 3.17 shows the click-through rate as a function of the prediction algorithm's ranking. The PRE-based model yields a click-through rate of 38.1% on its top ranked app, whereas MRU and SEQ produce only 27.7% and 35.4% respectively.

[10] According to Android Developer Console.

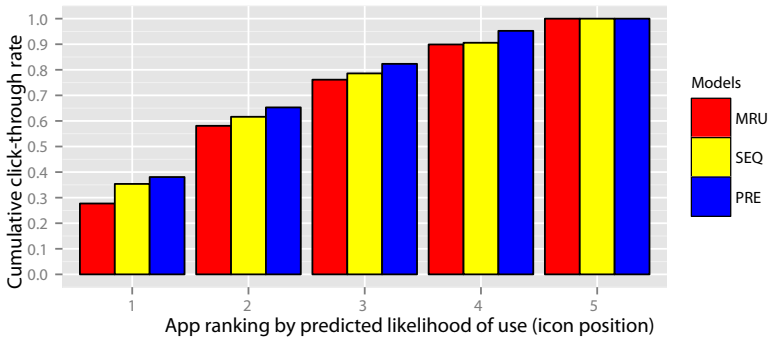


Figure 3.17: Cumulative click-through rate on icon positions on the adaptive menu for three different models.

Similarly, the PRE-based model outperforms other prediction algorithms at other ranked positions as well.

Our case study of the *AppKicker* application suggests that we were able to reveal differences in the models that we used for assigning applications to the shortcut menu, and that the PRE-based model worked best for supporting users with an adaptive launcher menu for applications.

Further, feedback that we got from users of the application through the Google Play Store (e.g., *"[I] was trying to find an app that allowed me to analyze which apps I used the most so I could save space on the home screen and this is the closest I've found"*, *"Helps to find apps faster"* or *"an interface that subtly blends into your home screen"*) suggests that the *AppKicker* application has met the design goal of providing a supportive functionality for users to launch their applications. Comments like *"I would like the ability to ignore apps"* and *"What about different sizes?"* give hints for future improvement of the application.

3.5 Summary

In this chapter, we investigated how people launch mobile applications on their smartphones and proposed an approach to support application launching on smartphones by extending current launcher menus.

First, as a means to study mobile application usage and to build other tools based on it, we conceptualized and implemented the *AppSensor*: a framework for tracing and analyzing application launches on smartphones in terms of usage counts and

usage durations. Next, we described a study based on a deployment of this framework in the large on the Google Play Store. For the first time — to the best of our knowledge — the method of deployment-based research by means of application store deployments was combined with fine-grained data collection on mobile application usage. In contrast to physical sensors (e.g. GPS for locations), we defined a virtual sensor for measuring the usage of mobile applications. The public deployment of *AppSensor* provided us with data from more than 4,100 users over a period longer than four months. Finally, we presented the design and implementation of an adaptive launcher menu that was built to help end-users start their mobile applications. In particular, this adaptive menu aims to shorten the search time for applications by adaptively presenting those applications to the user which he might start next. In short, this chapter included the following key contributions:

- The conceptual design of our research method for analyzing mobile application utilization in the large, namely the *AppSensor* as a virtual sensor for measuring mobile application launches.
- Results of a study in the wild showing that (among other findings) smartphone users spend almost an hour a day using applications but spend less than 72 seconds at a time with any given application (on average, per launch), and that average usage time differs extensively between application categories. A context-related analysis of our data led to the following conclusions (among other findings): (1) mobile phones are still used mostly for communication (text and voice); (2) some applications have somewhat intense spikes in relative usage (e.g. music and social applications), whereas others are more broadly employed throughout the day; (3) when people actively use their devices they spend less time with each application; (4) short sessions with only one application are much more frequent than longer sessions with two or more applications, and the first application within a session is very likely to be an application for communication; (5) when people are traveling they are more likely to use multimedia applications and they are surprisingly less likely to use travel applications.
- The design of an adaptive launcher menu for supporting launching of mobile applications, as well as an implementation of this proposal as an Android application, called *AppKicker*.

In the next chapter, we set out to understand how people customize their smartphone launcher menus, and we will cover further insights into how adaptive mobile launcher menus can be designed. In Chapter 5 we will make use of the *AppSensor*

described in this chapter to design and build a recommender system for mobile application usage. Finally, in Chapter 6 we will analyze data collected by the *AppSensor* to study the phenomenon of mobile application multitasking.

Chapter 4

Housekeeping Mobile Applications

As already discussed briefly in the previous chapter, mobile applications appear as a collection of icons on the user's device. In this chapter we will investigate how smartphone users interact with the icons of the applications that they have installed on their devices. Two studies will inform our understanding of how people organize all the applications they have installed, and a system for supporting this task will be proposed.

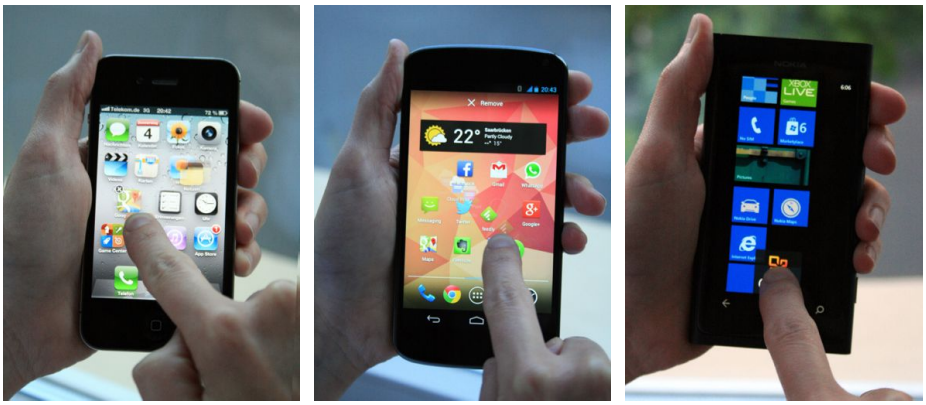
The results of this chapter have been presented in three publications [40, 31, 30]. Work related to this chapter can be found in Section 2.3.3 on launcher menus.

4.1 Introduction

By definition, housekeeping comprises the *"routine operations of a computer which make its work possible or more efficient, but do not directly constitute its performance"*.[1] In our case, the computer is a smartphone and the records that are being kept and maintained are mobile applications, which are installed on a user's smartphone.

Icons have been introduced with the classical WIMP interfaces, which consist of graphical user interfaces comprising windows, icons, menus and pointing de-

[1] OED Online: *"housekeeping, n."*, Oxford University Press, http://www.oed.com/view/Entry/88916, last accessed on 02.07.2013.



(a) On iPhone (b) On Android (c) On Windows Phone

Figure 4.1: Pictures of smartphone users arranging their icons in launcher menus on devices of three widely used mobile platforms. All platforms provide features for arranging icons based on drag-and-drop interaction, where users long press an icon to enable moving it.

vices [208]. Icons were introduced as graphical elements for representing *"objects, commands, and tools that were opened or activated when clicked on"* [208, p. 225]. While in Chapter 3 we addressed the aspect of activating an application by clicking on the corresponding icon, and we investigated how people utilize applications in terms of running them on their mobile devices and in terms of using provided functionality, in this chapter we will investigate how people organize applications on their mobile devices by means of arranging application icons. Figure 4.1 shows that icons can easily be arranged by drag-and-drop interaction in homescreen launchers. Understanding how housekeeping mobile applications is done on smartphones will allow us to help people get and maintain a better overview of their applications. Having an good overview of their applications and knowing where to find them is important for users to be able to effectively utilize the functionality that is available at their fingertips.

In both parts of this chapter we will investigate how people arrange icons in their smartphone launchers. In the next section we will provide a first study suggesting that context impacts people's icon arrangement, and present a system that exploits the relation of people's context to their icon arrangements. In Section 4.2 we will investigate people's habits that emerge over time, and present a system that exploits such patterns to support people as they arrange icons and curate their smartphone launchers.

4.2 Exploiting Icon Arrangements

The contextual relevance of mobile applications, i.e. how useful a particular application is in a certain context, can only be determined by the user herself. However, such a measure for an application's relevance is required as discussed in Chapter 3 for context-adaptive launcher menus and as we will discuss in Chapter 5 for context-aware recommendation. While in Chapter 3 we investigated the relation of an application's launches to its contextual relevance, in this section we propose an approach for a assessing this metric by looking into how people arrange icons within a mobile application menu.

This section gives rise to exploiting the user-defined icon arrangement as an implicit feedback for the relevance of services when the arrangement occurs in relation to user's context. A system correlating a user's context and with his icon arrangement will also allow developers and designers to learn about how the user assesses the application's contextual relevance, before the application itself is actually available and any other data can be traced.

4.2.1 Preliminary Study

To understand how people arrange icons on mobile devices and to evaluate the impact of their contexts, we conducted a study where we collected data from people in the wild, i.e. we observed our participants in their natural contexts how they arrange icons on a mobile device. We have provided our participants with instrumented smartphones, since we were rather interested in a very special aspect of smartphone usage rather than observing their natural smartphone usage. We have chosen the following four different contexts for our study, with participants being in different activities:

(1) At the weekly farmer's market, i.e. people buying groceries;
(2) during shopping in a mall, i.e. people carrying bags and looking for consumer goods;
(3) at the airport, i.e. people carrying baggage and going to travel by air;
(4) and at the university cafeteria, i.e. students who are out for lunch.

As interviewers we were able to ascertain that our participants were in the specified contexts by looking at the criteria mentioned.

(a) Screenshot of application (b) Positions in grid layout

Figure 4.2: Screenshot of the mockup application for the icon arrangements and positions in the grid layout. In (a) the translations of the icons' labels are, from top left to bottom right: *Last Minute*, *Hot Deals*, *Buddy Finder*, *Departure*, *Lectures*, *Cafés*, *Fruits*, *Price Comparison*, *Cheese*, *Vegetables*, *Parking Spaces*, *Arrival*, *Menu*, *Events*, *Organic Foods*, and *StudiVZ* is a German social networking platform for students.

A mockup application, shown in Figure 4.2, simulated an icon-based menu on a mobile phone. We asked the participants to arrange the icons on the menu so that the menu would fit to their current activity. They were able to use drag and drop for moving icons. The initial arrangement was randomized and our participants were totally free to spatially arrange the icons.

We designed dummy applications with icons showing 4 relevant and 12 irrelevant services for every context. In every context 4 icons symbolized a worthwhile value to the users, like for instance applications for price comparison or hot deals for the shopping context. We designed these icons and their labels such in a way that the value to the user in his context was obvious, e.g. information services on fish for the market, on discounts for shopping, on the flight schedule for the airport, and on the cafeteria menu (see Figure 4.2 for a full list).

Findings of Study

The study took place in Münster, Germany, in 2009. We interviewed 100 people, 25 for each context. There were 49% male and 51% female participants with an

Figure 4.3: Average positions of icons in different contexts, numbered from 16 (upper left) to 1 (lower right). Colors highlight the price comparison application (blue), the food menu application (green), the fruits (light orange) and vegetables (dark orange) applications, and the departure schedule application (red).

average age of 34 years. On average, they made 7.7 icon moves (min 2, max 16), which took 28.3 sec (from 6 to 85) on average.

Our data show that in different contexts our participants moved different icons to specific positions on the menu. Figure 4.3 shows this relation: Each row represents a distinct context, and each field within a row stands for an icon. The order of the icons within the rows is the same. The positions are numbered from 16 (upper left position of the menu) to 1 (lower right), as shown in Figure 4.2(b). Each field in Figure 4.3 has a certain width, depending on the average position of the icon among the participants separated by context — i.e. the wider the field the further up it was placed.

A comparison of the width of specific fields shows that the icons have been moved to specific positions in different contexts. For instance, on average people moved the application for price comparison to position 14.9 in the shopping context and to position 12.6 in the market context, whereas they only moved it to position 6 in the cafeteria and to position 6.3 at the airport. People moved the icon of the menu application to position 15.4 in the cafeteria context, and to positions below 7 in the other contexts. Figure 4.3 shows that other icons show similar relations.

4.2.2 System for Exploiting Icon Arrangement

The findings of our study informed the design of a prototype within a platform where end-endusers are able to develop their own mobile applications, called *procomo* (see [35, 34] for details of platform). It is a client-server architecture based on web technology which pushes contextually relevant applications to a client run-

(a) (b)

Figure 4.4: Prototype of the context-aware icon-based main menu (a), and a user rearranging the icons and making use of the trash bin (b).

time environment running on smartphones. These user-generated applications are basically small HTML/JavaScript-based widgets.

The application server hosts the mobile services which are authored by users. It also infers the contexts the mobile users are currently in by using a sensor data clustering approach [98]. The mobile client continuously sends context information to the server (i.e. location and time as well as estimated values, e.g. distances from home/workplace, and speed). In response it retrieves a list of applications ordered by the estimated contextual relevance; these are presented in the main menu as shown in figure 4.4.

The server also logs every icon arrangement a user makes. Therefore, snapshots of the menus are saved to a central database. The relevance of a service can be deduced as discussed above from the position of an application within the menu. Following the approach of context-collaborative filtering presented in [29], the prototype aggregates the relevances of services based on all user-given icon arrangements for each context. On context change, e.g. when moving to a different location, the menu gets flushed at once and new applications are loaded, which results in new icons appearing in the menu.

4.2.3 Discussion of Findings

Our findings support the approach of exploiting icon arrangement as an information source for context-awareness, as we propose in this section. The study suggests that context has an impact on the users' icon arrangements on mobile devices.

The study also suggests that distinct patterns for arranging icons might exist, since we have found one particular pattern in this study (i.e. relevant services on top, less relevant services on bottom). We may assume that due to cultural [144] and individual [109] properties of users and also particular designs of launchers on smartphones also other patterns exist.

To summarize, in this section we investigated how the current context of a user impacts his icon arrangement within a smartphone menu. We presented a first approach and a simple system for assessing icon arrangement of users by looking into how they visually layout their icons. Our findings suggest that context has an impact on how users arrange their menus: During different activities, they prefer different icons to be placed at specific positions. In the next section, we set out to gain a deeper understanding on which concepts people use for arranging their icons, and we will investigate if and how this phenomenon also occurs on user's personal smartphones, which they can customize over longer time periods.

4.3 Habits of Icon Arrangement

As discussed earlier in Chapter 1, the number of available mobile applications is steadily increasing. People have rapidly adopted application stores as a means to customize their smartphones with various functionalities that go beyond communication. Understanding the principles of their mobile application usage is crucial for supporting users within this new ecosystem. In this section, we investigate which concepts people apply when arranging application icons on their smartphones, and how their habit impacts the visual and hierarchical layout of launcher menus.

We designed a study following the idea of natural experiments (as described in Section 2.2). We asked more than 130 participants about their habits in icon arrangement and collected more than 1,400 screenshots of their devices' menus to further ground our findings based on two major platforms (iPhone and Android). Based on this data, we can distinguish five different concepts for arranging icons on smartphone menus, e.g. based on application usage frequency and applications'

functional relatedness. Additionally, we investigated how these concepts emerge in relation to frequency of application installations, removals and icon rearrangements, as well as users' experience levels.

Once installed, a new application resides on the user's device and is available for instant usage. One kind of menu for launching applications became standard: Icon-based menus that are arranged in a grid layout [77, 126], as shown in Figure 4.6, became common. These menus help people to organize, find and use their applications. However, since the screen size of mobile devices is limited, at some point the user has to decide on how to organize the icons, which we refer to as housekeeping in this chapter. Current smartphones may be able to show up to about 24 icons at once. Icons that do not fit on the screen can either be put on a new page to be reached by scrolling, or they can be organized hierarchically into folders to be reached by navigating. While there are intuitions and beliefs on how people manage their applications (also from Section 4.2), there is little published research on the topic, except results published from this thesis [31, 40, 30]. As a result, so far we have not been able to comprehensively support the process of housekeeping mobile applications. Important questions remain unanswered, for instance: Do people have certain concepts for arranging icons? If so, what are these concepts and how are they applied? How can we exploit the effort people put into maintaining their launchers? Do arrangements made by more-experienced users differ from those of the less-experienced? Or do people remove applications to solve the problem of limited space?

One major design goal for menus is to adapt them to the users' tasks [164]. This is of particular interest for mobile menus, since the tasks of mobile users [18] and the applications they use [38, 86] are perpetually changing, and the design of context-aware menus is a topic of current research (cf. [31, 126, 138, 229, 253]). However, in contrast to pre-designed menus, smartphone launchers are highly customized by the very users themselves. Customization itself has become a primary activity [109, 161], e.g. to make the device more efficient or to manage complexity. Yet it is unknown if the design goal of task-relatedness also emerges when users arrange their mobile menus themselves. This is the focus of this section.

4.3.1 Study Method and Setup

The study we describe in this section was inspired by a screenshot-based diary study on mobile task interruption [142] and the work on personalization by Tossel et al. [246]. We have adopted the method of a screenshot-based study for two reasons: Firstly, in contrast to the study we described in Section 4.2 where subjects were asked to arrange icons ad-hoc within a launcher menu mock-up, we did not want to bias our sample by the arrangement task itself at this point. Further, since the customization of functional phone settings happens over the long term [109] and usability is a long-term experience [153], the chosen design allows us to collect data that has evolved naturally. Secondly, we decided against using a logging application as proposed for mobile in-the-wild studies [117] as like done in Chapters 3, 5, and 6, since by introducing a dedicated application with its own icon, we would have biased what we wanted to observe, i.e. our participants would have had another icon in their launcher menus that would have been an artifact of our study. As a result, we have chosen to investigate iPhone and Android devices, since at time of our studies these were the only widespread representatives of the current generation of smartphones (allowing users to install applications and arrange icons) with the capability to easily take screenshots. This was not possible on most Windows Phones at the time the study was conducted. As a result, we thereby were able to collect data in the wild following the approach of quasi-experimental design without imposing too much effort on our subjects.

Our study had two steps: First, we asked volunteers to make screenshots of their menus for the purpose of analyzing their icon arrangements, and to send them to us by email. Secondly, we sent a short questionnaire to all participants. In four groups of questions, we asked about their device customization habits, general phone usage, personal info, and general comments. We set up a website with instructions, and recruited subjects by email invitation, Facebook and Twitter. Data collection was done during June and July 2011 for the iPhone, and during July and August 2012 for Android (when screenshots became possible on Android 4.0).

We asked people to send us screenshots of their customized launchers. As such, our samples might be biased in that we did not receive data from people who do not customize their menus at all. However, our goal was to investigate how people customize their menus, not whether they do so at all. The latter can be concluded from related work (cf. [28, 109, 161]).

Referring to the research approach laid out in Chapter 2 the study we present in this section is not a study through the application store, because we did not collect data through an application deployed to an application (store for the reasons stated above). However, the data is collected in the wild. This study can best be understood as an quasi-experiment, since every participant selects the condition that we want to study on his own; i.e., the participant decides which concept he wants to use for arranging his icons,

4.3.2 Results of Screenshot Study

In total we received data from 132 people: 1,486 screenshots from 106 iPhone users (with iPhone version 4 or earlier versions), and 144 screenshots from 26 Android users. Since taking screenshots on Android is only built into the latest version of the OS (Android 4.0) we had to rely on a smaller user base. 22 participants were female and 108 were male (2 participants did not disclose their ages). Their mean age was 28.32 years (SD 8.48). We reached participants from various countries: 60.5% from Germany, 11.4% from the United States, 4.5% from the United Kingdom, and the rest from 20 other countries.

We asked our participants to categorize their smartphone experience on a 4-point scale between *novice users* (level 1) and *expert users* (level 4). The mean level of experience of our participants is 3.46 (SD 0.71). Therefore, we clustered our participants into 58 less-experienced (those with levels 1, 2 and 3) and 74 more-experienced users (those with level 4).

Practices of Installing, Arranging, and Removing

We asked participants on a 5-point scale (0 times, 1-10 times, 11-20 times, 21-30 times, >30 times) how often they have installed applications, rearranged their icons, or uninstalled applications in the last month. The median is 1-10 times for all, i.e. in the last month our participants have on average installed 1-10 applications, rearranged their icons 1-10 times, and uninstalled 1-10 applications. We designed the scales based on anecdotal reports from an informal pre-study and to capture a wide range of frequencies for installing, removing and arranging applications. In fact, the answers on the three questions were distributed over the full range of the scale, although the medians were at the 1-10 option.

We then set out to find correlations between the variables we collected. We found that the more often people install applications, the more often they also uninstall applications (*Spearman's* ρ 0.79, $p < 0.001$). This suggests that people either try new applications — i.e., install them and remove them if they are not worth keeping — or that they remove older applications that they do not need any more when they install new ones. By removing applications when installing new ones, they either replace the functionality of the removed application with the new application, or they simply create free space for the new application. Further, we found that the more often people install applications, the more often they also arrange their icons (*Spearman's* ρ 0.68, $p < 0.001$). This suggests that people sort their icons when they have installed a new application, so the act of arranging icons is often triggered by a new application being installed.

Common Concepts for Arranging Icons

Beyond these basic statistics on our participants' menu structures and arrangement practices, we looked into our participants' concepts for arranging icons, and based on that our experts inductively labeled the data. In this section we extract people's concepts for arranging icons and investigate how the concepts found impact the structures of their launchers.

We asked our participants to describe the concepts they use to arrange their icons, if any. We chose a free text field over a predefined set of answers, since we wanted to explore existing concepts instead of providing pre-defined categories. Based on the participants' descriptions, we deductively extracted concepts for arranging icons following a grounded theory approach using open coding to analyze the qualitative data [207]. We found the following five concepts that our participants applied for arranging their icons in their launcher menus:

- *Usage-based icon arrangement*: People who apply the concept of usage-based arrangement order their icons by a specific criterion that quantifies an inherent attribute of a single application. In most cases, we found the frequency of using an application to determine this value. Many of our participants move frequently used applications to the first page of their devices. Some also said that they would move least-used applications to the last page of their menu — it is worth mentioning that from sorting of the first pages a sorting of the last pages does not follow implicitly. Additionally, some people used terms like importance or relevance to name the criteria that they

used to order their icons. The latter somehow relate to frequency, but are not necessarily associated with each other. We put these two concepts together since on the one hand they are indistinguishable from the wording that people use to describe their concepts, and on the other hand they both relate to an attribute that is inherent in the application of an icon.

- *Relatedness-based icon arrangement:* Participants who follow this concept cluster applications based on their functionality, i.e. applications that are related to each other are put into one folder or onto one page, e.g. the two social networking applications *Facebook* and *Twitter*. The similarity of two applications is judged based on people's subjective assessment. For instance, Twitter might also be clustered together with mail clients, when clustering communication applications. In contrast to the usage-based concept, this concept takes two or more icons into account when it comes to arranging the icons.
- *Usability-based icon arrangement:* A third concept that we found among our participants is the idea of organizing applications such that the usability of their device is optimized. For instance, one argument was to be able to easily reach icons with one's thumb, since the performance of thumb-interaction depends on icon position [184], or to have space to swipe through the screens without accidentally clicking on icons. Obviously the previous two concepts of usage-based and relatedness-based icon arrangement also contribute to usability, but people in this category have explicitly conceptualized and named usability aspects for arranging icons.
- *Aesthetic-based icon arrangement:* Participants who follow this concept have a tendency to arrange their icons in a way that is aesthetically pleasing to them. For instance, one user without icons on the first page wants to be able to see the background image showing her friends on the first page; other participants cluster icons by their color, e.g. a checkered pattern of brown and blue icons.
- *External concepts for icon arrangement:* We identified a fifth group of people who use external concepts to arrange their icons. These participants use sorting patterns that have evolved externally from their smartphones and apply them to their icon arrangement. For instance, people using this concept keep the arrangement that was pre-configured on the device. Others have stated that they keep their applications in the order of installation (default arrangement). One user said he would arrange his icons alphabetically.

(a) Usage-based (b) Relatedness-based (c) Usability-based (d) Aesthetic-based

Figure 4.5: Example screenshots of participants who used certain concepts for arranging their icons: (a) one participant who reports to *"put the most frequently used applications on the first screen"*; (b) a user with five folders on his first page who tries *"to group [applications] by what they do or what I use them for"*; (c) a participant who says he would *"keep third row available for easy swiping to the next page"*; (d) a participant who has created a checkerboard pattern: *"most icons are blue, so on my first page of icons it alternates between blue and brown and I try to keep that consistency throughout"*.

> Some people also explicitly stated that they have no concept for arranging their icons. Yet, since every icon arrangement has an inherent order, it is unclear how this order emerged. It is most likely that people who do not have any explicit concept also follow an external concept, e.g. just leave the arrangement as it was preinstalled or add the icons of new installed applications to the first free spot in the menu.

Hybrid Concepts and Co-Occurrence. It is worth mentioning that these five concepts are not mutually exclusive, i.e. a user may apply two or more concepts in parallel. For further analysis, all participants have been categorized based on the five concepts we found. To reduce the subjectiveness of the categorization, the labeling has been done by three different analysts whose results have been merged by the principle of majority rule. For this reason we can take their merged classification as ground truth. We have been able to partially cross-validate people's textual description given in their self-reports with the screenshots they provided: For people who said that they group by similarity, we found folders of applications, and those who claimed to exploit icons' colors have also been proven to be right; for instance the participant who said that "most icons are blue, so on my first page of icons it alternates between blue and brown and I try to keep that consistency throughout" is shown in Figure 4.5(d) — while the color blue is recognizable, the color brown is rather fuzzy. We had to trust participants' self-reporting feedback

	(1)	**(2)**	**(3)**	**(4)**	**(5)**
usage-based (1)	*79 [62%]*	35 [26%]	8 [6%]	3 [2%]	5 [4%]
relatedness-based (2)	35 [38%]	*76 [60%]*	7 [6%]	4 [4%]	4 [4%]
usability-based (3)	8 [6%]	7 [6%]	*11 [9%]*	2 [2%]	0 [0%]
aesthetic-based (4)	3 [2%]	4 [3%]	2 [2%]	*6 [5%]*	0 [0%]
external concepts (5)	5 [4%]	4 [3%]	0 [0%]	0 [0%]	*12 [9%]*

Table 4.1: Co-occurrences of different concepts for arranging icons. Fields show absolute number of participants (and relative numbers in parenthesis). The diagonal shows how often every single concept appears in our data.

on the usage-based concept, since we did not collect any statistics on application usage. For technical reasons we were not able to apply the *AppSensor* described in Section 3 for this study.

We also looked into the co-occurrences of the five concepts. On its diagonal, Table 4.1 shows how often the concepts that emerged appear within our sample. Only 10 participants did not give any answer as to how they organize their menus. Two of the ten participants using external concepts explicitly stated that they do not use any concept. As an interesting fact, these two participants graded their own iPhone experience as less-experienced (level-1 and level-2).

The most commonly used concepts for icon arrangement are relatedness-based (76 participants) and usage-based (79 participants). Table 4.1 shows the pairwise number of concepts' co-occurrences; the values on the diagonal show the number of single appearances. The most often applied tuple of concepts is the combination of the usage-based concept with the relatedness-based concept, which is used by 35 participants. The usability-based, aesthetic-based and external concepts appear less frequently together with the two other major concepts. Nonetheless, we tested for significant correlations but did not find any systematic couplings between the concepts.

These concepts that we describe in this work emerged both from iPhone and Android users, and all concepts appeared on both platforms. We did not find any concepts that appeared on only one of the platforms.

Specific Reasons for Arranging Icons

In addition to the aforementioned common concepts for arranging icons, we also found more specific and subtle reasons for customizing launchers.

Besides the first page, which is most commonly used for applications that are used frequently, some participants also mentioned that they use the last pages of their menus for applications they do not use often, *"silly apps"*, or applications *"that are never used but might come in handy some day"*. One user refers to his last page as the *"land of misfit apps"*, and explains that he puts applications there which do not fit into his sorting schema, which is usage- and relatedness-based. Interestingly, only one user reported that he consequently removes applications that he did not use for a month. Another user who follows the usage-based concept reports that he intentionally also puts applications on the first page if he wants to use them more often, e.g. an application for taking notes.

Further, for some people having as few pages as possible also seems to be a goal of arranging icons. One participant reported that he does so in order to have less pages to browse.

We also got comments from our participants suggesting that context of use plays a role when people arrange their applications. One user reported that he has a folder for applications to give them a try, when he has "a few minutes free". Further, the general purpose of the device also affects the arrangement. One participant reported that she tries "to put games in the back and work apps in front, because it's a work iPhone". We will study the impact of context on icon arrangement in Section 4.2.

Interestingly, one participant told us that he starts to arrange applications into folders when he loses track of which applications are installed. Only one participant reports that he makes use of the search functionality provided by the iPhone to search for applications. It is known that people prefer visual search over search by names, for instance, since the name of the item would have to be remembered [19], and further, searching for items for retrieval is cognitively more demanding than navigation [26].

In addition to the usability-related aspects we already have mentioned, one user explicitly explained that he tries to keep icons of certain applications "at the same position". Another user purposely keeps icons that look similar at different positions, to be able to distinguish them more easily at a quick glance.

(a) Page of an iPhone launcher (b) Folder of an iPhone launcher

Figure 4.6: Screenshots of iPhone launcher showing a) a page with icons of nine applications and four folders, and b) a folder as a submenu.

iPhone-specific Results

The iPhone's launcher has some constraints that limit the way people are able to arrange their applications' icons. Users are able to distribute icons over pages and cluster them into folders, as Figure 4.6 shows. They can swipe through the pages, and folders are represented by special icons, which can be opened by clicking them. Theoretically, people can have as many applications as they want and put them onto as many pages as they like. In our study, on one page they could have up to 20 icons (no participants was using an iPhone version 5 at that time, which has 24 icons), which can be arranged within a grid of four columns by five rows. The fifth row has a special function: its icons appear on every page as a quick start bar. In the first four rows above, the icons are arranged in a text-like flow from upper left to bottom right, i.e., users can only fill up rows icon by icon, without leaving any gaps. The hierarchy of the menu is limited to two levels: On the first level, people can have icons for applications and icons for folders, and on the second level people can put up to 12 icons of applications into folders.

iPhone Data Characteristics. Among the iPhone participants, there were some who customized their devices by *jailbreaking*[2]. Jailbreaking enables the users to install applications not available from the official Apple AppStore and to adapt

[2]Wikipedia: *iOS jailbreaking*, http://goo.gl/37msu, last accessed on 05.07.2013.

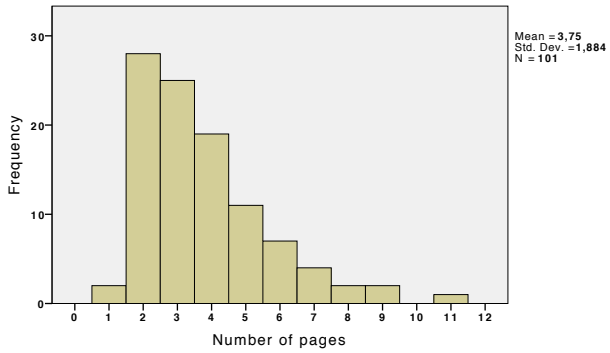


Figure 4.7: Frequency of number of pages in participants' launchers.

their devices to personal requirements and preferences. Four of them used options to customize their menus beyond the available standard options (e.g. more than four icons in a row, more than 12 icons in a folder, not filling up row by row). Since their menus are highly customized, they are not comparable to the other standard devices' menus. Therefore, and since these users most likely also had unusual high technical ability, we removed these four records from our data. We also removed one participant who submitted screenshots of his iPod Touch, since this is not a communication device in the first place and therefore not comparable to smartphone customization. Interestingly, this device had many more screens (122) than the other participants' iPhones.

As such, our cleaned iPhone data set is based on 101 participants, 1,166 screenshots (of 379 pages and 787 folders), and 3,415 unique applications shown as 9,649 icons. An average participant has a mean of 95.53 applications installed (min 22, max 278, SD 53.62), distributed his icons over 3.75 pages (min 1, max 11, SD 1.88), and created 7.79 folders for additional organization (min 0, max 37, SD 7.31). Figure 4.7 shows the distribution of our participants' number of pages. Most people have two pages in their launchers; two participants have only one page. The top applications that are installed on every device are the pre-installed iPhone applications, e.g. *Phone*, *Contacts*, *Notes*, *Compass*, *Mail*, or *Calendar* (since they cannot be removed from the device). On average, any one application was installed by 2.83 subjects (min 1, max 101, SD 8.797).

Impact of Concept on Icon Arrangement. Based on our categorization, we investigated whether the concepts have any impact on the user-defined menu structures. In this section, we analyze the data inferred from the screenshots to quantitatively ground the concepts that emerged.

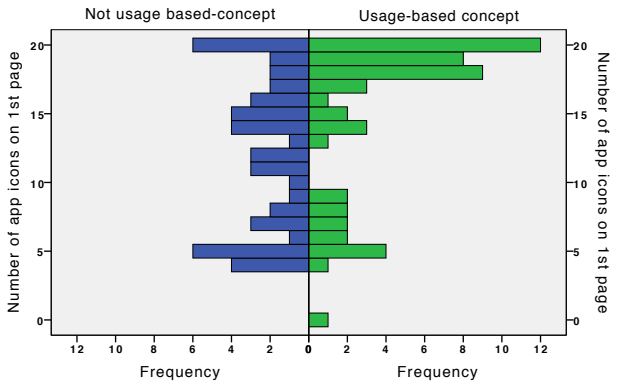


Figure 4.8: Histogram of number of application icons on first page grouped by usage-based concept. Left side (blue) shows distribution of participants using the usage-based concept; right side (green) shows distribution of participants not using the usage-based concept. Note that the x-axis is positive on both sides.

We found that the number of applications people have on their first menu page significantly differs between participants who apply the usage-based concept and those who do not (*t-test, t=2.475, p<0.05*). Figure 4.8 shows a histogram of the number of applications on the first page for both categories of users. The graph shows that people who arrange their applications by usage tend to have more applications on the first page.

The number of folder icons on the first page significantly differs between participants who apply the relatedness-based concept and those who do not (*t-test, t=2.198, p<0.05*). Figure 4.9 shows a histogram of the number of folder icons on the first page segmented by usage of the relatedness-based concept. It appears that people who apply the relatedness-based concept are more likely to have folders on the first page of their menus. This suggests that such participants also use the concept of similarity to cluster their most important applications.

Further, the distribution of the number of re-arrangements significantly differs between subjects who do apply the relatedness-based concept for arranging their icons and those who do not ($\chi^2 = 6.634$, $p < 0.05$). Figure 4.10 shows that people who keep their applications clustered by similarity do rearrange their icons more often. This suggests that these participants actively make use of the customization function to keep their applications in an arrangement that fits their own preferences.

Finally, we found a significant difference (*t-test, t=2.766, p<0.01*) in the average number of applications people put into a folder between participants who apply external concepts (mean 5.2) and those who do not (mean 6.8). It is likely that

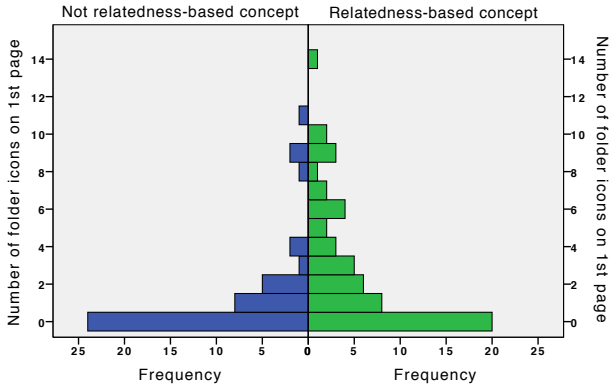


Figure 4.9: Histogram of number of folder icons on first page grouped by relatedness-based concept. Left side (blue) shows distribution of participants using the relatedness-based concept; right side (green) shows distribution of participants not using the relatedness-based concept. Note that the x-axis is positive on both sides.

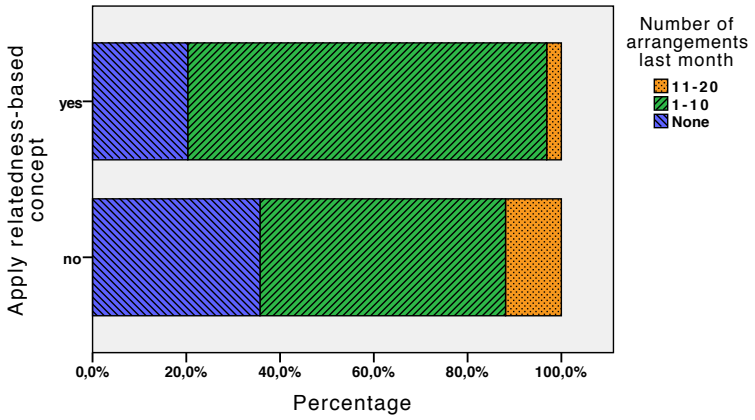


Figure 4.10: Effect of the relatedness-based concept on frequency of re-arrangements.

the external concepts people apply (e.g. the alphabet or order of installation) provide an order in only one dimension. Thus, people who apply an external concept are less likely to sort applications into folders and less inclined to hierarchically arrange them into a second dimension.

Grouping of Apps into Folders. Further, we looked into how people cluster applications into folders. Participants applying the relatedness-based concept reported that they use folders to group applications with related functionality. However, participants who did not explicitly state that they used this concept have also created folders and grouped applications. Therefore we did not distinguish between concepts when investigating folder arrangements.

A maximal co-occurrence can be found among those applications that are pre-installed on the iPhone. This is not surprising since these applications are installed on every device. For instance, the *Voice Memos* application appeared 74 times together in a folder with the *Compass* application, and 74 times together with the *Calculator* application. Next, *Compass* and *Calculator* appear together 64 times, *Voice Memos* and *Stocks* 60 times. Some of our participants have also reported explicitly that they cluster the original *iPhone* applications together.

Further, we looked into applications that people have installed from the *AppStore*. For instance, *Instagram*, which is an application for social photo sharing, was installed by 29 participants. Most often it appears together with *PS Express* (10 times), an application for photo editing, and *Photosynth* (10 times), which is an application for browsing large photo collections and creating panorama images, and the default *Photos* application for browsing pictures (8 times). Basically, these three applications provide follow-up actions after taking pictures. Additionally, *Instagram* also co-occurs with other applications for taking pictures, i.e. applications that basically provide the same functionality as *Instagram*. These applications are the default *Camera* application (8 times), which provides basic functionality for taking pictures, and *Hipstamatic* (8 times), which is a camera application that provides additional effects.

Next, we looked into what kind of applications people group together with *Facebook*, an application for taking the social network mobile that was installed by 82 participants. It appears that *Facebook* is most often clustered with *Twitter* (28 times), which is another application in the category of *Social Networks*. Additionally, other social network applications like *FourSquare* (18 times), *LinkedIn* (14 times), and *XING* (10 times) appear frequently together with *Facebook*. The second most frequent application appearing together with *Facebook* after *Twitter* is *Skype* (24 times), which is also listed under the *Social Networks* category on the Apple AppStore, but the main purpose of this application is communication.

For the *Games* category we investigated which other applications people cluster together with *Angry Birds*. Different editions of *Angry Birds* were installed by 34 of our participants. On their smartphones, it appears together with other applications of the *Games* category like *Cut the Rope* (18 times), *Fruit Ninja* (16 times), *Tiny Wings* (12 times) and *Doodle Jump* (12 times). Additionally, the iPhone's *Game Center*, which is a social gaming platform, appears quite frequently together with *Angry Birds* (16 times).

We also looked into applications for shopping. It appears that the *eBay* application, which is in the category *Lifestyle*, co-occurs most often with a German Craigslist-like application (16 times). Secondly, it also appears 12 times together with *PayPal*, which is an application for mobile money transfers and is in the category *Finance*, and 12 times together with *Amazon*, which is an additional marketplace which is listed under *Lifestyle*.

These examples provide evidence that people cluster related applications into folders. Based on our quantitative screen analysis, we can identify two additional reasons for putting applications together based on their relatedness: On the one hand, people put applications with similar functionality into folders. When they navigate to a folder and open it by clicking the folder icon in a first step, e.g. games, in the second step they can then decide which game to play. For instance, one of our participants has a folder on his first page containing two applications: the default short messaging application and *WhatsApp*, which is an alternative messenger that transmits text via data networks. On the other hand, people group together applications that belong to a certain workflow, e.g. photo editing together with camera applications, and payment applications together with shopping applications. Menus arranged according to these two approaches — functionally and thematically clustering — have also been found to be differently favored by cultures according to Kim and Lee [144]. We have found that these two approaches also emerge when users organize menus themselves, yet we have not been able to show any significant cultural differences.

Figure 4.11 shows the number of less-experienced and more-experienced users according to whether they rearranged their icons within the last month or not. It appears that more-experienced users rearrange their icons more often than less-experienced users. We further found that the more-experienced users make more use of folders in terms of filling them with icons. The mean number of applications a more-experienced iPhone user puts into one folder significantly differs from the number of applications a less-experienced user puts into a folder on average (*t-test, t=3.31, p<0.001*). More-experienced users fill their folders with more applications: On average less-experienced participants put 6.0 applications into one folder, while more-experienced participants put 7.1 applications into one folder. Also, from stationary computers we know that more skilled people apply more elaborate arrangement concepts more consciously [198].

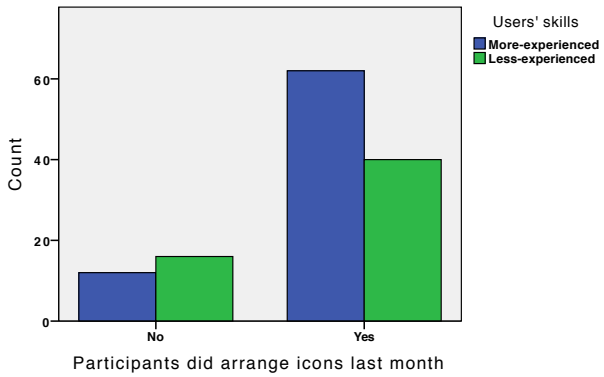


Figure 4.11: Rearrangements of icons within the last month, comparing less-experienced and more-experiences users.

Android-specific Results

Constraints of Android Devices. Android differs from iPhone and gives users more freedom in customizing launcher menus. People can change the launcher application as such an install launchers with different features, e.g. visual appearance. Further, people can place not only icons but also widgets on their screens. Widgets provide small self-contained UIs for self-updating data, e.g. on weather, news, stock markets, or social network streams. Furthermore, Android has a dedicated application menu — called the application drawer — that contains the icons of all applications that are installed, and from there people can drag-and-drop them to their screens. People can also place more than one instance of an application icon on their screens. Most interestingly, on Android people can freely place icons everywhere in the menu grid, while on the iPhone they can only start in the upper left corner and fill screens up to the bottom right. Android also provides a quick-start bar for icons of applications and folders that appears at the bottom of every page.

Android Data Characteristics. Our Android data set is based on 26 participants, 144 screenshots (of 115 pages and 29 folders), and 493 icons. On average our Android users had 4.32 pages (note that the number of pages is preconfigured on most Android devices and pages may be left empty), and 18.16 application icons on their pages. They used an average of 5.16 widgets, occupying 26.88 icon positions on average per user, i.e., our Android participants used more screen space for widgets than for application icons. Due to sparsity of our Android dataset we did not investigate people's grouping of applications into folders or analysis of single ap-

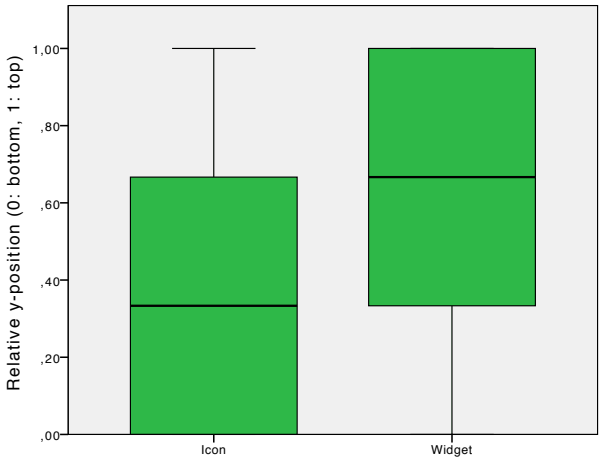
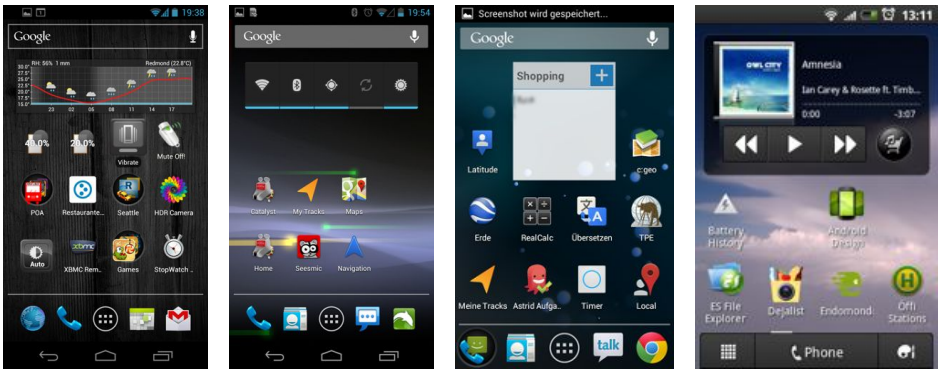


Figure 4.12: Relative y-position of widgets and icons on people's screens.

plications. However, people also do make use of folders on Android for arranging applications.

Investigating Free Icon Positioning. For Android we investigated where on the screen people place their widgets and icons, and analyzed which screen positions of the grid are filled with either icons or widgets. The median of the relative y-position of icons is 0.33, and the median of the relative y-position of widgets is 0.66 (with 1 being top and 0 being bottom edge of the grid). A Mann-Whitney U test revealed that there is a significant difference in the y-position between icons and widgets (*U=496, Z=-9.089, p<0.001*), with the former being placed more in the upper part of the screen, and the latter in the lower part of the screen, as Figure 4.12 shows. Since most widgets are not built for application launching but rather for mere data presentation or settings (e.g. do not have buttons to click on), one explanation is that at the lower part of the screen people can reach their application icons more easily to start applications when using their thumbs [184]. 6 participants left the clock and weather widgets at the upper screen positions, where they are usually predefined by device manufacturers. Figure 4.13 shows some examples of participants and where they have placed their widgets.

We also analyzed the horizontal placement: Icons were placed equally on both sides (median 0.5, with 0 being left side), and widgets have a tendency to be placed more on the left (median 0.33), though there is no significant difference.



(a) Weather (b) Control settings (c) Shopping list (d) Music player

Figure 4.13: Example screenshots of Android users who put their widgets to the top of the screen and some icons of their applications and folders to the lower part.

4.3.3 Implications for the Design of Launcher Menus

The results from our study of people's habits in smartphone launcher customization allow us draw some conclusion for an improved design of smartphone launcher menus and housekeeping of mobile applications, e.g. to support less-experienced users or arrangement in general, and to exploit the user-defined menu structure.

Support for Less-experienced Users

We found that less-experienced users install applications as often as more-experienced users, but do not arrange them equally often. Additionally, it appears that they have not yet developed a concept to arrange their icons. They might feel lost and lose track of the applications on their devices more easily. Therefore, we suggest supporting less-experienced users with functionality for better housekeeping of applications. This support for organization of applications can be stopped as soon as the users show an increase in application removal and arrangement on their own, or after some duration of device usage. Convenient patterns of application arrangement can be adopted from more-experienced users, e.g. clustering by application functionality and type (for instance following the categories of applications on the market), or by placing frequently used applications at the front. Fukazawa et al. [101] report that functions for automatic menu customization are most appropriate for novices users.

Although Ziefle and Bay [270] found significant differences between old and young people concerning the mental models that they build of their smartphone

menus, we did not find any significant effect of age on any of the variables we measured. Further, we did not find any significant differences concerning genders or countries.

Supporting Icon Arrangement

Five participants reported that arranging icons can be annoying and time consuming. One explained that it would be too time consuming to move an icon from the last page to the first page, and therefore reported leaving icons at random places occasionally. As a solution, one participant explained that she arranges her icons on her stationary computer and then synchronizes the layout of icons to her mobile.[3] Another participant reported that on his iPad he would put more effort into arranging icons. This suggest that it is easier to arrange icons on bigger screens. Since we found that the majority of people do arrange icons, we can assume that people do benefit from their arrangements. It is very unlikely that they invest the effort of arranging icons if they do not benefit from doing so. Thus, this suggests that icon arrangement on smartphones can be improved by supporting the user.

We found that the frequency of rearrangements relates to the frequency of installations. This suggests that new applications are sorted into the existing schema. Therefore one way to help people to keep their applications arranged is to provide assistance when installing new applications. The icon of the new application could be placed next to icons of those applications next to which other people have placed it, instead of just adding it to the first free spot in the menu. According to our data, somebody who installs *Hipstamatic* could be advised to place the icon into the folder where the icons for *Instagram* and the *Camera* already reside. Additionally, a device might advise its user to put icons of frequently used applications on the front page, since this is a common concept. As already proposed by Findlater and McGrenere [94], a mixed initiative menu where the user and an automatic system iterate on the design of the optimal menu might work best.

User-built Meta-applications

We found that people cluster complementary applications for workflows, e.g. photo taking, photo editing, and photo sharing (see Figure 4.14). Compared to stationary computers, where software usually provides richer and more comprehensive func-

[3] Arranging icons and synchronizing them to the iPhone is supported by *iTunes* (desktop software for managing *iPhone* content).

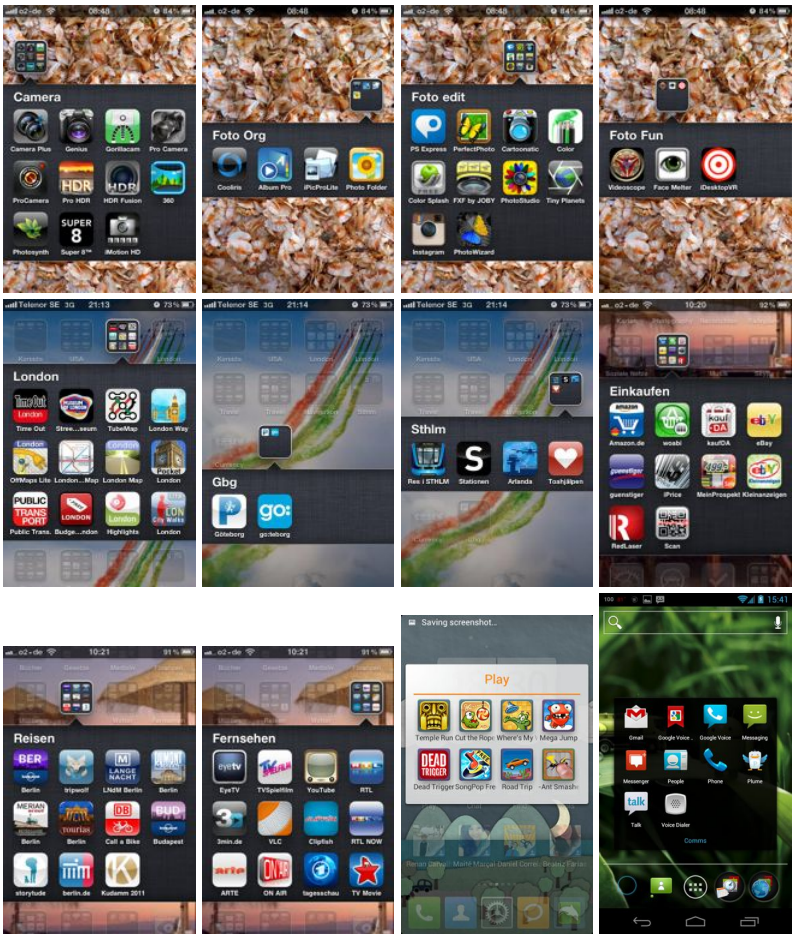


Figure 4.14: Application clusters of different users: photo-related (1-4), city-related (5-7), activity-related (8-10), games (11), communication (12).

tionality, mobile applications have a more specialized and self-contained functionality. As we found, people seem to take single applications as building-blocks and arrange them into meta-applications for certain tasks. Instead of having different menu options within one application (e.g. for photos), people cluster functionality of applications as building blocks, and encapsulate them behind a folder icon or in dedicated screen areas, as Figure 4.14 shows.

This is interesting for application designers: Knowledge about which other applications have been placed in the neighborhood of an application is valuable for the designer of that particular application, since this will provide him with insights about what functionality might be worth integrating into the application itself (e.g. payment options into shopping applications, picture sharing options into camera

applications). In Chapter 3 we concluded that a similar relation between applications can be concluded from application usage chains. The analysis of the spatial relation between icons further gives rise to a better understanding of user behavior and needs. In addition, this is interesting for mobile operating systems: Once the system determines that a user is clustering applications together that provide a complimentary functionality, the system might provide support for a UI that goes beyond folder icons. Instead of arranging icons into folders, the system might provide a new synthesized application that incorporates the single applications as building blocks and acts as a distinct application on its own.

The Case for Context-aware Paging

The iPhone design, as well as pioneering work on context-aware mobile launcher menus, suggests that application icons should be ordered from top-left to bottom-right in a text-like flow related to contextual relevance of applications. This is also the conclusion of Fukazawa et al. [101] as well as Kamisaka et al. [138] and Shin et al. [229] who use this design approach to build and study adaptive launcher menus. According to our results, this assumption needs to be revised. The results of our Android screenshot study shows that people place application icons on the lower instead of the upper part of their mobile screens for launching applications, and put other content above.

As such, for purely adaptive icon menus we conclude that the mapping of relevance of applications to screen positions should be the opposite of what is used up to now: Application icons should be shown from most important at the bottom to least important at the top, to make the application that is most likely to be launched next most easily accessible. For combining adaptive and static menu items within split menus [224], we propose to adapt our Android participants' pattern of putting static icon menus at the lower part and the adaptive content at the upper part of screens. This will combine fast access to static icons (leveraging motor memory) with space to present adaptive content.

However, we do not suggest implementing auto-sorting icons in adaptive launchers as proposed by others (e.g. [229]) since this would break the mental models users build of their menus [270]. We instead suggest implementing adaptivity on the higher granularity level of pages and folders. We found evidence that people dedicate folders to specific contexts, and this provides evidence that people build their own task-related menus. Therefore, we propose that launchers should sup-

port users by forwarding them directly to the place that contains the application with the highest probability of being launched. Reasons for this are that usage of mobile applications (e.g. weather applications, social applications and games) is not equally distributed over the the course of the day, as shown in [38, 229] and Chapter 3 of this work, and in addition such applications can be found on different pages and within different folders (see Figures 4.6 and 4.14). In contrast to automatically rearranging icons — where people have the feeling of losing control [229], which might be emphasized or caused by the phenomenon of change blindness or inattenational blindness [77] — this would save time taken to search for the application and navigate to it, and yet keep the user's icon layout and mental model in sync. This implies that for mobile context-aware smartphone menus — which are motivated by users' perpetually changing contexts [18, 38, 229] — one should leave the myopic level of single applications for arranging icons and instead provide adaptive support on a higher level: e.g. jump to a folder/page (in the case of iPhone/Android-like launchers), scroll to the right position in an application list (in the case of WindowsPhone-like launchers), or open the right meta-applications as introduced earlier.

Exploiting Icon Placement and Spatial Proximity

People do not necessarily remove applications to solve the problem of limited space. We encountered the phenomenon of people having a special place to move unused application icons, referred to as "the land of misfit apps" by one participant. Either they place them on a special page in their launchers, or bury them in special folders. This is particularly the case for applications that cannot be uninstalled from a device, but also for applications that are only rarely used or do not fit to a user's general device usage. Since the deinstallation of applications does not always happen, this "burying" of applications can be used as an implicit signal about application quality, e.g. to inform application recommender systems. Also, a rating of applications might be inferred from the icons' position on the screen to build valuable user profiles, with applications on the first page implicitly rated as good, and applications on the last page implicitly rated as bad.

Since applications' categories are defined by the developers, who might have various reasons for putting an application into a specific category (e.g. into *Social* for exploiting it as a marketing label), the categorization schema of application markets can be enriched from the user's perspective. Two applications belonging to

the same category are not necessarily related. For the end-user, it would be better to give a task-related overview.

Further, we have found participants who cluster their applications in a way that is even more specific than the market categories. For instance, Figure 4.14 shows four folders of a user who has assorted applications related to photography, but the user has clustered them into more specific partitions: general camera applications, applications for organizing photos, applications for editing photos, and applications for making funny photos. Similar fine-grained sorting schemata can also be found among other users and for other topics, e.g. racing games vs. brain-twister applications, or traveling by car vs. traveling by public transport.

We have seen previously that clustering of applications into folders results from people's subjective assessment of relatedness, and people put effort into the arrangement of their icons and create a valuable — yet unused — source of information. In the line of thinking of Shipman et al. [230] we propose to exploit the spatial layout of application icons within people's launchers to infer relatedness between applications. This is complementary to exploiting temporal chains of application launches as discussed in Chapter 3 (cf. also [38]) and complementary to clustering applications based on their textual descriptions [24].

Further, we propose that looking into the ways that people arrange their icons might also reveal interesting information about the users themselves. For instance, extroverts with lower level of agreeableness have been found to spent more time customizing their phones' look and feel [54].

Similarities and Differences to Desktop Computers

While for desktops Barreau and Nardi [19] found three different types of information to be organized — ephemeral, frequently-used, and archived — we could not find an area within smartphone menus where people place ephemeral information, i.e. fast changing icons. This is surprising since one would expect people to have ephemeral states since their mobile environments are perpetually changing. Though we did not cover any temporal dimension in our study, we found that users do not arrange icons as frequently as one might assume (median of 1-10 times in the last month). Instead, on smartphones ephemeral information (e.g. m-ails, todo lists, appointments) is contained within applications and cannot be embedded into the launcher itself. However, we also found that frequently-used icons have a spe-

cial role, and that archived icons exist in the form of loosely organized launcher subparts.

We found a strong relation to context of use for icon arrangements. While on desktops users adapt their organization to their current task [19], we argue that this phenomenon is even more specific for mobile devices. For smartphones one can find a stronger and more diverse context-related arrangement of applications; e.g. we assume that on a desktop computer one would only rarely find menus customized for specific locations or shopping.

Further, and most interestingly, we found ergonomic aspects of smartphone interaction to have a major impact on icon arrangement. This motivation was explicitly mentioned by participants of our study, and this is also implicitly suggested by the results of the Android study, where people place icons at the bottom of screens.

One Launcher Fits All

Overall, subsuming all aspects, it appears that people divide their menu into three common conceptual spaces that are distributed on the menu pages: (1) most often used and important applications, (2) applications that relate to each other, (3) and least used and unimportant applications. A common spatial distribution is: most frequently used applications on the first page, followed by pages with folders for applications that are related, and on the last screen applications that either are only used rarely or that do not fit into any cluster of related applications. Further, one-handed interaction should be taken into consideration when designing launcher menus.

The concepts we found for mobile launchers basically determine how people arrange applications. These patterns are applicable on smartphones where people can move the applications' icons, and which cope with the lack of space by allowing users to have applications on different virtual spaces (e.g. pages or folders, or scrolling a long list of tiles on the Windows Phone). We propose this conceptual distribution to smartphone designers to build launchers that need to work for all users. These practices relate partially to what people do on stationary computers.

While other platforms have their own constraints for arranging icons, we argue that the concepts we have found for Android and iPhone devices are independent from specific constraints for icon layouts.

4.3.4 System for Supporting Icon Arrangement

In this section we present a system that supports customization of smartphone launchers. Its design is informed by the study of launcher menus we previously presented, and mainly driven by the finding that people put related applications onto the same pages and into the same folders within their menus. This system is designed to exploit this common pattern by mining association rules for icons. Based on that we are able to give hints to users on where to put icons, how to alter their launcher menus, and which new applications to install.

The ideas of this system are to crowd-source the mental effort that people put into their icon arrangements, and to exploit this knowledge base to support other people during their icon arrangement.

Prototype Implementation

Given the phenomenon of icon arrangement by application relatedness, we aim to exploit it to give hints on how to arrange icons. We follow a crowd-sourcing approach, assuming that the way most people arrange their icons also is beneficial for other individual users.

Design. Our system is based on the idea that an optimal icon arrangement emerges from combining common patterns that other people apply. The main finding of our preliminary study that informs our design is that people tend to put related applications close together, i.e. onto the same page and into the same folder. Of course, however, some people might have specific personal reasons for arranging icons that are different from common patterns. Therefore it is worth mentioning that this system only suggests placement of icons, arrangement of icons, and installation of applications, and does not compel the user to make any change or alter the launcher without the user's approval. Further, to keep the user's mental model of his menu aligned with the menu's actual structure, we do not change the structure automatically but rather recommend that the user does so, by presenting hints. Then it is up to the user whether to follow the recommendations or not.

Implementation. Since there is no API on the iPhone for accessing the menu structure, we use a computer-vision based approach as a workaround for our prototype: Within our system people are asked to upload screenshots of their launcher menus. From that we can visually recognize the applications' icons and reconstruct the menu layout. This is a technical limitation of our current prototype. We used

the same implementation as we did for analyzing the submitted screenshots of the study presented previously.

Association Rule Mining

We exploit the relation between co-occurrence and relatedness by mining association rules from the screens. Currently our recommendation of icon moves is following a crowd-sourcing approach: Based on the launcher menus of users who have already submitted their screens, we make menu structure recommendations to other users. The association rule mining of our prototype implementation is being leveraged by the data set of the screenshot study presented previously. The association rules in our system have the form of $\{a_1,\ldots,a_n\} \Rightarrow a^*$.

We use the *Apriori algorithm* [3], which allows us to mine for specific relations of items in large data sets. A common use case of this algorithm is to analyze basket data in retail contexts. In short, it provides rules saying how likely it is that a product is in a customer's basket if specific other products appear in the same transaction. TO use this algorithm, we treat single menu pages and folders as single transactions to mine the rules from the data we have. One rule can be interpreted as follows: People who have icons of applications $\{a_1,\ldots,a_n\}$ on a page (or in in a folder) also have the icons of application a^* on the same page (or in the same folder). Every rule has a confidence value, which is the conditional probability that application a^* appears on a screen given that it already displays the icons of applications $\{a_1,\ldots,a_n\}$.

Support for Arranging Icons

Based on the associations of icons, which the system mines form the collected data of menu structures, it is able to support users with hints for arranging icons, with placement advice for icons of new applications, and with recommendations for installing new applications.

Arrangement Hints. To give hints for re-arrangement of icons we filter out those rules where a^* is not installed by the user. Then we look for pages and folders of the user that contain all applications $\{a_1,\ldots,a_n\}$ of a rule. If we find such a set and if a^* is currently on a different page, we will recommend putting a^* onto the same screen. Figure 4.15(b) shows a list view that gives instructions to the user as to which application to put near which other application. In the figure,

(a) Recommendations for applications (b) Presenting hints for arrangement.

Figure 4.15: Screenshots of our system to support icon arrangements. (a) shows a list of recommendations. The applications might be of interest to the user because other people have put them onto the same page with applications that the user already has installed. (b) The system provides suggestions on how to arrange the icons.

the system recommends putting Facebook near Skype (both social applications), Google Earth near another Map application (both mapping applications), and the IMDb application near YouTube (both movie-related).

Placement Advice. The user can ask the system for placement advice for a specific application, for instance when the user wants to install the application and is unsure where to place it. The system looks up all the rules having a^* equal to the application in question. Then it returns those pages of the user that contain $\{a_1,\ldots,a_n\}$ of the found matching rules. The user is then advised to put the application onto one of these pages after installation.

Suggesting New Applications. Besides giving support for icon arrangements, the system also exploits the association rules to give recommendations for applications that are not yet installed by the user might be of interest to him. Therefore we take all rules with $\{a_1,\ldots,a_n\}$ being a subset of the user's applications and aggregate their consequent applications a^* to a list ranked by the confidences of the rules. From that list we recommend the top 10 applications. Recommendation of applications will be discussed in Chapter 5 in more detail.

Usually, association rules are used in recommender systems based on market basket analysis, i.e. a transaction comprising items that people have bought. In our case,

we not only consider which applications the user has installed, but also how she has clustered the applications onto pages and into folders. That means we exploit even more implicit feedback beyond purchasing statistics.

4.4 Summary

In this chapter, we investigated the housekeeping of mobile applications done by end-users on their smartphones. We presented two studies on icon arrangement in smartphone launchers: one suggesting that icon arrangement is influenced by context, and a second one investigating long-term icon arrangement. Based on the insights gained into how people interact with the icons in their smartphone menus, we have built two systems that support and exploit housekeeping operations on smartphones.

In the first study, presented in Section 4.2, we investigated if context has an impact on people's icon arrangement on smartphones. We gained first insights suggesting that users in different contexts move different applications to the foremost position on their devices. These findings suggest that a user's context has an impact on his icon arrangement behavior, and the screen position of an icon relates to the relevance of an application. Based on that, we presented the idea of exploiting a user's icon arrangement to infer the relevance of the installed applications, and we presented a system implementing this idea. We also discussed that these preliminary findings suggest that further insights are needed to understand how people arrange their icons, which motivated another study.

In the second study, presented in Section 4.3, we investigated which concepts naturally emerge when people arrange their icons on their smartphones. On the one hand we have found common concepts that are used by most people, and on the other hand we found very specific reasons for arranging icons. The majority of smartphone users arrange application icons for four reasons:

(i) so they can reach the most-used applications quickly,
(ii) to cluster similar applications together so they can easily choose between alternatives and follow-up applications for a certain task,
(iii) so that their launcher looks nice,
(iv) or so they have good usability.

These concepts emerged from a qualitative study of more than 130 smartphone users. Quantitative evidence of certain characteristics was found in the analysis of more than 1,400 screenshots of our participants' launcher menus. Further, we found that the concepts people apply impact the layout of icons, e.g. arranging application icons based on application-similarity results in more folders on the first page and re-arranging icons more often. Finally, we discussed how the inherent value of icon arrangements can be exploited (e.g. to improve categorization of mobile applications on mobile application stores), how mobile application launchers can be improved (e.g. by recognizing users' self-built meta applications), and how context-aware launchers could benefit from taking pages and folders instead of icons into account as a higher level of granularity. We also compared sorting icons on smartphones to sorting information on desktops in general. The results of this study informed a system for supporting the housekeeping of smartphone launchers by providing support for arranging the icons of installed applications and placing icons for new applications.

The work presented in this chapter provides insights into how people arrange icons in their smartphone launchers, and provides an understanding of people's habits and how context impacts the process of housekeeping of applications on users' devices. With the results presented in this chapter we were first to study this topic (e.g., see publications [31, 40, 30]), and first to provide evidence for patterns of housekeeping of mobile applications that is drawn from a large data set going beyond anecdotal findings.

Chapter 5

Discovering Mobile Applications

Before people can launch their applications or have a set of applications that they need to housekeep, they first have to download new applications and install them on their device. As introduced in Chapter 1, mobile application stores are the main resource for new applications. In this chapter we deal with people's problem of finding new applications on such repositories, and provide assistance in the form of recommender systems help them to find valuable new applications.

The results of this chapter have been presented in five publications [36, 42, 140, 29, 32]. Work related to this chapter can be found in Section 2.3.4 on context-aware recommender systems.

5.1 Introduction

As described in Chapter 1, the recent evolution of technology in mobile computing and the rise of mobile application stores has eased the development and the distribution of mobile applications. This has led to an increasing number of available applications that are accessible on mobile application stores for end-users. With an increasing number of available applications, the user's problem of discovering valuable applications has come into existence. Discovering describes the process

of finding something unexpectedly or in the course of a search.[1] While in Chapters 3 and 4 we investigated how people interact with the applications they already have installed on their devices (launching them and arranging their icons), in this chapter we deal with the end-users' challenges of finding new applications to install and use them on their smartphones. In particular, this chapter is motivated by the question of how to support users in discovering useful new applications for installation. Searching for applications on application stores is the most used and most preferred method of finding new applications; second is referring to friends.[2]

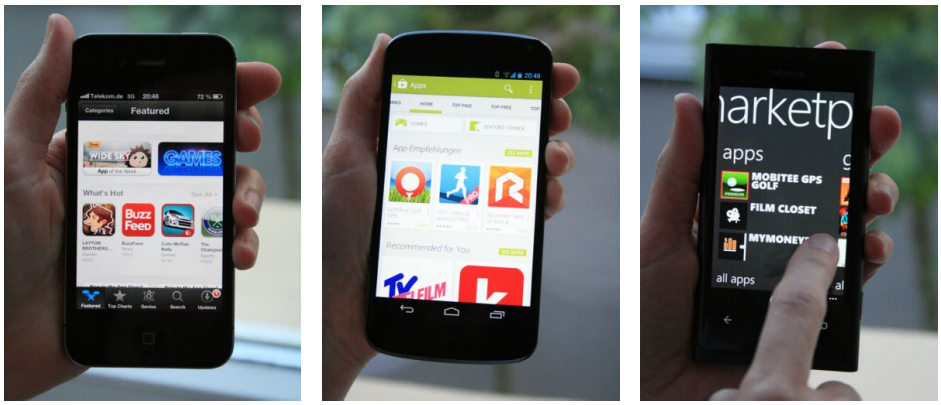
In many domains assistive systems, which are called recommender systems, have been developed to address this challenge. The general goal of recommender systems in general is to guide users to relevant items in a large mass of items [206], in particular in situations when user have not personally experienced the alternative available items [202]. Domains where recommender systems are used are for instance recommendation of books, music, videos, or friends on social networks (see also Chapter 2). For the first mobile phones that were capable of having additional applications installed onto them, such recommender systems have not been available. Currently, the most used recommender systems that suggest mobile applications neglect that the usage of smartphones can be characterized by a perpetually changing user context, as we explored in Chapter 3 and as can been concluded from related work (see Chapter 2).

In this chapter, we develop an approach for supporting the discovery of mobile applications with a focus on improving current systems by incorporating contextual information and signals such as discussed in the previous chapters. As such, we introduce the idea of context-aware recommendation to the ecosystem of mobile applications.

The outline of this chapter is as follows. First, in Section 5.2 we explore the design space for recommender systems that suggest mobile applications on end-users' devices. We look into the different dimensions and techniques for capturing the required pieces of information to run such a system, such as information about the users, the items, the contexts and the corresponding relevances. Next, in Section 5.3 we present the deployment of a recommender system for mobile applications. The architecture of this supportive system is based on the presented design space and is implemented as a recommender system for mobile applications on the An-

[1] Oxford Dictionaries: *"discover"*, Oxford University Press, http://oxforddictionaries.com/definition/english/discover, last accessed on 02.07.2013.

[2] Nielsen Company: *The State Of Mobile Apps*, http://goo.gl/Ubhc8, last accessed on 11.06.2013.



(a) iOS AppStore (b) Google Play Market (c) Windows Phone Store

Figure 5.1: Users browsing though application stores of major phone platforms. Common features are recommending applications by popularity (see e.g. "What's Hot" in (a) and "top free" and "top paid" in (b))

droid platform. The purpose of this system is twofold: For our work, it serves as a proof of concept and as a testbed, and in addition it is freely available for end-users on the Android market. Finally, in Section 5.4 we present a new approach for the evaluation of context-aware recommendation of mobile applications.

5.2 Design Space of Recommender Systems

Most applications that are available on mobile application stores serve their users for special purposes and have been developed to support its users during specific tasks or to meet certain needs, like for instance finding directions, finding the best price, or taking pictures. This can already be seen from the categorization schema of applications on application stores with classes for applications like lifestyle, sports, social and productivity, which we already discussed in Chapter 3. Figure 5.1 shows users browsing through application stores of the major mobile platforms. Current stores provide support by recommending popular applications like "What's Hot" in Figure 5.1(a), or personalized recommendations as in Figure 5.1(b), but current application stores typically do not take into account context for recommending applications.[3]

[3]The latest version of Apple's iOS incorporates location to show applications that are popular in the user's neighborhood. At time of writing this thesis iOS 7 only a developer preview was available, and the feature was not available for end-users.

Once installed on a mobile device, an application can be used anytime and anywhere. However, since mobile device usage is characterized by perpetual changes in the user's context [18], the set of useful and required applications also changes as we found in Chapter 3. This occurs both because a user himself is mobile and changes location and activity, and because the environment of a user also changes. In Section 2.1 we defined context as any piece of information that is relevant for a user's interaction with a system, e.g. with respect to individuality, location, time, relations, and activity [272]. For the notion of context in the current chapter in particular we refer to the user's activity for conceptual aspects, while we will use location and time of the day for technical aspects and our implementation.

The relation between context and mobile applications is widely neglected in current recommender systems that are available to support the discovery of mobile applications. Since a mobile user's interests and information needs (cf. [67, 66]) as well as application usage (cf. Chapter 3 and [251]) are dependent on context, the recommendations a system gives should be adapted to changes in context. For instance, it is not considered that people are less likely to use game applications when at work than while being at home [251], and that most users at home might be more interested in applications for leisure time activities than in applications for productivity. Further, applications for scanning barcodes to retrieve product information and compare prices of products can obviously be used in different contexts, but in particular they are of interest and provide value to the user when she is close to items that have barcodes, e.g. products that she sees while shopping. Also, applications for retrieving the latest train schedules, for instance, should especially be recommended to people who are at a train station, plan to go by train or are currently traveling by train. Since systems can be improved by incorporating context information [160, 105] we argue that commercial systems that recommend mobile applications — like for instance Apple Genius, the Amazon AppStore or Google Play Market — underachieve since they neglect the ever-changing context of their users and solely exploit sales figures and user ratings.[4]

In summary, and in contrast to most related context-aware recommender systems presented in Section 2.3.4, mobile applications are a novel type of item to be recommended for two reasons in particular:

(1) Context plays a crucial role in the recommendation of mobile applications.

[4] Jacqui Cheng: *App Store's new sections still don't solve discovery problems.* http://goo.gl/uEvPz, last accessed on 20.06.2013.

(2) The usage of a mobile application can be continuously captured alongside the context information due to the nature of a mobile application.

The latter means that applications are basically software processes that can be observed by means provided through the operating system, and this can be used to trace interactions with the application, as described in Chapter 3. Incorporating such observational information on interaction goes beyond user ratings or implicit feedback based on sales statistics, as we will discuss in Section 5.4.

In this section, we contribute a conceptional foundation for supporting discovery of mobile applications by exploring the design space for recommender systems that suggest mobile applications in particular. We outline options for building such systems and enabling the context-aware recommendation of mobile applications. As an example and as a proof of concept we will present a prototype for a context-aware recommender system in Section 5.3, that is designed within the capabilities of the design space that we will explore in this section.

5.2.1 Construction of Design Space

The main idea of context-aware recommender systems can be subsumed within the following formula [2]:

$$users \times items \times contexts \rightarrow relevance$$

This mapping indicates that the relevance of an item depends on a user and her current context, where the relevance in the course of our work describes the utility value of an application for a user in a certain context. This formula further covers the dimensions of the design space of recommender systems that aim to incorporate context, as is the case for recommending mobile applications. In the remainder of this chapter, we refer to an application's relevance as a variable describing how useful an application is to the user in her current context. Nota bene, this value cannot be estimated a priori since it is determined by the user's perception of whether something is actually useful or not, but it needs to be formalized so that it becomes ascertainable for a software system. In Chapters 3 and 4 we looked into launching of applications and icon arrangement of applications as possible signals for deducing the relevance of applications.

Capturing the Required Parameters

Context-aware recommender systems aim to suggest those items to a user that are relevant to her in a specific context. Computationally, this tasks entails computing for a given user and her actual context, those tuples $(user, item, context)$ that have the highest relevance values, and then to decide on the subset of items to recommend. To anticipate the relevance of a certain item for a user in a certain context, the space of

$$users \times items \times contexts$$

needs to be filled with values for the mapped relevance values. Based on these data as a knowledge base and by applying different available algorithms, for instance adapted versions of collaborative filtering using pre-/post-filtering (e.g. ([2, 16]) or context-aware models (e.g. [140, 228]), those relevance values that have not been recorded beforehand can be estimated to enable the recommendation of the most relevant items. For the most part, this is the case, since the knowledge base is sparse and a recommender system usually only aims to recommend items that a user has not yet installed, i.e. no data is available for that combination of user and item. To fill the knowledge base and to improve the accuracy of the given recommendations, the four parameters, as a tuple of $(user, item, context, relevance)$, have to be captured as often and as accurately as possible.

In the following paragraphs, we will conceptualize and describe the design options for modeling and capturing these four parameters in the domain of mobile application recommendation. For each parameter, we will distinguish between an explicit and an implicit option for capturing the required pieces of information. Explicit capturing means that the user is required to input some information and thereby gets interrupted from his current task.

Implicit capturing means that the data can be gathered from users' actions, which are not primarily executed by the user to interact with the system, but which can be exploited and interpreted by the system [219]. The recommender system can automatically extract valuable information from observation of users' interactions with smartphones without any user disruption or effort [12].

The distinction between explicit versus implicit capturing is important and worth mentioning, since in the domain of smartphone applications the technology gives new means to capture these parameters. It is not only possible to track context information like the user's location due to the sensors that have become common

on mobile devices. The most important advantage, and a unique point about recommendation of mobile applications, is that the user's experience of an item takes place where the software of the recommender system itself is implemented and running, i.e. on the user's smartphone. Therefore, there are new technical as well as conceptional opportunities to implicitly observe the experience and capture the required relevance values (as already described for the concept of the *AppSensor* in Chapter 3). In contrast, for instance, to a recommender system for books, where one can only with difficulty capture a user's experience of a book implicitly after he bought it (e.g. for how long and how often he has read it of if he has read it at all), this is possible when recommending mobile applications. The user's experience of an item is an elementary parameter in a recommender system since it impacts the relevance and value the application provides. More on exploiting this fact and using it as feedback for evaluation will follow in Section 5.4.

Ascertaining Users and Items. The knowledge of the user's identity is fundamental for personalized recommender systems. To give personalized recommendations, the systems are required to relate all captured information to a unique distinguishable user. A user herself could explicitly notify the system of her identity, e.g. by logging into the system or unlocking the mobile phone with credentials. For an implicit capturing of the user's identity, the fact that mobile phones are personal devices can be utilized [49]. Therefore, all information that requires a reference to the user's identity can also be related to the user's device without any loss of information and without any effort for the user. However, this implicit capturing is only possible when the recommender system is running directly on the device. In contrast, software for stationary computers that recommends mobile applications (e.g. Apple's iTunes[5]) requires an explicit identification of the user.

In the domain of mobile application recommendation, the items to be recommended are obviously the mobile applications themselves. Mobile applications can be either native or web-based. Although web-based applications — i.e. applications which are running in a browser environment built with web-technologies like HTML and JavaScript — have advantages over native applications, the large majority of applications that are being used on mobile devices are native applications, even for accessing content provided through the Internet [245]. The recommender system needs to be able to track which application the currently captured param-

[5] Apple: *iTunes — Everything you need to be entertained*, http://www.apple.com/itunes/, last accessed on 06.07.2013.

eters relate to. On the one hand, this can be done explicitly by having the user reference a certain application or the system ask about the relevance of a certain application. On the other hand, since the nature of the items is very close to the nature of the recommender system itself — i.e. pieces software being executed on the smartphone — the items in this case can also be tracked implicitly. On smartphones the available applications can be queried, unless the mobile operating system is prohibiting access to required method calls, as already described in detail in Chapter 3.

Determining the Context of Use. While the concept of recording relevance values within a two-dimensional matrix of $users \times items$ is common within most state-of-the-art commercial recommender systems, the adoption of context has added a new dimension to recommender systems [2]. Taking this new dimension of context on its own, we are only beginning to understand how it impacts mobile technology use. In order to provide users with context-aware recommendations, the system needs to recognize the context of the user that corresponds to the point in time when the relevance of an application is being determined. As described earlier, the consideration of context is important for the domain of mobile application recommendation, since mobile users are on-the-go and their contexts, their tasks and their interests are perpetually changing (cf. e.g. [18] and Chapter 3). Context mediates the perceived merit of a mobile service [160].

These pieces of information can either explicitly be provided by the user, for instance in the form of labels assigned to their contexts, or alternatively those variables can implicitly be obtained from the sensors on the user's device as well as from other sources. For explicit context capturing, the users are required to annotate their current context with attributes and provide information on their current activity. For instance, this can be done by providing keywords (as for the approach of context tags that we describe in [33]) or choosing their current context within an ontology. Such ontologies for example can distinguish between *traveling*, *working* or *shopping*. It is worth mentioning that this does not have to be done in real time. A user could also annotate her past activities or provide information on her future actions.

Current mobile technology also makes it possible to implicitly capture and interpret information about the user's current context. For instance, location, time, and acceleration data can be used to reason about a user's activity [185]. Such data can be obtained in real time from the sensors of the smartphone. Other sources for

collecting information on a user's context are the calendar, conversations, and activity streams of social networks, which can also be accessed through APIs within applications and services running on smartphones.

Quantifying the Relevance. The relevance parameter refers to the value that an item holds for a user in her current context. In the domain of mobile application recommendation, this relates to the usefulness that an application has for a user for solving her current task or to fulfill her current requirements, e.g. to recognize a song in a disco or to compare prices of products. As stated previously, this parameter describing the usefulness of an application in a specific context cannot be captured or quantified directly.

An explicit capture of the relevance requires the user's attention. However, some ways to capture the relevance of items within recommender systems have evolved and became common over time. Firstly, people can leave a free-text comment for an application, which serves as a guideline for other users. On the one hand, free-text comments are very expressive and can help other people to form an opinion on the relevance of an application, but on the other hand their semantic meaning is not directly readable by a machine. Further, absolute rating schemas like the 5 star Likert scale or the metaphor of *thumbs up / thumbs down* have become widespread and are therefore easy for users to understand. They are also very clear in expression since the user directly transforms his experience into a computable value. Additional relevance feedback can be designed as a relative ranking of applications, e.g. by letting the user sort the applications (as in the mockup application we used in the study presented in Section 4.2).

Implicit relevance feedback on mobile devices is available through observation of the user's device interaction. The relevance of an application therefore can be correlated with certain measurable variables. Based on the assumption that an application is only relevant when it is actually used, the application usage can be exploited for capturing relevance. For instance, it can be analyzed how often and for how long an application has been executed (cf. the concept of the *AppSensor* described in Chapter 3). Application installs, uninstalls, and updates can also be logged. Also, the arrangement of the application icons holds promising as as source of implicit relevance feedback, as described in [31] and Chapter 4.

Accessing the Recommendations

Besides the capture of the presented parameters, all collected pieces of information need to be aggregated before recommendations on applications can be given to smartphone users. The aggregation makes it possible to query the data for the relevance of an application for a user in a certain context. In general, the idea of incorporating context information into recommender systems is not new, and algorithms can be adapted from existing approaches, e.g. for splitting the relevance measures of applications according to context [16, 55] or creating new models for incorporating context [228, 140, 139].

Similar to data gathering, as described before, the user's access to derived recommendations can also be designed either in an explicit or implicit way. For an explicit recommendation access, the user will himself pull recommendations from the system. This is the case for the largest currently available recommender systems for mobile applications, i.e. Apple's Genius on the iPhone and the Android Market Store on the Android phone. Both recommender systems grant access to the recommendations directly on the user's device, when the user opens a dedicated application store application and asks for recommendations. Implicit recommendation access should follow a push-based approach [188]. Such an access can be designed directly via the mobile main menu, as presented by Vetek et al. [253]. The recommended applications can directly be started from the main menu, which in this case acts as a context-aware menu if there is no distinction between applications that are already installed and those that have not yet been used before.

5.2.2 Discussion of Design Space

The design space presented in this work comprises four different dimensions with two options each. It allows for 16 different conceptual designs for systems in the domain of mobile application recommendation. Further, for each option different implementations are possible, e.g. for the option of implicitly collecting signals for modeling relevance feedback, one can choose from different implementation possibilities. Further, the presented options are aimed more at exploring the field and constructing the space rather than at providing a holistic overview of possibilities.

Combination of the design options are, however, constrained by factors like the platform and mobile operating system that a recommender system is being built for, as not every operating system provides access to all of the possibilities previ-

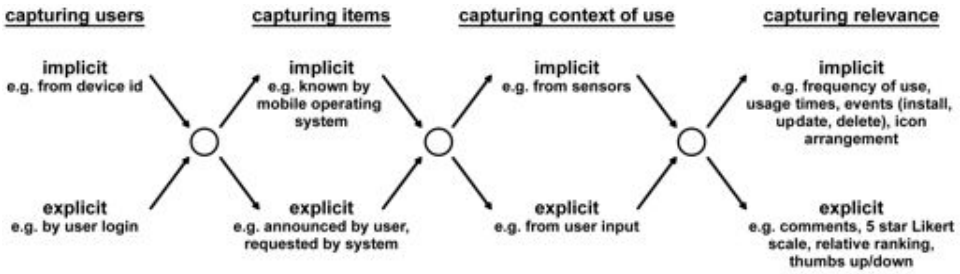


Figure 5.2: Design space for context-aware recommender systems that suggest mobile applications.

ously discussed. For instance, there are currently web-based recommender systems for mobile applications that are used in browsers on stationary computers (e.g., the online platform *appolicious*[6], that cannot make use of implicit parameter capture, since they cannot apply means for tracing application usage directly on the device. Further, not every configuration of options provided by the design space is reasonable. For instance, an implicit capturing of the relevance values will be counterproductive due to the need for additional user interaction, if an implicit capturing of the corresponding application itself is not possible. However, the design options within each dimension are not mutually exclusive. Especially for the parameters of capturing context information and the relevance values, explicit and implicit implementations for capturing can be combined. For instance, a user's explicit rating on a 5 star Likert scale can be combined with the number of utilizations of an application, which we analyzed in detail in Chapter 3.

The overall design goal of a recommender system should be to support the user in the process of retrieving appropriate content. Having the recommender system ask for feedback from the user will create overhead on the general application usage, as we will discuss in general for mobile applications in Chapter 6. Therefore, a mobile recommender system should favor implicit over explicit parameter capture where possible to avoid creating additional effort for the user, especially because of the negative impact of mobile task disruption and limited cognitive resources (cf. Chapter 6 and [12]).

With the inclusion of context as a new dimension, the complexity of recommender systems increases. Firstly, new reasonable algorithms for data aggregation and

[6]Appolicious: *iPhone apps and iPad apps*. http://www.appolicious.com/, last accessed on 11.06.2013.

relevance estimation are required to cope with this new dimension. Alternatively, existing approaches can be adapted to make recommendations context-aware, like for instance context-based splitting of item ratings for collaborative filtering [16, 55]. Secondly, the effect of the cold start problem increases, as we will encounter in Section 5.4. While hitherto the problem appeared if no data for a user or for an item was available, sparse data on the context dimension will emphasize this problem. Thirdly, the amount of data within the system increases, since it is multiplexed by the cardinality of the new dimension. Additionally, the implicit parameter capture (in contrast to explicit capture) will allow more meaningful data to be collected.

5.3 Deployment of a System Recommending Mobile Applications

In this section, we will present an architecture blueprint and a concrete implementation of a recommender system that suggests applications based the Android platform and its application store. The system is designed based on the design space discussed previously in Section 5.2. Following our research methodology, a deployment of this system will serve as a testbed for the evaluation of different approaches for recommending mobile applications that we will discuss in Section 5.4.

5.3.1 Architecture

Our approach aims to support users to find new valuable mobile applications in application stores by reducing the set of available applications to the set of contextually relevant ones. The user is able to install and make use of these applications on her smartphone, as Figure 5.3 shows. Therefore, the output of the proposed architecture for a context-aware recommender system is the set of relevant applications. The input of the system is information about users' contexts, along with logged data about the users' interactions with their mobile applications.

The next sections describe the design of the three required core components from the bottom up: the context reasoning, the usage interpreter, and the recommender system.

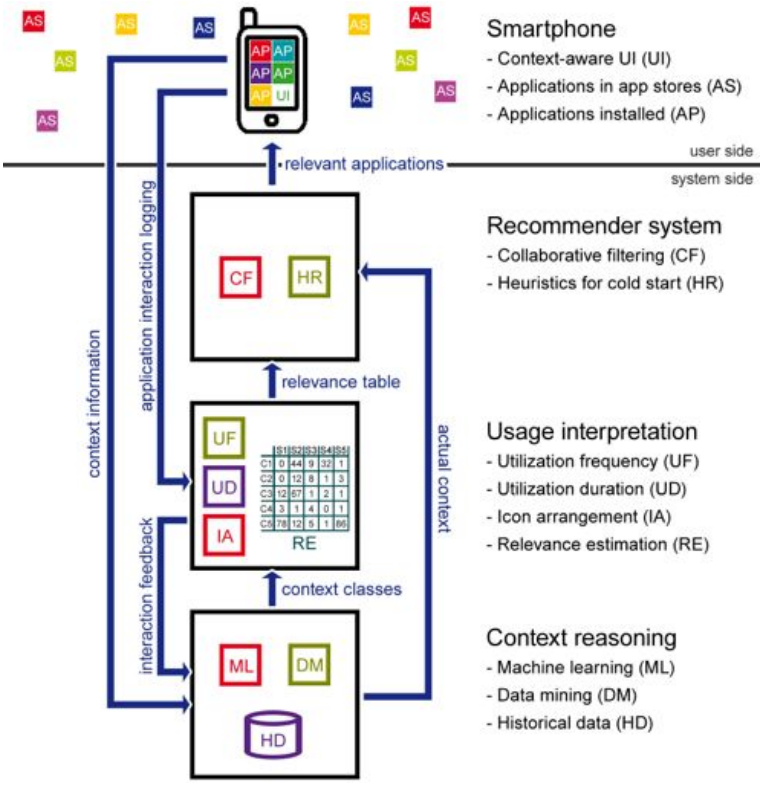


Figure 5.3: Architecture blueprint of recommender system to support mobile application discovery.

Context Reasoning

Context is any information that characterizes the situation of a person as defined in Chapter 2. On today's smartphones, there are now many of sensors providing pieces of information that are valuable to this end, e.g. local time, user's speed, brightness of light, temperature of environment, device's acceleration, a person's geographic position and azimuth. Such low-level physical context features can be measured directly. Zimmermann et al. [272] suggest relations as an additional context category. Raw data can be enriched with such knowledge for deriving more context information, e.g. user's mean distance to friends or other social contacts, density of people nearby, distance traveled in the last hour, or the user's current distance from home.

Especially for physical context features, but also for more vague concepts like collaborative tagging, metrics are available. Methods from machine learning and data mining can be used to reason about contexts. For example, the contexts *at-*

work, *at-home* or *on-the-move* can be distinguished by low-level data [251]. Based on a large pool of historical data, patterns and clusters can be found that represent different contexts (see e.g. [99]).

Interpreting Service Usage

With knowledge of the user's actual context, we can observe his behavior accordingly and bring it into relation with his context. By interpreting the actions of the user we can infer knowledge about the relevance of a service in a certain context. In a *waiting-for-the-train* context the application for the train schedule will be used presumably much more often than a service for bank transactions. Since the context of use mediates the perceived merit of an application [160], we can assume that in certain contexts irrelevant services are used not at all, and relevant ones are used much more often — as we also analyzed in Chapter 3. In contrast to incomputable properties like the relevance of a service and the satisfaction of a user's current needs, this property can simply be related to observable user behavior like launching of applications (cf. Chapter 3) or context-related arrangement of icons (cf. Chapter 4).

Another metric can be derived from the duration somebody is using an application. For example, if the train schedule is used during the whole ride, it is likely more relevant than the flight schedule if the latter is only used briefly. Although both might be used only one time, the usage duration of the latter is shorter. Because the user will focus her attention only on situationally useful applications, the application utilization time can be correlated with its relevance — assuming that the longer the application is used the higher its relevance is.

Besides all this positive feedback making a service more relevant, there is also a benefit to negative feedback, which gives a hint about irrelevant applications in a context. For example, this can be concluded from the deletion of an application and from observation of when people move them to special places in their menus (see Chapter 4). Also, a temporal decay of the relevance of applications seems promising. Applications become irrelevant when they are not used anymore.

In the architecture designed in this section, and as shown in Figure 5.3, the interpretation of application usage leads to the creation of a table that contains the relevance measure of a service for each context class. All figures are aggregated for an application in a context, e.g. by summing up the usage counters.

Recommender Engines

The recommender system will suggest applications to the user which have been useful for other people in the same context. Having figured out the user's context, a lookup in the table gives the weighted relevancies of the applications. In the case of fuzzy context reasoning, the likelihood of the contexts needs to be considered in the relevance weights.

The applications are pushed to the user's device from the most relevant to the least relevant according to their weights. Due to the physical limitation of screen size there must be a reduction to the most relevant applications. Also, the user should not be burdened with too much functionality — since being mobile already is cognitively costly and mobile device usage is already intermitted with other activities (see Chapter 6 and [45, 142, 181]).

A cold start problem (see Section 2.3.4) occurs when a user is in a context where no application usage has been recorded before. In this case, simple heuristics and other filters can be used, e.g. location- or content-based. Users should also be able to search and browse for services that are not yet recommended. This problem further occurs when a new application is available in the recommender system. When a similarity measurement for the applications is available, the relevance weights could be bootstrapped from similar applications.

Usage-based Context Reasoning

Figure 5.3 also shows a flow of information from the user-given feedback into the component for context reasoning. The inclusion of the application usage data can extend the context inference. For example, the context *traveling-by-train* might be independent from the weather condition (people go by train in sunshine and rain). But if people use the *rent-a-bike* service in this context only when the sun is shining, and *call-a-cab* only when it is raining, it will be beneficial for the users if the system is able to subdivide the context, record the application usage statistics accordingly and give corresponding recommendations. By reasoning context not only on raw physical sensor data, but also on the people's mobile application usage, the recommendation becomes more valuable for the user. This relates to the idea of distinguishing a tourist from a shopper in a mall by looking into whether she is using a shopping list or sightseeing application on her smartphone, as discussed in Chapter 3.

5.3.2 Implementation

As a proof of concept of the proposed design space, we have built and implemented a recommender system for mobile applications. It is realized based on the Google Android platform, since it provides APIs to access the required context information as well as capture the application usage as described in Chapter 3. The system follows the design space explored in Section 5.2 and is based on the architecture laid out in Section 5.3.1. The technical components are a mobile client and a server. Conceptually, the system consists of the following three parts:

(1) A logger that runs on the smartphone to capture the user's identity, context information and the application usage,
(2) a central unit that runs an inference engine to reason about the user's context, relates it to the application usage, and then provides recommendations,
(3) and a user interface that offers recommendations on applications to the users.

The mobile application implements the logger for the implicit parameter capture and the recommendation interface. The logger runs as a background service and keeps track of the location, including its accuracy, the local time and speed. Furthermore, it records the start time of applications and their corresponding runtime. The identity of the user is equated with the technical identity of the device. The user is able to start and stop the logger and share her usage statistics, as Figure 5.4(a) shows. All recorded data is stored in a local database on the smartphones and is periodically uploaded to the server.

Figure 5.4(b) shows the user interface that visualizes the recommendations. It is realized as a widget that can be installed to the home screen of the device. It refreshes periodically, triggered by changes in context information (e.g. location), or when the device wakes up from standby. If the user clicks on an icon, the application either starts directly, if it is already installed on the device, or the click invokes an installation routine on the Android Market Store, if the application is not installed. As Figure 5.4(a) shows, the user is able to turn off the recommendation of already-installed applications. Currently we are not able to make this more transparent, since we cannot automatically install applications (due to permissions that the user has to grant to the applications, e.g., for security reasons related to the location API, and because users may have to make payments before they are able to download applications).

(a) Setting (b) Widget (c) Context tags

Figure 5.4: Screenshots of the prototype for proof of concept: a) settings to control logging, b) access to recommendations through a widget, and c) input of contextual keywords.

In an intermediate version of our recommender system, the user is also able to insert keywords to describe his current context, as shown in Figure 5.4(c). We supported two forms of input. Firstly, a user is able to insert new tags by typing. Secondly, the user can choose tags from a list of his recent tags or from a list of tags that other people used near the user's current location. The inserted context tags are used as explicit pieces of context information that allow the user to tailor the recommendation of the system. Since the tags were also forwarded as a status message to a social networking site, those context tags become an implicit parameter if the user's primary motivation for entering them is related to the social network. However, we removed this approach from the system, since it was not adopted by the users of our system. We described the idea itself as an approach called *Context Tags* in [33].

The mobile application uploads the recorded data to the server, which sends back the recommendations for applications. The server makes all incoming data persistent in a central database. This data is used by the inference engine that applies clustering algorithms working on the sensor data to reason about the contexts. In our prototype implementation, the inference engine uses location and time as context information, as we will describe and discuss in the next section. Additionally, the server maintains metadata about the applications, which is pulled from the Google Android Market by a crawler (e.g. application type, icon, title), and about

the devices, which is logged in addition to the data described earlier (e.g. version of operating system).

5.4 Usage-centric Evaluation

Besides designing and building a recommender system to recommend mobile applications to the user, users' perceived quality of the system needs to be taken into account when building a recommender system for mobile applications. As described in Chapter 2, current approaches mainly focus on social aspects (e.g. [103, 4]), context-awareness (e.g. [264, 228, 140, 76]) and characteristics of application markets (e.g. [227]). Exploiting context seems especially promising since smartphone usage is subject to ever-changing contexts [137, 38, 251]. As such, the set of required applications also varies according to the user's changing tasks, e.g. from picture taking to navigating to looking for a restaurant. For the evaluation of recommendations of mobile applications it must be noted that mobile applications are a special type of items to be recommended: their usage is not only conditioned by users' contexts, but the usage of a mobile application itself can also be tracked alongside the user's interaction with the application — since the application is itself a software process that can be observed (see Chapter 3 and [38]).

Most importantly smartphone use can be characterized by "dead" applications: fewer than half of installed applications are actually being used [229]. Also, in Chapter 4 we found that some users are reluctant to delete their applications and instead move them to special places in their menus. Therefore, choosing installations as an evaluation metric might result in even more applications residing unused on a user's device. Consequently, new approaches for evaluating application recommender systems are required: They should go beyond ratings, click-through-rates or download statistics. Hence, this section gives rise to new paradigms for evaluating mobile application recommender systems.

This section makes two key contributions to the discovery of mobile applications:

(1) We introduce a usage-centric approach for observing people's engagement with mobile applications at different stages along applications' life-cycles beyond installation. Therefore, we adopt the concept of conversion funnels to the mobile ecosystem, presenting *AppFunnel*. Based on this we can track the performance of different recommender engines according to this funnel.

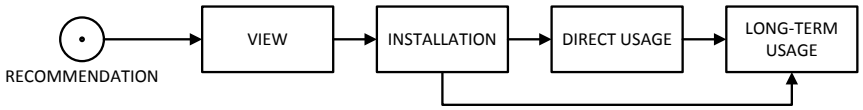


Figure 5.5: The *AppFunnel*'s stages along a user's application interaction sequence.

(2) We present results of a case study of *AppFunnel*. This shows capabilities for ascertaining the performance of different recommender engines by applying metrics to the *AppFunnel*.

Further, we will discuss findings showing that a context-aware recommender engine performs better when aimed at direct usage, and a non-context-aware engine performs better when addressing long-term usage of applications.

5.4.1 Concept of AppFunnel

The idea of the *AppFunnel* is to draw conclusions on the relevance and user experience of a recommended application from his observed interactions with the application, which is based on the concept of the *AppSensor* described in Chapter 3. *AppFunnel* aims to provide a quantitative feedback on the performance of a recommender engine for evaluation purposes. To follow, we describe stages of mobile application engagement that we are able to observe, as well as describing different metrics that can be used to measure the recommendation quality.

Stages of Application Engagement

To ascertain an application's usefulness to a user, we adopted the concept of conversion funnels [11]. A conversion funnel basically describes the ratio of people who reach a follow-up stage in a step-based process. The most comprehensive conversion for mobile application recommendations represents the conversion from a recommended application to one that is used in the long-term. In order to reach this conversion, users go through several stages that represent different events resulting from their interactions with recommended applications. As shown in Figure 5.5, we can distinguish four different stages in a user's action sequence for determining a conversion funnel for a mobile application after it has been recommended.

After the recommendation of a list of applications to the user, the first stage that an application can reach is the *view stage*. Reaching this stage is related to the

event of a user clicking on a specific application that has appeared in the list of recommended applications. This stage reflects the lowest interest that a user might have in an recommended application; it can be drummed up, for instance, by the name, the icon, other users' ratings, or the category of the application (depending on what the system shows to the user at this point). By retrieving application details, the user can view additional information like an application's description, screenshots, or other users' comments.

If a user downloads an application from the application store and installs it to his device, then the *installation stage* is reached. From reaching this stage it can be concluded that the user has an even stronger interest in the application. In addition, an application reaching this stage tells us that the user finds its description and the comments that other users have made on the application interesting enough to download and text the application. For some applications, this also means that the user is willing to pay for the application.

Directly after the installation of an application, the user might reach the stage of *direct usage*. This means that a user launches the application shortly after it has been installed, which indicates a further increase in the interest level, i.e. he is interested enough to actually try it. If the user needs the application in his current context and if the application appears to be useful in the actual context, this might be an additional motivation to directly use the application. As such, this event is not necessarily critical and might not appear for applications that are not launched right away. Instead, the user might wander away from the application and try it later.

Finally, an application might reach the final stage of *long-term usage*. This stage denotes the final funnel conversion and it indicates that an application has reached the highest possible level of a user's interest: It shows that a recommended application turned out to be sustainably useful and engaging in the long term. In comparison to the previous stages, this one can only be determined from post-processing of historical application usage data, and not from observing a single event resulting from user's application interaction. As we will show in the next section, an application's relative usage time can be used to determine a threshold for considering application usage to be long-term usage.

The four stages in this funnel can only be reached sequentially following the order indicated by the arrows shown in Figure 5.5. Every stage can be traversed several

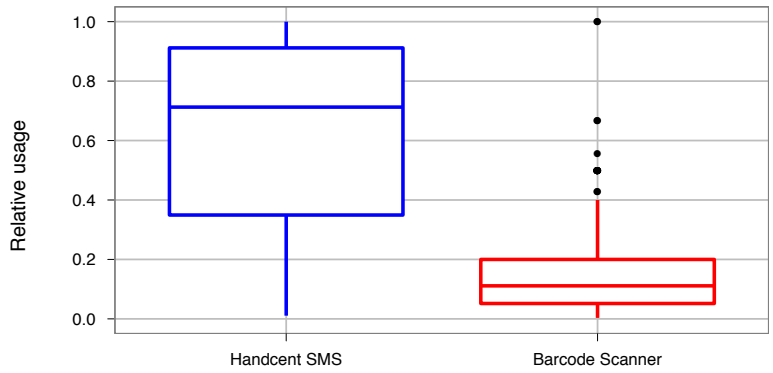


Figure 5.6: Relative usage of two applications (paired data of 197 users).

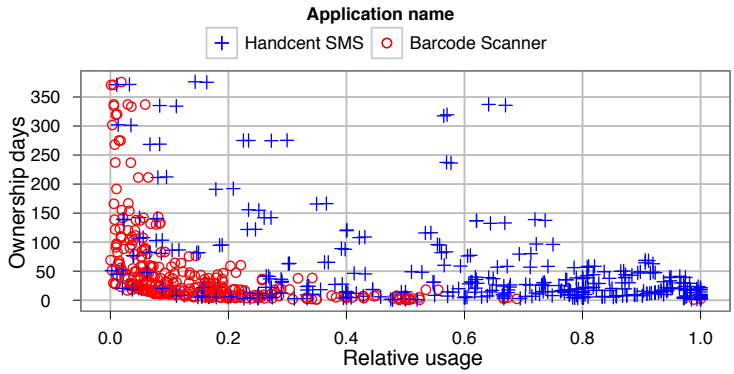


Figure 5.7: Relative usage versus number of ownership days of two applications (paired data of 197 users).

times, always starting from *view stage*, if a user has uninstalled an application and retrieves a recommendation for this particular application again.

Determining Long-term Usage

For determining the long-term usage of an application, we calculate its relative usage $r = u/t \in [0,1]$, with u denoting the number of days during which the application was used and t denoting the total number of days that the application was installed. The granularity of days is reasonable, since the quotient r for relative usage should address long-term usage and the question is whether a user keeps using an application or not. Aggregating based on days additionally compensates for the effect that application usage varies over the course of a day, as described in Chapter 3.

The relative usage is influenced by an application's type, meaning that some applications naturally have a higher and others a lower relative usage; e.g. communication applications are used more often than applications providing functionality that is solely useful in particular situations (as discussed in Chapter 3), like shopping on the weekend. Based on the data provided through the deployment of the recommender system as described in Section 5.3 and Chapter 3, this can be illustrated with the two applications *Handcent SMS*[7] and *Barcode Scanner*[8]. Figure 5.6 depicts the relative usage of these two applications across a group of 197 users. The lower and higher borders of the boxes show the lower and higher quartile respectively. We only considered paired cases, i.e. such users who had both applications installed. The plot shows a significant difference (*Wilcoxon Signed-rank test, W=16510, Z=11.77, p<.001, r = 0.59*) in the relative usage of these two applications, with a high relative usage for *Handcent SMS* and a low relative usage for *Barcode Scanner*. The median relative usage for *Handcent SMS* is about 71%, denoting that people use this application on nearly 5 days a week; the median relative usage for *Barcode Scanner* is only 11%, meaning that people use it less than one day per week.

If a user removes an application from his device, it may not be automatically concluded that the application was of no value to the user, in particular if the application was installed on the device for a long duration. As such, removals of applications require additional analysis of the length of the time periods the applications have been kept on the device by the users, and we can also compare ownership times between different users. For instance, if an application gets uninstalled after a certain ownership time that is longer that the median ownership time of all other users for this particular application, then the relative usage might potentially indicate a long-term engagement even though the application as removed. Further, taking into account these aspects will also allow us to find such cases of users not finding applications beneficial and uninstalling them after short periods of ownership times. For instance cases where all events of installing an application, using it, and uninstalling it appear within the same day, would result in a maximum relative usage of 1.0, although this should obviously not be counted as a long-term usage of the application. For our two example applications, such cases of relative usages are shown in the bottom right corner of Figure 5.7 for people who used *Handcent*

[7]Handcent SMS is a popular free messaging application for smartphones, see http://goo.gl/5dwy1, last accessed on 15.07.2013.

[8]Barcode scanner enables to scan barcodes and decode information such as URLs or contact information, see http://goo.gl/WNrpK, last accessed on 15.07.2013.

SMS for only a short time. To summarize, there are cases in which high values of relative usage do not indicate an original long-term usage and, thus, should be ignored. For instance, if the ownership time of a particular application is long for the majority of users, it can be assumed that a high relative usage resulting from a user who only had the application only installed for a short time can be ignored, as they might have just installed and tried the application before removing it.

5.4.2 Case Study of AppFunnel

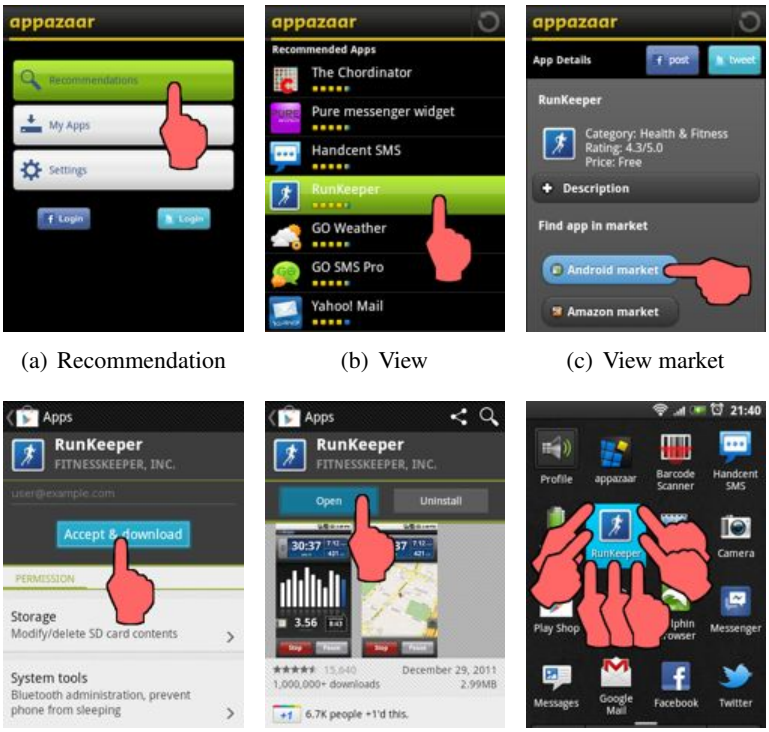
We conducted a case study to examine the validity of our approach and to explore possible conclusions that can be drawn based on data collected from the *AppFunnel* framework. To do this, we implemented the proposed framework within a deployed application recommender system and put it to the test in the wild.

Testbed

We used the *appazaar* recommender system described in Section 5.3 as our testbed. At the time of our analysis the system had been installed by 6,680 users[9]. We do not know any demographic information about our users, but since our testbed *appazaar* is deployed on the *Google Play Market*, we may assume that those people who have installed the *appazaar* application are a good representation of those people who can be expected to use a mobile application recommender system.

In our testbed we had to add a *view market stage* between the *view stage* and the *installation stage*. This is specific to *appazaar* since it builds upon but is not integrated into the *Google Play Market* itself. The only way to recommend applications for installation on Android devices is to forward the user to the *Google Play Market*, e.g., because he has to grant the application's permission request. Figure 5.8 sketches the screens a user traverses from the view stage to long-term usage. The *view market stage* is reached when a user decides to review details about an application, in particular he can have a look at other users' comments on the application, and through an additional button he has the option to install the application. Since the user intentionally has to perform an additional click for installing the application, reaching this stage indicates a stronger interest towards the application since the user wants to give it a try.

[9] According to Google Play's Android Developer Console.



(a) Recommendation (b) View (c) View market

(d) Installation (e) Direct usage (f) Long-term usage

Figure 5.8: Example screens of a user traversing the stages of our testbed's conversion funnel. This usage-centric action sequence can be captured for evaluation of the different recommender engines.

Applying Metrics to the AppFunnel

We extended the testbed to collect data about the given recommendations and the corresponding action sequences taken by users as determined by our conversion funnel. As such, for all given application recommendations we kept track of which stages the recommended applications reached (view / view market / installation / direct usage / long-term usage).

In analogy to well-known and widely-used metrics like click-through rates or pay-per-click, we can define metrics for recommended applications reaching the different stages and conversions between those stages. In our case study we have chosen metrics for the different stages as counters for how many applications per user have reached a certain stage, e.g. the long-term usage metric is the number of applications that have been used in the long-term by different users. Conversion rates are the quotients of two counters of stage events, e.g. the ratio of the number of people

	Non-personalized	***Personalized***
Context-less	– Application popularity	– Usage-based CF
Context-aware	– Application-aware filtering	– Location-aware CF – Time-aware CF

Table 5.1: Recommender engines put under test.

who viewed a recommended application to the number of people who used it in the long term.

Testing Recommender Engines

The focus of this section is on the *AppFunnel* as an evaluation framework and not on investigating new algorithms for new engines themselves. Thus, we have used the recommender engines that already have been developed and deployed within our testbed for evaluating our framework. As shown in Table 5.1, the engines under test can be classified according to the two dimensions of personalization and context-awareness. The personalized engines take information about specific users into account, i.e. application usage history. The context-aware recommender engines generate recommendations based on users' current context, e.g. time, location, previously used applications.

Within *appazaar*, users' requests for application recommendations were scheduled to the different recommender engines randomly. This allows for counter-balanced within-subject testing of the engines. All engines that utilize collaborative filtering are implemented based on the *Apache Mahout* framework.[10]

It is worth mentioning that not every recommender engine was able to produce results for every request. The system suffers from the cold-start problem [216], in particular for the context-aware engines, as described earlier when exploring the design space. When no recommendation could be presented to the user, no user interaction with the recommendation list was possible, and thus no *AppFunnel*-related data was collected. However, to present applications to the user regardless, the fallback strategy for all engines is to return a random set of applications. Since the interaction with such random data does not contribute to the evaluation of the engines itself, we do not take it into account. The five engines that we used in this case study to evaluate our approach, and their characteristics, are as follows.

[10] Apache Mahout: *Scalable machine learning and data mining*, http://mahout.apache.org, last accessed 07.06.2013.

- **Application popularity:** The first recommender engine implemented in *appazaar* is based on applications' popularities measured by their usage. Since installations alone do not reveal much about the quality of an application[11], this engine ranks applications according to their popularity based on their global usage in terms of total number of launches. This engine is neither personalized nor context-aware. Therefore, this engine was able to return results for all issued requests for recommendations, since it did not require any specific additional data.
- **Usage-based collaborative filtering:** The second engine implements an approach based on collaborative filtering. It utilizes users' application usage data as implicit feedback encoded as binary values for user/item pairs: 1 meaning a user has used an application and 0 meaning a user has never used an application. This engine is personalized in terms of modeling the user herself, but not context-aware. Since we implemented this recommender engine using a user-based collaborative filtering algorithm, the cold-start problem is a threat to this engine. It is not able to recommend applications to those users who did not submit any data to our servers (either because they are new to the system or have declined to do so[12]).
- **Application-aware filtering:** This recommender engine is based on the idea that people's usage sessions between screen-on and screen-off have a specific contextual scope. For instance, there is an increased likelihood that people stay with games once they have used a game, or with social applications once they have used a social application [38]. For application usage prediction the precedent application is a also strong predictor (see Chapter 3 and [229]). Therefore, this application-aware engine recommends applications that are similar to those that were used recently within the same session. The application-aware filtering is a context-aware but non-personalized approach. Since this engine is based on the applications recently used by the user, this engine failed whenever the user asked for recommendations without using an application other than *appazaar* previously in the same session.
- **Time-aware collaborative filtering:** The types of applications that people use change during the course of the day as we found in Chapter 3, e.g. news applications are more likely to be launched in the morning, and multimedia

[11] Jakob Nielsen: *iPhone Apps Need Low Starting Hurdles*, http://goo.gl/KR09G, last accessed 07.06.2013.

[12] Users were able to opt out from uploading data, since we wanted to give them control over their data for privacy reasons.

applications are more likely to be launched in the evening [38]. To exploit this phenomenon we implemented a recommender engine that incorporates the time of the day (in hours) as additional piece of context information. We extended the previously described collaborative filtering engine following a context-splitting approach [16, 55] and split the users according to the hour of the day. The user-at-time/item pairs denote whether a user has used an application at a specific time of the day or not. In the same way as the two other collaborative-filtering based recommender engines suffer from the cold-start problem also this engine does. However, the possible contextual values of this recommender engine are bound to a set of discrete values representing the hours of the day.

- **Location-aware collaborative filtering:** Location has a high impact on mobile information needs [137] and application usage patterns [251]. Therefore — analogous to the time-aware engine — we also tested an engine that exploits the location of mobile device usage. This engine is also based on the context-splitting approach, modeling pairs of user-at-location/item denoting whether a user has used a specific application at a certain location or not. This engine is personalized and context-aware. By its design this engine has the same drawbacks like the *usage-based collaborative filtering*, since it uses the same core algorithm. In addition, this engine's location awareness further decreases its success rate, simply because there is a huge variety of different locations given by geographic coordinates. Locations have been discretized into cells of approx. 30m x 30m (to match e.g. sizes of people's home offices, stores). Consequently, the collaborative filtering algorithm of this engine can only provide successful recommendations for those locations where enough user data is available; otherwise it cannot return any well-reasoned recommendations.

For the location-aware engine, the data is split to a higher degree than for the time-aware engine. Since for the time-splitting approach the data only has to be split by the 24 hours in a day, this results in a more dense data set to be used by the collaborative filtering algorithm for a particular hour than for a particular location.

Results

The main purpose of our case study was to examine the capabilities of the *AppFunnel* framework to evaluate different recommender engines by applying metrics. As

determined in Chapter 1 for the goals of this thesis the following analysis does not compare the different engines to find a new best performing algorithm for running recommender engines in this domain. Rather we aim to illustrate the different conclusions that can be made from observations taking the *AppFunnel*-based framework into account.

The data we are analyzing was collected from December 5, 2011, to March 5, 2012, i.e. over a period of three months. In total it contains 287 *AppFunnel* events (204 views, 50 market views, 20 installations, 8 direct usages, 5 long-term usages) contributed by 45 users. The events are distributed across the tested recommender engines as follows:

- 137 for *application-popularity filtering*,
- 47 for *usage-based collaborative filtering*,
- 35 for *application-aware filtering*,
- 48 for *time-aware collaborative filtering*,
- and 20 for *location-aware collaborative filtering*.

Within our testbed, the determination of direct usage is performed directly on the users' smartphones. If a user launches a recommended application within five minutes after its installation, this usage is considered to be a direct usage. In order to determine long-term usage of a recommended application, we perform several steps offline, i.e. in a post-processing step on our server. Firstly, we calculate all applications' relative usage across all users who have used it. For each particular application, we then pick the lower quartile of users' relative usage values as a threshold. If a particular user's relative usage is higher than this threshold, we consider her usage of this particular application as a long-term usage. Thereby we filter out those application users who had the application installed but did not use it long enough for the application to be considered as achieving an engaging long-term usage. Secondly, we calculate the median ownership time for the application across all users who have used it. If a particular user's ownership time is less than the median ownership time, this ownership is filtered out from long-term usage. Thereby we cut off those cases where people have a high relative usage but only had the application installed for a very short time, as discussed earlier.

Due to the reasons discussed above, the number of cases in which a recommender engine has returned a list of successfully estimated recommendations might differ between the engines (e.g. due to cold-start problem). Consequently, for a comparison of the recommender engines' performances, the data collected from the

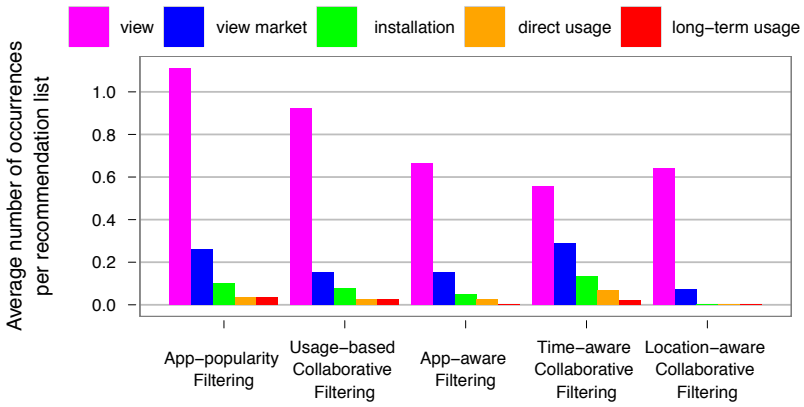


Figure 5.9: Average number of events counted for each recommender engine and each of *AppFunnel*'s stages.

AppFunnel cannot be analyzed based on the raw data. Therefore, we normalized the number of counted *AppFunnel* stages by the number of successful recommendation lists returned per engine. The resulting data represents the average number of events related to recommended applications per successful recommendation list. Since every resulting recommendation list can contain more than one application which people might interact with (see Figure 5.8(b)), the relative number of action sequences per stage can be more than one. This normalization counteracts the fact that not every recommender engine was able to produce a result set for every request. The data is shown in Figure 5.9.

Conversion Stages

The conversion stage metric depicts how many applications have reached a certain stage in the *AppFunnel*. It appears that the *application-popularity based engine* resulted in the most views of recommended applications. Having a relative number bigger than 1 means that on average people have viewed more than one application from a list of applications recommended by this engine. As for all tested recommender engines, there is a large drop-off in numbers for the following stages, as Figure 5.9 shows.

The *usage-based collaborative filtering engine* resulted in the second most views of recommended applications. However, this engine also shows a high drop-off after the view event. This engine successfully generated a great initial impression

with applications that appeared to be valuable for users on a first glimpse, but were discarded after they have taken a look on the application details.

In contrast to the previous two engines, the *application-aware engine* has a lower number of views of recommended applications. In addition, this engine has only resulted in direct usage and did not produce any long-term usage, which can also be a result of users just installing similar applications driven by curiosity. Users did try the installed applications immediately, but did not use them regularly in the long term.

The *time-aware collaborative filtering engine* has resulted in the least view events. However, out of all engines this one has the highest number of action sequences that have reached the three stages of view market, installation and direct usage. Although this engine has recommended applications that resulted in long-term usage, these are fewer than the number of applications that were directly used after they were installed.

The only engine that did not result in any installations or usage events (neither directly nor long-term) is the the *location-aware collaborative filtering engine*. This is interesting, since it has even a slightly higher number of view counts than the time-aware engine. Although the view interests were no less than for the other context-aware engines, there was already a larger gap between the view and the market stage. This engine produced no installations, and therefore also had no direct or long-term usage.

Conversion Rates

The conversion rate metric denotes the relative number of action sequences that users follow from one stage to the next. As can already be seen in Figure 5.9, the further along the stage within an action sequence, the fever events occur. Therefore, for all transitions a conversion rate naturally is between 0 and 1 (including endpoints). Although it is possible to calculate conversion rates between all successive stages of a conversion funnel, the most interesting ones related to mobile application usage are: *view to installation*, *installation to direct usage*, and *installation to long-term usage*. These conversion rates are shown in Figure 5.10. They are based on the absolute and normalized numbers that also have been taken into account for Figure 5.9.

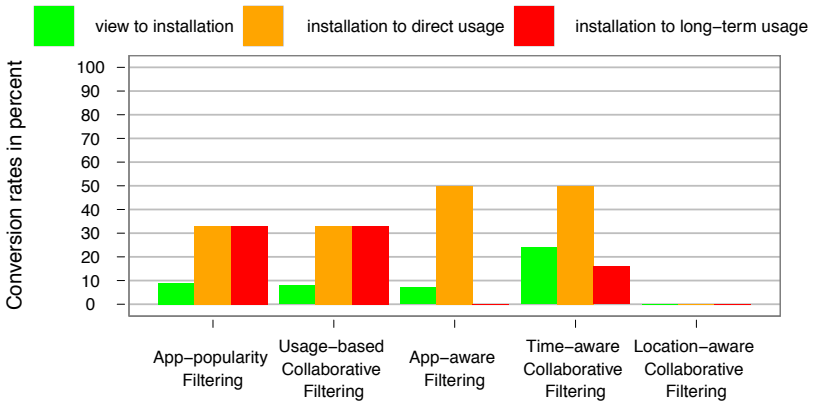


Figure 5.10: Conversion rates of tested engines. The location-aware CF did not result in any of these conversions (see Figure 5.9).

From the view stage to installation the engine based on *application-popularity* has a conversion rate of 9%. Overall, for all engines few people installed applications that they viewed. In general, the conversion rates from installations to usages — either direct or in the long term — are higher. For the application-popularity engine both conversion rates (*installation to direct usage* and *installation to long-term usage*) are above 30%.

The *usage-based collaborative filtering* engine has a conversion rate of 8% from views to installations. The conversion rates from *installation to direct usage* and *installation to long-term usage* are both higher than 30% — the same as the application-popularity based engine. For these two recommender engines it can be concluded that a lot of users who found a recommended application interesting enough to download it also have tested those applications directly after the download, and also did not remove these appliations and rather used them in the long term.

The *application-aware filtering* engine has a 7% *view to installation* conversion rate which is almost equal to that of the *application-popularity* and *usage-based collaborative filtering* engines. This engine did not result in any *long-term usage* stages; therefore, its *installation to long-term usage* conversion rate is 0%. In contrast, the *installation to direct usage* is 50%. It can be inferred that applications recommended by this engine provide a specific functionality, which is useful at the current moment but is not needed on a regular basis.

With a rate of 24% the engine based on *time-aware usage collaborative filtering* has the highest conversion from *view to installation*. Further, its *installation to long-term usage* conversion rate is 16%. Similarly to the *application-aware filtering* engine, it has a high *installation to direct usage* conversion rate of 50%. Since this engine recommends applications that are typically used at the same time of day in which the recommendation request is sent, it can be expected to have a high *installation to direct usage* conversion rate.

Since there have been no installations for applications that were recommended by the *location-aware recommender engine* (see Figure 5.9), there are no conversion rates for this engine.

At the bottom line it can be seen that the two context-less recommendation engines resulted in more views of recommended applications than the others. However, the preliminary results of our case study suggest that the two context-aware engines resulted in highest conversion rates from installation to direct usage, while the two context-less engines resulted in highest conversion from installation to long-term usage.

5.4.3 Discussion of the Case Study

The case study presented in the previous section is tied to our testbed as well as to the recommender engines we put under test. In this section we will discuss the *AppFunnel* in the light of the results achieved for the tested engines, as well as how the framework contributed in this work can be applied to the evaluation of other recommender engines, and how metrics can be defined.

Recommender Engines Tested

It is not surprising that the recommender engine based on applications' popularities resulted in the most views. It is known that naive non-personalized approaches can compete with well-elaborated algorithms [73]. While naive approaches are much easier to design and implement, the more elaborated recommender engines used in this study would require more testing and fine-tuning, which is not the focus of this work.

The high number of views of the application-aware filtering engine might be a result of this engine's design, since it recommends applications which have a high

similarity to applications which have just been used by the user within the same session. Such similar applications very often contain similar keywords in their names or have similar icon symbols. Thus, the users might be driven by curiosity to find out more about these new applications.

The bad performance of the location-aware engine might be a result of the same factor that causes the low success rate of this engine: Locations defined by geographical coordinates might result in too-fragmented usage data that is too sparse and therefore not suitable for a valuable collaborative filtering, which follows from the splitting approach. Optimization of this particular engine is subject to future work. However, the *AppFunnel* framework allowed us to trace down this issue in the first place. If we had taken a conventional evaluation approach, e.g. solely based on counts of downloads or installations, the location-aware engine would have been seen as having better performance than the time-aware engine. Only the usage-centric *AppFunnel*-based approach enabled us to investigate this issue by keeping track of user actions that follow on views of a suggested application. Also, only this approach enabled us to discover that while the location-aware engine results in clicks, it did not result in user value in terms of direct usage or long-term usage.

From its design it was expected that the time-aware engine would result in a high relative number of applications used directly, because this engine recommends such applications which are typically used in the same hour of the day that the user asked for recommendations. This can be concluded from the user behavior we found in Chapter 3.

Two additional events that can be tracked along a mobile application's lifecycle are updates and uninstalls of an application [38, 103]. However, the update does not tell us much about a user's engagement with the application, since frequency of updates is specific to each particular application and its developer, and updates can be done automatically without any user intervention. Additionally, we do not consider the removal of an application as feedback. This can also be exploited, but requires further refinements and post-processing, since the removal cannot be taken as an additional stage in the *AppFunnel* as such. Firstly, the removal is of a negative rather than positive feedback, and the *AppFunnel* currently is based on positively growing engagement from stage to stage. Secondly, not every user necessarily removes an application he does not require anymore (e.g., he can also

remove only the icon of the application from his home screen, cf. Chapter 4). In contrast, the *AppFunnel*'s stages necessarily follow from the user's intent.

Application of AppFunnel

It was interesting to see that — irrespective of the location-aware engine — the two context-aware engines caused different events within the *AppFunnel* than the two context-less engines. The results of our case study suggest that the performance of the different paradigms of recommender approaches shows more diversity in terms of stages reached after a recommended item was clicked on, i.e. *installations* and *direct usages* or *long-term usages* are more diverse than *views*, especially in terms or conversion rates. This suggests that the usage-centric evaluation approach — as introduced by *AppFunnel* — enables a more elaborated evaluation of recommender engines that suggest mobile applications and their design goals, e.g., fostering download and installation counts, or providing instant support in the form of applications that can be used directly, or pursuing a user benefit in terms of long-term application engagement.

Choosing from AppFunnel's Metrics

Recommender systems are commonly implemented in e-commerce websites for the sake of increasing a merchant's turnover and revenue. We argue along with Konstan and Riedl [150] that incorporating the user experience into algorithms will affect the user's benefit from a recommender system, and that a more comprehensive set of measures than those which are currently being used are required. For the recommendation of mobile applications, the *AppFunnel* goes beyond measuring the quality of a recommender engine in terms of installations. In the case of paid applications, an increase in revenue is already reached as soon as the customer installs a recommended application. This, however, only is an intermediate stage that an application can reach in the *AppFunnel*. Especially considering that a large portion of installed applications are lying idle [229] it is reasonable that recommending for installations alone is myopic and should not be a top-priority goal.

Based on the *AppFunnel* we can reason about the quality of a recommender engine that suggests mobile applications beyond commonly applied metrics, like *click-through rate* and *pay-per-click events*. However, we suggest considering the metrics we introduced as additional metrics for evaluation of recommender engines rather than a replacement. Our metrics aim to extend evaluation from a vendor's

view to a usage-centric view of recommender system performance — i.e., which application was of actual use for a user in her current context or in the long term.

As we have seen in the presented case study, one metric might conflict with others. Hence, a single recommender engine might not have the best performance best in all metrics. How to chose and ascertain the metrics should be decided according to a recommender engine's design goal. As seen in our case study, a context-aware engine might achieve good results when looking into direct usage, but have bad results when taking into account long-term usage. This is sufficient for a context-aware recommender engine, since its goal is to address the actual user needs, while a recommender engine generally modeling user taste should aim for higher performance according to long-term usage.

We did not take into account applications' prices. The price of an application might impact whether it will be installed or not, but this is an attribute of the application rather than being inherent to the application's usage. Prices can be taken into account when designing new recommender engines. Applying the *AppFunnel* approach, one might find that an engine recommending expensive apps to a user who only installs free apps might result in many view events but perform badly from the standpoint of view-to-installation conversion rates.

Our case study shows that by applying the *AppFunnel* and incorporating the metrics we introduced, one can investigate the performance of recommender engines for different goals related to user experience. In our case we realized that an engine with average performance in terms of views performed poorly in terms of usage.

Usage-centric Evaluation in Other Domains

Other researchers and practitioners can benefit from our work by applying the metrics to mobile application recommendation or other domains, where one can track the use of recommended items beyond click-rates or billing statistics and re-using items is reasonable, e.g. e-books. For instance, since electronic music players are able to trace whether people are actually playing recommended songs, similar concepts for collecting feedback can be built.

We believe that a deployment-based approach of incrementally developing new recommendation paradigms alongside with emerging fields where no data is available for evaluation, alongside with developing the system itself, is a reasonable approach. The underlying scientific method has been described in Chapter 2.

5.5 Summary

In this chapter, we investigated the challenge of discovering mobile applications in a growing mobile ecosystem providing an increasing number of mobile applications. We presented a design space for mobile application recommender systems, described a system we implemented and deployed on the Android platform, and presented a usage-centric evaluation framework for assessing the performance of algorithms in recommending mobile applications.

The current chapter described the need for context-aware mobile application recommendation and explored the design space for recommender systems within this domain. The novelty of investigating mobile applications as items for recommender systems is the ability to draw conclusions on the contextual relevance of an application by observing signals like the usage time and the usage frequency. We argued that mobile application recommendation can benefit from integrating context into recommender systems, since it strongly influences a mobile user's needs. We explored the design space for context-aware recommender systems for mobile applications with a focus on parameter capture and recommendation access. Different options for implicit or explicit capture of information about the user's identity, the applications, the contexts and the relevance values have been described. A brief discussion explained the advantage of an implicit (as opposed to explicit) parameter capturing. We presented a prototype implementation of a recommender system for mobile applications called *appazaar*. This system recommends applications to mobile device users based on the actual usage of the applications as a relevance measure related to different contexts.

Finally in this chapter, we presented *AppFunnel*: a framework for the usage-centric evaluation of recommender systems that suggest mobile applications. The main part of this framework is the usage-centric concept of a conversion funnel for tracking mobile application engagement: from viewing a recommended application, to installing it, to using it in the short term, to using it in the long term. The *AppFunnel* allows for evaluating recommender engines beyond click-through rates and download statistics. We have implemented and tested our framework based on a deployed system in the wild, and presented results of applying metrics to the *AppFunnel*. Although the testbed's recommender engines are simplistic, we were able to find that the ones that address the current context of the user show a higher performance in direct usage, while non-contextualized recommender engines result in a higher long-term usage of installed applications.

We were first to propose the approach of context-aware recommendation for the field of recommending mobile applications by presenting the design space [32, 42] that we describe in Section 5.2 in this work.

Chapter 6

Multitasking Between Mobile Applications

While in the previous chapters we investigated patterns of application launching (Chapter 3), how people organize their applications (Chapter 4), and how we can help them better discover new applications (Chapter 5), in this chapter we will look into the interplay and inferences between multiple applications on single smartphones and its impact on user experience.

The results of this chapter have been presented in two publications [37, 157]. Work on multitasking and task interruptions that relates to the studies and investigations conducted in this chapter appears in Chapter 2, Section 2.3.5.

6.1 Introduction

As described in Chapter 1, mobile phones have evolved from solely communication-oriented devices for phone calls to multifunctional toolsets, and people use smartphones for various other tasks beyond communication. The increasing integration of various functionality into one piece of hardware comes at a cost that we will study in this chapter. Since people's tasks often change — either intentionally because they start doing something different, or unintentionally by being interrupted — they might also switch the application they are using on their smartphones from one moment to the next to gain better support for their current activity.

In this chapter, we first we investigate people's multitasking behavior between applications on smartphones. We describe our analysis of a large-scale dataset of mobile application usage that in particular provides insights into application switching behavior. We will draw special attention to cases where the usage of applications on the smartphone is being interrupted by incoming phone calls. From this study we will learn that the costs of phone call interruptions can be exceedingly high. In light of this study, we secondly will revisit the design of user interfaces for mobile phone call applications. We argue that the evolution of smartphones' phone call user interfaces has not kept pace with the evolution of mobile phones into multifunctional devices, and we will present different new designs to mitigate the problem of call interruptions.

6.2 The Phenomenon of Application Interruptions

Given the growing number of tasks that can be supported by smartphone applications, and the growing amount of time that people spend using applications on their smartphones — as described in Chapter 1 — it follows that the probability of people being interrupted by a call while using applications on their smartphones will increase. Such interruptions of application usage might happen either intentionally, i.e., at the user's own will, or unintentionally, i.e., not intended by the user, under various circumstances. Jin and Dabbish [134] refer to such intended switches from one application to another prior to a task's completion as *"discretionary task interleaving"*. Studies of unintended interruptions, like external and social interruptions from messaging and notifications, contribute a large body of related research (see Chapter 2). However, the body of literature on mobile interruptions is relatively small (see Chapter 2). Examples of mobile interruptions are for instance that one might be writing an e-mail and then switching to a game before the writing was finished (intended interruption), or one can be searching for a location on a map while a phone call comes in that one has to answer (unintended interruption).

In this section, we analyze the phenomenon of application interruptions on smartphones in the light of two different cases, as noted above:

- We look into users' intended back-and-forth switching between applications.
- We investigate unintended interruptions triggered by incoming phone calls.

Although task interruptions are an inherent problem in smartphone usage — as sketched previously — little is known about this phenomenon at the fine-grained level of single application usage, which we made accessible by developing the *AppSensor* described in Chapter 3. Since the low-level large-scale analysis of mobile application interruptions was not yet developed, essential questions regarding to mobile application interruptions remain open. For instance, how often do such interruptions happen? What is the impact of mobile interruptions on a user's task performance? And in particular, at what costs do we allow interruptions on smartphones?

Most related previous works have studied task interruption on stationary computers (cf. Chapter 2, esp. [25, 128, 214]), and — due to its high relevance to safety — a large body of research looks into call multitasking while driving (cf. [158, 238, 237, 129, 47, 130, 48]). In this work, we reach out to study the phenomenon of multitasking on smartphones and go beyond stationary computers. In particular, we argue along with Karlson et al. [142] that a number of challenges that might derail fluent task flows on mobile devices do not exist on stationary computers, and that from the perspective of interaction, being mobile is cognitively costly and mobile interactions come in short bursts [181]. Applications are designed to provide one particular functionality for a task, and they are typically used in a fast-paced mode, which increases the demand for efficient multitasking.

In this section, we present a conceptual framework for investigating mobile application interruptions on smartphones, which is based on the *AppSensor* we introduced in Chapter 3. This framework can help researchers to analyze application interruptions on mobile phones. Further, we analyze a large-scale dataset and present results of an in-the-wild observational study. When discussing application chains in Chapter 3, we provided an example for a mobile task of a user switching between a few applications that helped him search for a place to get a coffee. By contrast, in this section, we concretely look into the cost of these interruptions on the completion time of tasks. Therefore, in the context of this work, and acknowledging its limitations, we consider the task completion time as the time spent using a single non-interrupted application that is launched and then closed after some time.

6.2.1 Log-based Study of Application Switching Behavior

For our study, we analyzed the dataset that we collected by means of the *AppSensor* in the wild, which we already presented and partly analyzed in Chapter 3. In this study of application multitasking we use an updated data dump of the deployed system with possibly different users, other applications and from a later timeframe than in Chapter 3.

According to our research approach introduced in Chapter 2 the method that we apply in this study is an natural experiment. The treatment that we observe are the interruptions of application switches and phone calls, which naturally occur in our participants' smartphone usage.

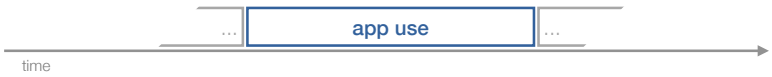
In total, this updated dataset contained around 5,495,815 samples of launched applications. Data comprises in total about 15,756 different applications used for one and a half years (532 days, from August 2010 to January 2012) by 3,611 unique users. As in our study in Chapter 3, those users were spread out geographically and distributed worldwide. The means of recruiting participants and the underlying deployed system that implemented the *AppSensor* for data logging were the same as in Chapter 3.

Analysis of Logging Data

Since *AppSensor* only provides us with the logs of mobile application usage, we require an additional layer for detecting and analyzing interruptions on top of the fine-grained traces of application usage. We developed means to interpret application usage logs and detect interruptions as well as to measure the timing of interruptions and estimate their costs, as we will explain next.

Detecting Interruptions. As Figure 6.1 shows, we consider an application to be interrupted when the foreground activity of a smartphone changes from one application to another, and then returns to the previous one. While Figure 6.1(a) shows the non-interrupted usage of one application, Figure 6.1(b) shows the case when the usage of a particular application is being interrupted by another one. As a result, we consider the application that was used in between to be interruptive on the primary application that the user returns to. More formally — in terms of the *AppSensor* — we consider application a to be interrupted by application b, if

$$as(t) = a \ \wedge \ as(t+1) = b \ \wedge \ as(t+2) = a$$



(a) Non-interrupted case of application usage

(b) Interrupted case of application usage

Figure 6.1: Detection of interruptions of mobile application usage.

and there are no other applications being used between times t and $t+2$. Additionally, we impose the following constraints on application switches to be considered as application interruptions:

- Phone call applications are not considered to be interrupted. Phone calls are considered to be atomic interactions, with each call being a single one.
- Launcher applications are not considered to be interrupted. Launchers do not provide any task-related functionality beyond providing a central menu for launching other applications.
- Launcher applications are not considered to be interruptive. If the user switches to a launcher and then returns to an application, there has not been any use of any other functionality related to another task in between.

Imposing these constraints on our analysis of application traces leaves the two possible cases for application interruptions, which we aimed for earlier: Applications can either be be deferred, i.e., interrupted on purpose by the user, or interrupted by an incoming phone call. We were able to distinguish incoming from outgoing phone calls based on whether a dialer or phone book application was used before, which indicates an outgoing call since the user dialed a number. From here on, we will refer to the former case as *application switching* and to the latter case as *call interruptions*.

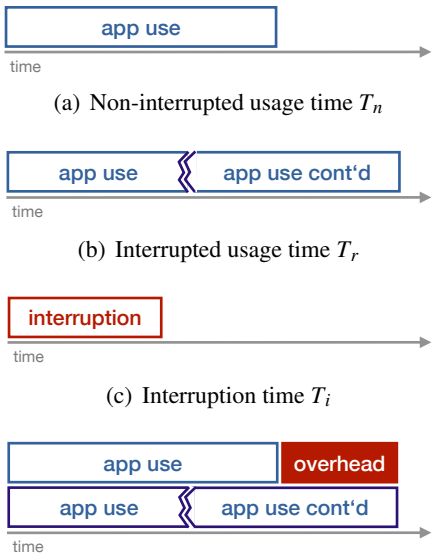
Please note that applications might be deferred for a longer time due to external environmental events that are not directly related to a user's smartphone usage, in which case the device might change to stand-by mode. For instance, a user might be prompted by a passer-by to chat for a while. To avoid misleading results in our analysis, we use a time window to handle such cases. Since the average application usage length per launch is about 1 minute (cf. Chapter 3 and [38, 250]), we have chosen this specific duration for the time window as a threshold. That

being said, if a series of consecutive data samples were found for the same user at the same day for a particular application without any other application in between, such application uses were considered as different activities if the time between occurrences of application uses exceeded 1 minute.

Computing Interruption Overheads. Based on the aforementioned detection of application interruptions, we are able to quantify the overhead that is introduced by an interruption of an application. Therefore we introduce the following variables to measure usage times of applications, as sketched in Figure 6.2. Figure 6.2(a) shows the application usage duration for the non-interrupted case, which we denote as T_n. Figure 6.2(b) shows the case of an application use being interrupted by use of another intermediate application. The usage duration of the primary application T_r in this case is decomposed into two parts, with T_a being the duration before and T_b being the duration after the interruption, with $T_r = T_a + T_b$. As Figure 6.2(c) shows, we can also measure the time of the interruption itself denoted as T_i, which essentially is the usage time of another application. Finally, we are interested in finding the overhead that an interruption introduces by comparing the non-interrupted usage duration T_n to the interrupted usage duration T_r. As Figure 6.2(d) suggests, this overhead time can be estimated as $T_o = T_r - T_n$. As such, T_o is a measure of the extra time that might be introduced by the interruption time T_i. In literature, such overheads are also cited as *"resumption lags"* [128, 214], and usually they lead to a decreased performance in the primary task [12], which in our work is the use of an application.

It is worth noting that those variables that denote the usage time of applications and duration of the interruption are naturally greater than 0 (i.e., $\{T_a, T_b, T_i, T_n\} \in \mathbb{R}_0^+$), while the estimated overhead might also be negative (i.e., $T_o \in \mathbb{R}$). The overhead might be negative, e.g., if applications which people usually use for a very long time (e.g., games like *Angry Birds*, cf. Chapter 3) are not resumed after interruptions have happened (e.g., it might not be possible to continue playing, or the user might lose interest). For desktop applications it has been shown that interrupted tasks are not resumed half of the time [162].

For our analysis we only investigated paired cases of application uses, since appropriate overheads can only be computed for those cases. Therefore we only considered those cases where, for a particular application and a given user, our dataset contains both the non-interrupted and the interrupted case of application use. To quantify how often interruptions happen, however, we mined the whole dataset for



(a) Non-interrupted usage time T_n

(b) Interrupted usage time T_r

(c) Interruption time T_i

(d) Estimated overhead time T_o

Figure 6.2: Computing an application overhead when interrupted. In particular we are interested in the overhead caused by an interruption.

	Call interruptions	**Application switches**
Interruption data samples	776,922	970,543
Interrupted users	1,929 [1,676]	2,926 [2,609]
Interrupted applications	1,373 [487]	4,626 [1,043]

Table 6.1: Summary of dataset used for study of application interruptions. Numbers of balanced (paired) cases are given in brackets.

interrupted cases to get a more comprehensive understanding. To analyze the log-based dataset provided through the *AppSensor*, we grouped the application usage traces based on samples' timestamps per day and per user. Table 6.1 provides an overview of the scale of our dataset and in particular of all the cases we found. As stated, we removed unpaired cases from our further analysis; final numbers are given in square brackets in Table 6.1.

6.2.2 Results

Table 6.2 and Table 6.3 show descriptive statistics on the cases of call interruptions and application switches that appeared for users in our data set. We macro-averaged our measures, since we wanted to give equal weight to each user and

	Call interruptions	Application switches
Daily interruptions (% usage)	3.2 (2.2)	8.3 (5.3)
Interrupted applications	3.3 (2.6)	8.7 (7.2)

Table 6.2: Occurrences of interruptions. Mean (and SD in parentheses) per user on usage in terms of application launches.

	Call interruptions	Application switches
Regular application runtime (s)	24.8 (31.8)	18.9 (24.4)
Runtime when interrupted (s)	107.1 (121.1)	87.9 (75.5)
Interruption duration (s)	12.5 (8.1)	23.7 (19.3)
Overhead duration (s)	43.2 (65.9)	34.4 (40.7)

Table 6.3: Costs of interruptions. Mean (and SD) per application in seconds.

their application usage, because the amount of data contributed per user differs; i.e., we averaged individually for every single user, and then averaged over those user-based means. Outliers were considered when the mean exceeded 1.5 the interquartile range which were automatically removed when computing the statistics.

Note that we are reporting the results for call interruptions and application switches after removing the outliers and unpaired cases from our data here (the size of the cleaned data set is given in square brackets in Table 6.1). A Shapiro-Wilk test revealed that our data is not normally distributed ($p < .0001$ in all cases). Therefore we used the Kolmogorov-Smirnov test, which is non-parametric and does not require the underlying data to be normally distributed[1].

Unsurprisingly, we found that interruptions related to application switches are more frequent ($D^+ = 0.50, p < .0001, Cohen's\ d = 1.24$) and involve more applications ($D^+ = 0.36, p < .0001, d = 0.98$) than interruptions caused by incoming phone calls. Table 6.2 shows that only 3.2% of daily application usages (in terms of launches) are interrupted by incoming phone calls, whereas for 8.3% of application usages an interrupting switch happens. These differences were found to be statistically significant. This relates to the finding about application switches within chains of application usage we presented in Chapter 3: Users tend to switch back and forth between applications they already have opened within the same session.

Besides the occurrences of mobile application interruptions, we also looked into their costs in terms of introduced overhead as resumption lag, based on the data

[1]Kolmogorov's D statistic refers to two-tailed comparisons, while D^+ and D^- refer to one-tailed comparisons.

(a) Correlations of occurrences

	n_i	n_a
n_i	–	0.26^{***}
n_a	0.27^{***}	–

(b) Correlations of time measures

	T_r	T_i	T_o
T_r	–	0.12^{**}	0.68^{***}
T_i	0.09^{*}	–	0.13^{**}
T_o	0.51^{***}	0.22^{***}	–

Table 6.4: Correlations of interruption measures. Tables show correlation coefficients ρ for phone call interruptions above the main diagonal, and for application switching below the main diagonal (statistical significance: $^{*}p < .05$, $^{**}p < .01$, $^{***}p < .001$).

summarized in Table 6.3. Overall, we found that those applications interrupted by phone calls take more time to complete than those interrupted by application switches ($D+ = 0.06, p = .04, d = 0.18$). Notice the differences in application runtime when compared to its normal usage ($D = 0.09, p = .003, d = 0.20$): When an application is interrupted — either by internal or external interruptions — the runtime of the application is delayed up to a factor of 4. Most importantly, we found that interruptions caused by phone calls result in a significantly higher overhead time on the interrupted application than the user's intentional application switching ($D^{-} = 0.08, p = .006, d = 0.17$), with overhead in the case of phone call interruptions being 43.2 seconds and in the case of application switches being 34.4 seconds. These results suggest that interruptions caused by incoming phone calls are more disruptive with respect to their resumption lag than interruptions due to application switching.

We also investigated the correlations between the variables we measured, which appear in Table 6.4. In most cases we found them to be weakly or moderately correlated. The strongest correlations were for both types of interruptions — i.e., application switching and incoming phone calls — between (1) the runtime of an interrupted application T_r and its overhead T_o ($\rho > 0.5$) and (2) the number of daily interruptions and the number of interrupted applications ($\rho > 0.2$).

Overall, we found on the one hand that mobile application interruptions at the application level are an infrequent phenomenon: only about 3% of daily application usages are interrupted by calls, and 8% by switching back and forth between applications. On the other hand, we found that if application interruptions do happen, the resumption cost may be tremendously high.

6.2.3 Discussion

Interruptions can have a severe effect on task performance and user experience [12]. Hence, knowing about the phenomenon of mobile application interruptions implies a need to cope with this issue, and to improve future designs of both single applications and smartphone operating systems. In particular, using the introduced framework to assess and measure the impact of interruptions, i.e., the duration of interruptions and the resulting resumption lag, will help to improve the design of system UIs and applications that are built to support multitasking between applications.

A first, rather obvious idea to support better multitasking on smartphones is to help the user to better be able to regain the context of the interrupted application when she comes back. Tossel et al. [245] report that a large fraction of web-site re-visits happen after interruptions by phone calls. In such cases the support of task reconstruction might leverage the navigation paths on the website which are available in the browsers history. On stationary computers, users explain long resumption lags through the loss of the context associated with the task when switching [128]. As such, it has been been found to be helpful for recovering from the interruption to give pertinent cues guiding the user back into her previous task [135, 87] so that she can regain context within the application after an interruption has happened, or after the user switched away from an application for other reasons. However, while this is a reasonable technique for desktops, this is usually not applied on smartphones [142].

From interruptions on stationary computers it is known that people start to intentionally interleave tasks — like switching between applications — when they are waiting for the primary task to finish or to continue [134]. This might also be an explanation for the application switching behavior we found, since many mobile applications require that content is downloaded from the Internet, what might take some time in some cases. Further it is know, that external interruptions might continue to have an effect even after the interruption, as people start to interrupt themselves after they had been interrupted externally [75]. The use of an application that was interrupted by a phone call or any other interruption beyond our scope might also result in higher application switching behavior afterwards.

For desktop computers, approaches for mitigating the impact of interruptions have, for instance, been proposed by Iqbal and Horvitz [128]. For example, reminding the user of the task that she has left unfinished, or assisting her in efficiently recon-

structing task context, are two ways to help. For the mobile domain, we believe that the main factor is to reduce the overhead time that results from application interruptions. In this line of thinking, we suggest helping the user to keep up the task state of the primary application while switching to the interruptive application, or to help the user recover the task she left in the primary application when returning after the interruption. Hereafter, we contribute design recommendations for implementing these ideas for smartphones.

Design Implications

Our findings could inform new mobile interaction designs that aim to reduce the overhead that results from application interruptions. We present two design implications, motivated by solutions proposed in the field of stationary computers (cf. [248]). On the one hand, we propose preventive strategies that are effective before the user switches to the new task, and on the other hand, curative strategies that are effective after the interruption, when the user proceeds with her primary task. We propose the following approaches for implementing such supportive functionality for mobile multitasking:

- **Preventive strategies** are about preparing the user for being interrupted. Indeed, before a task interruption occurs, the smartphone user could be better prepared to leave the current task. For instance, for incoming phone calls the caller usually waits on the line for some seconds. Postponing the call a bit longer (say, 500 ms) might provide time to give the user an auditory, visual, or haptic signal that soon the phone application will pop up. Thus, the user would be able to save a mental state of the current open application and keep his most recent activities in the application in mind before the interruption happens. On stationary computers in a office scenario, this could be to first finish writing the current sentence in a word processor, before answering a ringing phone. Further, with current phone applications users receive a full-screen visual notification of incoming calls. We believe that gradually overlaying this notification onto the user's current primary application would also allow him to take a subconscious snapshot of his most recent action, which might allow him to more easily return to his primary application after the interruption. This particular approach might also work for the interruptions that occur due to application switching.

- **Curative strategies** are about guiding the user as she goes back into the interrupted tasks. The main requirement when the user resumes a previously interrupted application is that she has to reallocate cognitive resources, which becomes increasingly difficult if the resource demands were high [128]. For this reason, the user should be supported when resuming her interrupted application for immediate and easy continuation of her previous task. For instance, this could be achieved by automatically replaying the last *N* milliseconds of UI interactions for better task-reconstruction [213], such as navigation through a website, or replaying the last scroll interaction to a certain position on a long document or text input into a form, to give a hint of what she was doing before the interruption occurred (see e.g. [102] for visual cues on maps). The mobile operating system could also leave a visual on-screen cue such that the user could remember at any time to which task she is switching back, as done on stationary computers [87]. Another way of visually helping the user find her way back to her previous task and continue working in the primary application would be to vanish the screen of the interrupting application in the direction of the last focus point of interaction on the primary application (e.g., last touch point, or text input field), in order to guide the user to the screen position she left when the interruption took place.

Limitations of the Study

The results presented in this section are dependent on the quality of the underlying dataset. The data was collected by a deployment of the *AppSensor*, as described in Chapter 3. As such, the presented study on application multitasking has the same threats to validity and correctness as our study on mobile application launching.

As already stated, there are other disrupting factors beyond smartphone usage itself that lead to people interrupting their application usage and changing tasks [25], e.g., social interruptions by people in a person's current vicinity. Moreover, as described by Karlson et al. [142], people might change to a nearby stationary computer if they are hindered or frustrated from conducting the task on their mobile. Due to our study method of conducting a large-scale study in the wild, we unfortunately could not keep track of these issues and other environmental influences in our data. Our study design and method — in the first place — allow us to report that mobile application interruptions do happen, and to calculate their probability

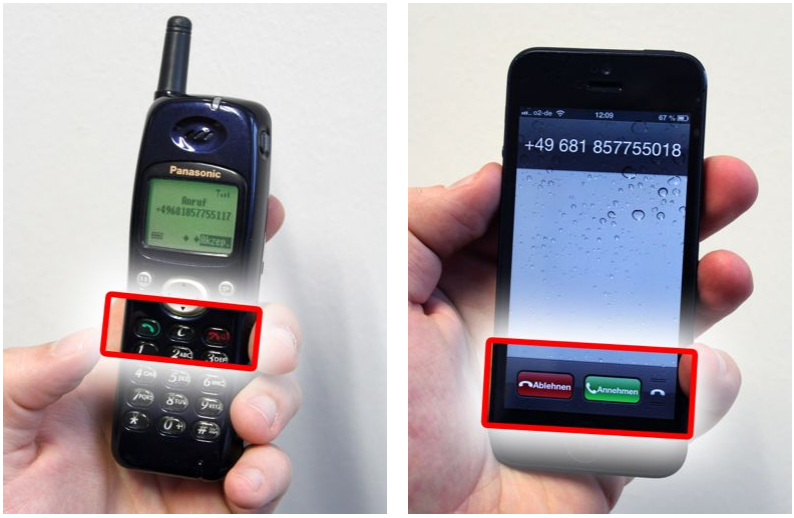
and show that they add significant costs to mobile application usage. However, since our observational study was non-controlled and was based on data analysis, we cannot make any other conclusions. Due to the large scale of our user sample, their worldwide distribution and — presumably — large diversity in backgrounds, we may assume that our study provides a higher external validity than common interruption studies conducted in controlled laboratory settings [116].

Finally — as previously mentioned — we recognize that a one-on-one mapping from tasks onto applications is hard to convey, since a task can span over a single application (e.g., writing an email) or many (e.g., when searching for a restaurant one might look up its address in a browser and then use maps to locate the place). That being said, it remains unclear to which degree the ecological impact of mobile application interruptions — as revealed in our study — would relate to the user's actual cognitive load and higher-level goals. For instance, if a user is interrupted by a phone call while she is preparing a meeting on this smartphone (e.g., checking dates on this calendar and sending emails for inviting attendees), we cannot know if the call creates an overhead on her higher-level task, which relates to the resumption lag that is introduced on the micro-level of application interactions.

6.3 Re-Designing Phone Call Applications

Smartphones became multifunctional devices, which nowadays integrate a huge variety of services beyond pure communication capabilities. In addition to making phone calls, people can for instance use their mobile phones for navigation, mobile payment, making videos, taking photos, or gaming and entertainment. The variety of activities for which people use their smartphones, and the number of installed applications, is steadily increasing (cf. Chapter 1).

The previous section provides evidence that this dense integration of various functionalities comes at a non-negligible cost: People have to multitask between services. They have to switch between applications, which introduces an overhead on the application usage time. As the previous study revealed, this overhead is particularly high when application usage is interrupted by an incoming phone call. The interruption might increase the completion time of the interrupted application by a factor of four. Although we previously found that this happens only rarely (cf. [157] and Section 6.2), the overall smartphone usage time is anticipated to



(a) Panasonic EB-GD90 (b) iPhone 5

Figure 6.3: Only slight evolution of phone call UIs from a) single-purpose to b) multi-purpose mobile phones: Essentially only the red and green hardware buttons have been replaced by touch screen buttons.

increase further.[2] This will also result in an increased probability of smartphone applications being interrupted by incoming phone calls. Hence, rethinking user interfaces for phone call applications becomes indispensable.

Related work (cf. Chapter 2, Section 2.3.5) mainly investigates phone calls interrupting primary tasks not done with the phone, e.g. driving or office work, and only a little (cf. e.g. [5, 46, 96, 157]) is known about task interruptions on smartphones. However, although it is known that phone calls can be interruptive, new designs for call notifications with concurrent mobile application usage have not been investigated yet.

In a typical non-mobile desktop scenario, people are not interrupted abruptly when a call comes in. Since the incoming phone call happens on a separate device, people have a higher degree of freedom for deciding when they allow the interruption to happen, i.e., to switch from the previous task on the computer to the incoming call on the phone. For instance, when somebody is writing an e-mail on a desktop computer and the phone rings, one can first finish the current line of thought and finish writing the current sentence before answering the phone. In a mobile

[2] Pew Research Center (M. Duggan and L. Rainie): *Cell Phone Activities 2012*, http://goo.gl/YoP0p, last accessed on 08.07.2013.

scenario, the user's decision space is very limited, since the interruption happens on the very same device. When a phone call comes in, not only does the phone call application open instantly, but also the user must either accept or decline the call — which compels him to make a decision [107], which is cognitively costly and amplifies the effect of the interruption if inappropriate [96]. Further, the effect of change blindness might make the effect of the interruption even worse if the underlying application updated itself while the interruptive screen was shown [77].

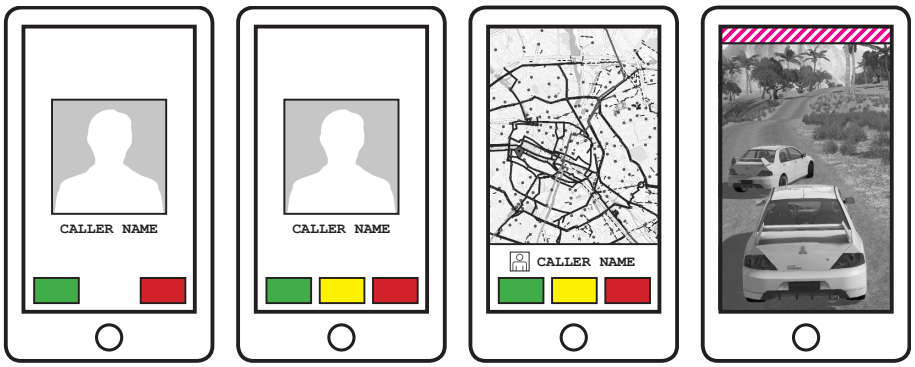
The ability to make mobile phone calls empowers social interactions and makes smartphones social devices in the first place, as we found in Chapter 3 smartphones are most likely used as communication devices at least during the day. However, the current design of phone call applications also has some social implications, as they lock the smartphone when calls come in until the user makes a decision, or the user has to let the phone ring until the caller hangs up or the answering machine takes over (using the home button for multitasking does not work on the latest versions of Android OS and iOS). For instance, if a user is searching for a place on a map and a call comes in, she cannot use the map until the call is over (declined, ended by caller, or accepted and finished). This case of declining a call relates to the category of enforced unavailability due to practical inconvenience [212].

The current design of phone call applications is a remnant of times when the main purpose of mobile phones was communication by making phone calls. A comparison of the two mobile phone call UIs shown in Figure 6.3 suggests that when mobile phones evolved to multi-functional devices, the design of phone call applications essentially remained the same. Apart from the fact that hardware buttons have been replaced by touch screen buttons and additional content can be shown (like pictures, the caller's birthday, or social network information), the user's options for handling incoming phone calls essentially did not change.

In this section, we revisit the design of mobile phone call applications in the context of multipurpose smartphones. We aim to mitigate the negative effect of phone calls interrupting mobile application usage. We explore the design space and suggest to new designs for mobile phone call applications.

6.3.1 Extending Phone Call Applications

While mobile phones have evolved significantly in recent years from single-purpose communication devices to multi-purpose devices, the fundamental design



(a) Current phone apps (b) Added postpone option (c) With multiplexing (d) Background notification

Figure 6.4: Extending the design space for mobile phone call applications from (a) current accept/decline dialogs to (b) accept/decline/postpone dialogs, (c) multiplexing and (d) background notifications

of phone call applications did not evolve accordingly. While its implementation leveraged new hardware and software capabilities, the fundamental decisions people are able to make when they receive a call did not change. Currently, when a call comes in, a modal dialog opens where the callee can either decline or accept the call. In the previous section, we found that the current user interfaces of phone call applications (phone UIs) often lead to increased overhead when application usage is being interrupted by phone calls [157]. In this section, we present work on revisiting phone call UIs for multipurpose smartphones. We contribute a new design space for mobile phone call UIs, going beyond the simple accept-or-decline dilemma.

After analyzing current implementations of phone call applications on some popular smartphone models, we found that they commonly have two shortcomings, which may particularly amplify the effect of interruptions:

1. Current phone call applications use full-screen modal dialogs to notify the user of incoming phone calls. This instantly visually detaches the user from his previous application and thus might lead to a high impact from call interruptions, as visual interruptions are known to have a severe impact on small screens [77].
2. Current phone call applications only provide the user with two options: to promptly either accept or decline the incoming phone call. This unavoidable required decision (accept vs. decline) may amplify the effect of an inter-

ruption, as it is known that inappropriate interruptions significantly delay primary tasks [96]. Further, accepting the call pulls the user's attention further away from the previous application to the phone call, and declining may have additional negative side effects.

Figure 6.4(a) sketches the design of currently predominant phone applications. So far, the most-used design for phone call applications only provides options for either accepting or declining the incoming phone call. Some implementations provide additional options, e.g. shortcuts to sending messages like *"I am currently in a meeting"* to the caller when declining the call, or augmenting the call with additional information like the caller's profile picture or birthday.

As found in the study presented in the previous section (cf. also [157]), this particular design for call applications results in high overheads in usage times of the primary applications, which might e.g. be writing a mail or searching for something on the Internet. As described earlier, this design results from adopting phone call UIs of the previous generation of mobile phones with hard buttons to the current generation of smartphones providing touch screens. In fact, current mobile operating systems allow users to multitask between different applications, including the phone call application, during phone calls. However, current phone call applications do not provide any options to the user for multitasking or handling incoming calls beyond accepting or declining the call.

In this work, we tackle the two issues described by revisiting and extending the design space of phone call applications as follows: In particular, we increase the user's freedom in deciding when to pick up a call by introducing the possibility of postponing an incoming phone call. Furthermore, we reconsider on the design of user interfaces for phone call applications to mitigate the interruptive effect of incoming calls while an application is being used. The key contribution of this work is to extend the conceptual design of current phone call applications to allow for a higher degree of multitasking and additional options to handle incoming calls as described hereafter and shown in Figures 6.4(b) to 6.4(d).

Option to Postpone Incoming Calls

A first improvement to mitigate the effect of phone call interruptions is to give the callee an additional option for when to pick up the call. We provide a third option for postponing the call, rather than accepting or declining it. Hence, the user can easily and quickly return to his previous task without a need to decide how to react

to the incoming call. The approach of postponing calls brings a user's ability to pick up the call at will from landline phones to mobile phones. Thus it becomes possible on mobile phones to have the case where a user is working on an (up to now stationary) computer and the (up to now landline) phone rings, and he can either immediately answer the call or first finish his primary task, e.g. writing an e-mail.

As Figure 6.4(b) shows, this option can be implemented as an additional button (besides accept and decline) when the full-screen call notification pops up. When a call is postponed, the phone call application should go into the background and the user can continue working in her previous application. The caller will wait on the line. At the choice of the user or after a certain amount of time, the call application will come to the foreground again, and again the user has the three options to accept, postpone or decline the call.

Multiplexing Notifications

Space-multiplexing of primary and secondary tasks on the device's limited screen real estate has been found to provide the user with more control when interrupted [5]. Also Ratwani et al. [197] suggest designing interfaces in a way that users have visual access to their primary task when interruptions occur. Therefore, a second approach to mitigate phone call interruptions is to alter the visual appearance of call notifications, as Figure 6.4(b) suggests. Rather than having a full-screen notification as in current phone applications, we propose to divide the mobile screen's limited space into two areas: a smaller area, where the user is notified of the incoming call, as shown at the bottom in Figure 6.4(b); and a second larger area, where the user can continue working on her primary task, which for example might be navigating on a map (Figure 6.4(b). Again, the user has the aforementioned options for reacting to the interruption.

Following on the idea of space-multiplexing, the user can keep his attention on the primary application and he will not be visually decoupled from the previous application. For instance, he can continue reading, or finish the sentence he was just typing.

Background Alerts

Another way to notify users of incoming calls could be through background alerts, which will occupy very little space on the screen. As can be seen in Figure 6.4(d), notifications for incoming phone calls can also be signaled by leveraging the multimodal capabilities provided by smartphones, like vibration for tactile or ringtones for auditive signals. This results in incoming calls being put into the background immediately. Hence, the user can stay in the primary application, continue with his current task, and reach a stopping point to leave his application. This allows the user to intentionally leave his current application to take the call when desired, or even to leave the phone unanswered.

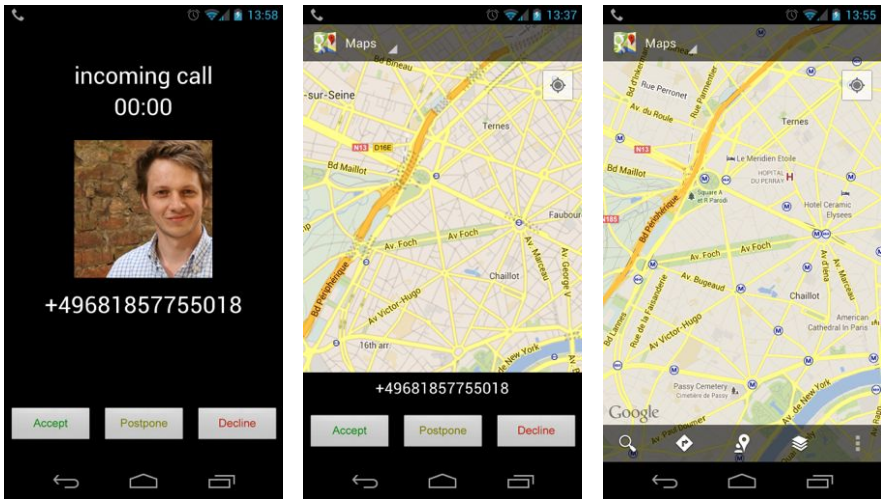
Scheduling on Completion

Messages have been found to be best delivered on the transition of activities [123]. Therefore, an additional approach for scheduling a call interruption is to wait until the user closes his current application. Following on this idea, when a call comes in, the user will not be immediately notified of the call; rather when he closes the current application, a notification will be shown. Since the user is done with the recent application, a full-screen notification with a postpone option is reasonable. Since we cannot know for how long the user will stay in the current application, after a certain short amount of time, a fallback to one of the aforementioned approaches is required.

6.3.2 Prototype Implementation

While in previous generations of mobile phones the phone call application was the device's essential part, and everything else was built around it or integrated into it, phone call applications on the current generation of smartphones are just pieces of software like any other application running on the device. This enabled us to build a prototype application that implements the four aforementioned approaches to notify users of incoming calls and to test new ways of handling calls.

We implemented the presented design options for call notifications for Android-based smartphones. For managing calls, we have implemented an application to simulate an extension of a Bluetooth headset, and we were thereby able to manage handling of phone calls. Figure 6.5 depicts the four different UI designs for a user navigating on a map as a primary task. Figure 6.5(a) shows the option to push



(a) Option for postponing (b) Multiplexing notifications (c) Background alert

Figure 6.5: Screenshots of our prototype showing 3 of 4 new approaches to notify the user of incoming calls.

a postpone button, 6.5(a) shows the call notification multiplexed with the user's map application so that he is able to continue working on his primary task, and 6.5(c) shows how the user is able to continue in the primary application when the incoming call is put into the background. Note that in the latter case the user is notified of the incoming call only by the icon in the screen's top bar.

6.3.3 Discussion

We designed this approach for mitigating the problem of call interruptions in a preventive way, as discussed in Section 6.2. However, the problem of the increased resumption lag when switching between applications remains. Further, the method we proposed to mitigate phone call interruptions is a preventive one, and curative ones remain unexplored. However, we decided to address this particular problem since incoming phone calls introduced the largest overhead. While curative approaches for mitigating the interruptions might be the same for phone call interruptions and application switching interruptions, preventive approaches have to be designed differently since multitasking between two applications on the one hand or an application interrupted by a call on the other hand are two different problems due to the nature of a phone call.

Besides adding the opportunity to postpone a call and extending the UI aspects of the notification (full-screen, space-multiplexed, background alert, on application completion) to mitigate the problem of call interruptions, we also need to be concerned about the caller, about other modalities like ring tone and vibration, and about which UI option should be chosen to display the call.

Reasoning on Predicted Overhead. The four extensions of current phone call applications differ in the degree to which they pull away the user's attention from the application that he currently is using to the call. As described in Section 6.2, we can measure the overhead that results from a phone call interruption. To reason about how to handle an incoming phone call, i.e. which UI to use to notify the user of the incoming call, a model that predicts the overhead will be beneficial (as Iqbal and Bailey [127] have build a model for predicting the resumption lag for desktop interruptions). While for instance the accept/decline/postpone dialog might be best suited when the user is making an entry in his calendar, a space-multiplexed notification might be best suited when he is watching a video, and a background notification might be best suited when the user is playing a highly engaging game. Besides the content (e.g. determined by who is calling), such factors relate to the *receptivity* of mobile interruptions [97].

To predict the resulting overhead, various variables are available: Besides the interrupted application's name and type, the user's last reaction to an interruption, and the caller ID as well as contextual factors like time of day or location can also be taken into account. Social context has also been found to impact the caller's interruption decision making and the callee's availability (cf. [111, 212]). Such a model will be beneficial when reasoning about how to react to an incoming call: If the overhead is estimated to be high, a less intrusive background notification will be chosen; if the overhead is estimated to be low, a more interruptive accept/decline/postpone dialog might be the best option.

Further, interruptions have been found to improve performance on simple tasks and to lower performance on complex tasks [233]. Therefore the call-notifying UI should also be chosen with regard to the complexity of the interrupted application. The higher the degree of the user's immersion in his primary application, the higher also the risk is that he will not see the multiplexed notification due to the matter of display blindness on mobile devices [77].

Keeping the Caller Waiting on the Line. When the receiver postpones a call or the call is put to background automatically, the caller will be kept waiting on the line. Since related work (see [10, 241, 221] and Section 2.3.5) found that it has a positive effect when the caller is aware of the callee's status, one idea might be to notify the caller as to what is happening. One possibility could be the design of special call-progress tones (e.g., for *"call was postponed"* or *"call was notified in background"*), or we could for instance use speech-synthesis to tell the caller that *"the callee is currently watching a movie"*.

Other Modalities. The notification scenarios described in this section mainly address visual attention. Different modalities have shown different degrees of interruptiveness (e.g. [8, 197]). In addition to visual notifications, incoming calls are also being signaled by auditive and haptic signals. A holistic design needs to consider these ways of notifying the user of incoming phone calls as additional modalities. One possibility could be changing the ringtone to unobtrusive sounds. Hence, the user could be notified about an incoming phone call in an ambient way. As another possibility, we could apply different vibration patterns to create haptic notifications in accordance with the visual notification. The integration of different modalities therefore needs to be addressed in the future, and to be aligned with the visual notification.

6.4 Summary

In this chapter, we began to address the matter of multitasking between applications on smartphones. As discussed, the saliency of this issue will increase, since the number of services provided through smartphone applications is perpetually increasing as is the time that people spend using their smartphones as daily companions (cf. Chapter 1).

In this work, we have shown that application interruptions on smartphones can have a serious impact on the performance of the users. In our study of smartphone interruptions, we have replicated previous findings of previous lab studies on desktop interruptions and extended them to the mobile domain by exploiting a large-scale dataset of mobile application usage. We have observed that application switching behavior and incoming phone calls are a non-negligible source of disruption and therefore they should be diminished. Our study reveals three general findings:

1. Application interruptions rarely happen on smartphones; but when they do, they can be really costly for the user. This poses a wealth of new challenges for mobile designers and smartphone vendors.
2. Application switching behavior does not happen as often as it is presumed. While smartphones make it possible to change the focus of interaction, users are reluctant to do so. One reason might be that there are no mechanisms or suitable interaction techniques (yet) to support regaining context after switching between mobile applications.
3. Phone call interruptions add a significant overhead on the interrupted application in comparison to those due to application switching. This was expected, as incoming calls potentially can happen anytime. However, it was surprising to notice the cost on the interrupted application: up to four times compared to its normal usage in terms of runtime.

Furthermore, we have discussed possible approaches to reduce the overhead caused by application interruptions and to help users resume task flow. For the case of phone call interruptions — and in the second part of this chapter — we revisited the design of mobile phone call applications. Motivated by the high interruption cost of incoming phone calls on concurrent application usage, we explored new UIs for notifying users of incoming calls. We extended users' options for how to handle incoming calls by a *postpone* function in addition to *accept* and *decline*. Further, we presented the approaches of multiplexing, background alerts and scheduling on application completion for call notifications. Besides new design options, we presented a prototype implementation, and discussed challenges and future work.

Chapter 7

General Conclusions

In this last chapter we will summarize the key contributions of this work. We will also discuss new opportunities and challenges for future work that result from our work but that go beyond the scope of this thesis. Finally, we will close this thesis with final remarks.

7.1 Major Contributions

The work presented in this thesis has focused on understanding the principles of mobile smartphone usage, and assisting people to leverage the huge variety of applications that they can use on their devices. Successfully understanding mobile application usage is challenging due to the huge mobile ecosystem and large variety of applications and use contexts of smartphones.

In the first part (Chapter 3), we investigated how people launch the applications they have installed on their devices and developed adaptive icon menus to ease launching of applications. Next (Chapter 4), we presented results of a study on mobile application housekeeping, and presented a system to help people with arranging icons in their smartphone launchers. Thirdly (Chapter 5), we looked into how people can find new applications that are of interest and valuable for them and presented approaches for context-aware recommendation of mobile applications to support mobile application discovery. In the fourth part (Chapter 6) we investigated how people multitask on their smartphones and looked into the cost of mobile application interruptions, especially as regards multitasking with phone

calls. Since call interruptions can add a significant overhead to application usage, we developed and presented a new design for phone call UIs.

The contributions of this work were made by applying the methods of research through the application store and other methods where reasonable to the field of smartphone usage. Further, we developed and studied prototypes to solve the shortcomings of current state-of-the-art technology as well as literature. The work presented in this thesis has made contributions to the four fields of launching (L), housekeeping (H), discovery (D) of and multitasking (M) with mobile applications — and to mobile HCI in general:

(L1) A framework for contextualized tracing of mobile application usage.
(L2) A large-scale study on mobile application usage.
(L3) An adaptive shortcut menu for supporting mobile application launching.

(H1) A study suggesting that context impacts icon arrangement.
(H2) A study of people's concepts for arranging icons in launcher menus.
(H3) A system to support icon arrangement in smartphone launchers.

(D1) A design space for mobile application recommender systems.
(D2) An architecture for recommender systems suggesting mobile applications.
(D3) A new method for a usage-centric evaluation of recommender systems.

(M1) A study of the phenomenon of mobile application interruptions.
(M2) A phone call UI for lowering the impact of mobile phone call interruptions.

In addition to these contributions, which individually can be used to inform the understanding of smartphone usage and to improve future design of smartphone interaction, we also contribute the essential technical systems implemented during the course of this thesis as open source[1] as well as parts of the data we collected, where possible given legal and ethical considerations[2]. This contribution will allow other researchers on the one hand to produce their own results, draw their own conclusions and gain new understanding based on our data, and on the other hand to build their own systems based on the approaches that we have developed and provided in this work.

We cannot claim to have invented the approach of conducting research through the application store itself, since it seemed a natural way to go given the circumstances described in Chapter 2. However, given the number of citations of our paper on

[1] http://matthiasboehmer.de/code
[2] http://matthiasboehmer.de/data

AppSensor [38] (61 at the times of writing this thesis[3]) and since it was one of the first papers in the field, we can claim that we have impacted the field by using this approach. As such, this can also be seen as a contribution on its own.

The contributions achieved in this thesis have been published and presented at conferences and workshops [29, 30, 31, 32, 33, 36, 37, 38, 40, 42, 157, 183].

7.2 Future Work

The fundamental findings we provide in this thesis open new opportunities and leave open challenges that can be addressed in future work. To summarize: One could exploit the information on mobile application usage, refine and extend our studies, improve presented applications and systems, and build a single solution integrating all results, as we discuss next:

Exploiting Signals of Application Interaction

In this thesis, we began to understand the end-user's utilization of mobile applications. Based on our observations of user behavior, we described patterns and concepts that people use to inform our understanding of smartphone usage. In Want's *"you are your cell phone"* line of thinking [258] such signals provide rich information about the users themselves, and can be used to learn about and react on the users' profiles. Our work gives rise to identifying and interpreting such new signals (e.g., application launches, events of icon arrangements). This gives the opportunity to extend the large body of research done to understand usage of websites on the Internet (usually based on click-through rates and navigation paths) to the mobile ecosystem of applications. Such an algorithmic grounding and modeling of the user behaviors described in this dissertation (e.g. icon arrangement and application launching) will further improve our understanding of application usage, and this will further give rise to its exploitation for improved support of smartphone usage.

As discussed in Chapter 3, it seems promising to deduce a user's context from the applications that he is currently using. This additional piece of context can be used for other applications as input or with the smartphone's operating system itself to

[3] According to Google Scholar, http://scholar.google.de/citations?user=cwlk77oAAAAJ, last accessed on 15.07.2013

adapt functionality to context. Further, one might look for relations between users' application interactions like the concept of icon arrangement (see Chapter 4) and general user profiles like personality traits (cf. [59, 60, 125, 54]) as proposed by Raento et al. [191].

Extension and Refinement of Studies

The results and conclusions presented in this work are based on studies of application usage. The method of research through the application store, which this work mainly relies on, bears some limitations, which can be addressed in future studies. For instance, we cannot know the ground truth of the behavior of our participants when looking into application launches (Chapter 3). For future refinement of our large-scale study of mobile application usage we propose to use the experience sampling method as done by Church et al. [63] to gain additional insights into users' contexts when using applications.

To dig deeper into multitasking of mobile applications, future work might look into the impact of notifications on application usage behavior. We assume that incoming notifications are also a good predictor of application launching behavior. They can be treated as external interruptions like the incoming phone calls we analyzed in Chapter 6. As such, a study of notifications will further improve our understanding of application interruptions (as already studied on stationary computers; see Chapter 2).

Improvement of Presented Applications and Systems

For discovery of mobile applications we think that better recommender engines can be built by focusing on improvement of the algorithms. For future work, optimizing the recommender engines implemented in Chapter 5 seems to be promising (cf. e.g. [139, 140, 228]). Further, integrating other sources of information to inform algorithmic supportive functionality seems to be promising, e.g. the large amount of meta-data that is available for applications like, such as comments provided by other users. Such new approaches should be evaluated based on the framework we presented in Chapter 5. In addition, we plan to evaluate different measures for implicit relevance feedback; as proposed in the design space for mobile application recommender systems (Chapter 5), the idea of integrating icon arrangement (both contextual and habitual) as implicit feedback seems promising — but has not been studied in this work.

One Integrated Single Solution

One natural next step to the work presented in this thesis would be an integration of all findings and proposed systems and applications into one single solution, and the study of such a smartphone under aspects of general usability and user experience. This would be a smartphone that tracks application usage and supports launching, that analyses changes in user icon arrangement and acts based on them, recommends new applications or makes them instantly available to use, and supports multitasking between phone calls — i.e. the primary device functionality — and other applications. However, while it would be possible to build such a single integrated solution that exploits all findings of this thesis and all proposed supportive functionality into one single solution, this system would demand a high degree of low-level access to fundamental operating system functionality. Such a prototype would result in a proprietary branch of a whole operating system that we would not be able to evaluate with the method of research through the application store, which we rely on in this work. We would instead have to distribute a new smartphone with a customized software stack, which would require substantial resources if it were to be done on a large scale. This piece of future work is instead left for industry to inform the design of a new smartphone.

7.3 Closing Remarks

This thesis has investigated the current generation of smartphones resulting from the recent development of hardware and software, which transformed mobile phones into multi-functional toolsets over their 40-year evolution (see Chapter 1). We can only foresee the future in a tautological way and discuss both options: Smartphones will either continue to improve or disappear — i.e., we will proceed either with *integration* or *dis-integration* of smartphones into our daily life as described by Harmon and Mazmanian [110].

On the one hand, elaborating on the thinking of the recent evolution of smartphones, one might anticipate that mobile computing power, devices' interaction capabilities and functional richness and variety will further increase. This will call for further refinements of the approaches and solutions presented in this thesis, as well as new approaches to tackle the problems of discovery, launching, housekeeping and multitasking. Pioneering work has already been done, e.g., by leveraging the space around the user as a virtual extension around the body (cf. e.g. [58, 56]).

On the other hand, one might argue that smartphones will disappear in the future. The big advantage of applications as digital services over dedicated information appliances is that digital goods are much easier to duplicate, ship and distribute than physical goods, as we know from the transition of analogue music that was shipped as a physical good to digital music that can be "shipped" over the Internet.

There is ongoing and groundbreaking research in the area of tangible interaction and wearable computing in which everyday objects become smart through enrichment with sensing, computing and networking capabilities; see for instance the *Bottle Interface* [131], *MediaCups* [21], *ShoeSense* [13] and even the human hand as *Imaginary Phone* [108]. Two other good examples of common artifacts which are currently starting a similar evolution as the mobile phone and becoming mainstream are wristwatches and eyeglasses. While wristwatches were originally designed to tell the time centuries ago, they are now becoming smart multipurpose artifacts which serve functionalities far beyond telling the time (cf. e.g. [27, 186, 193]). Similarly, eyeglasses were initially invented to correct human sight in case of visual impairment. They are now in transition towards becoming devices augmenting human sight with digital information beyond correcting visual impairments. While groundbreaking research was done years ago (see e.g. [201, 236, 92]), such devices are now becoming mainstream with first products like *Google Glass*.[4]

The one thing that mobile phones, wristwatches and glasses have in common in their evolution is that all become smart in terms of integrating more functionality. We may assume that one piece of tangible user interface will always be used for more than one service. That is — literally comparing to smartphones — one piece of hardware will run more than one application. Not only did the phone became a smart-phone, but things will also become smart-things — especially when considering paradigms like the *Internet of Things* [9] and *Ubiquitous Computing* [261].

First research projects are working in the direction of extending the concepts of applications and application stores to other smart devices (see e.g. [69] for public displays or the project *SmartF-IT*[5] for industry domain). As such, we could add the investigation of extension of our research questions to other systems running *"apps"* like glasses, watches, public displays or industrial machines to the body of future work stated in Section 7.2. However, the main challenges that arise when

[4]Google: *Glass*, http://www.google.com/glass/, last accessed on 30.06.2013.
[5]http://www.smartf-it-projekt.de, last accessed on 30.06.2013.

functions beyond a *thing*'s main purpose are embodied into one single piece of hardware will remain:

- How to launch the right service at the right time?
  (See application launching, Chapter 3)
- How to organize the different applications?
  (See application housekeeping, Chapter 4)
- How to discover what can be done with the smart thing?
  (See application discovery, Chapter 5)
- What is the impact of adding functionalities beyond the main function?
  (See application multitasking, Chapter 6)

With the work presented in this dissertation, we lay the foundation to address these four central questions (discovering new applications, launching useful applications, housekeeping installed applications and multitasking between applications) by investigating them for the first device that became not only smart but also ubiquitous in our daily life: the mobile phone.

One more thing...[6] At the first workshop on Mobile HCI in 1998, Johnson [136] made a comparison between a mobile phone and a watch to illustrate how commonplace using a mobile phone will become in the future — i.e. like wearing a watch. However, ironically he remarked that the analogy would be limited *"since a watch will normally only tell the time, date and ring an alarm"* [136]. This example shall illustrate how we need to rethink existing approaches and be aware of making wrong assumptions during these times where revolutionary technologies are moving into our daily lives and changing the nature of *things*.

[6] In remembrance of Steve Jobs, who had a major impact on how people think about mobile phones today. http://en.wikipedia.org/wiki/Stevenote, last accessed on 11.06.2013.

Bibliography

[1] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and Context-Awareness. In *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 304–307, London, UK, 1999. Springer-Verlag.

[2] G. Adomavicius and A. Tuzhilin. Context-Aware recommender systems. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, chapter 7, pages 217–253. Springer US, Boston, MA, 2011.

[3] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, SIGMOD '93, pages 207–216, New York, NY, USA, 1993. ACM.

[4] Z. Ahmet and K. V. V. Mattila. Mobile service distribution from the end-user perspective: the survey study on recommendation practices. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '12, pages 573–588, New York, NY, USA, 2012. ACM.

[5] F. Alt, A. S. Shirazi, A. Schmidt, and R. Atterer. Bridging waiting times on web pages. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*, MobileHCI '12, pages 305–308, New York, NY, USA, 2012. ACM.

[6] M. G. Ames. Managing mobile multitasking: the culture of iPhones on stanford campus. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 1487–1498, New York, NY, USA, 2013. ACM.

[7] G. Anthes. Invasion of the mobile apps. *Commun. ACM*, 54(9):16–18, 2011.

[8] E. Arroyo and T. Selker. Self-adaptive multimodal-interruption interfaces. In *Proceedings of the 8th international conference on Intelligent user interfaces*, IUI '03, pages 6–11, New York, NY, USA, 2003. ACM.

[9] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787–2805, 2010.

[10] 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. Tech.*, 26(3):247–259, 2007.

[11] A. Bagherjeiran, A. O. Hatch, and A. Ratnaparkhi. Ranking for the conversion funnel. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 146–153, New York, NY, USA, 2010. ACM.

[12] B. P. Bailey, J. A. Konstan, and J. V. Carlis. The effects of interruptions on task performance, annoyance, and anxiety in the user interface. In *Proceedings of IFIP International Conference on Human Computer Interaction*, INTERACT '01, pages 593–601, 2001.

[13] G. Bailly, J. Müller, M. Rohs, D. Wigdor, and S. Kratz. ShoeSense: a new perspective on gestural interaction and wearable applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 1239–1248, New York, NY, USA, 2012. ACM.

[14] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy consumption in mobile phones: a measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, pages 280–293, New York, NY, USA, 2009. ACM.

[15] L. Baltrunas, M. Kaminskas, B. Ludwig, O. Moling, F. Ricci, A. Aydin, K.-H. Lüke, and R. Schwaiger. InCarMusic: Context-Aware music recommendations in a car. In C. Huemer and T. Setzer, editors, *E-Commerce and Web Technologies*, volume 85 of *Lecture Notes in Business Information Processing*, chapter 8, pages 89–100. Springer, Berlin, Heidelberg, 2011.

[16] L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. In *RecSys: Proceedings of the third ACM conference on Recommender systems*, pages 245–248, New York, NY, USA, 2009. ACM.

[17] L. Barkhuus and V. Polichar. Empowerment through seamfulness: smart phones in everyday life. *Personal and Ubiquitous Computing*, pages 1–11, 2010.

[18] L. Barnard, J. Yi, J. Jacko, and A. Sears. Capturing the effects of context on human performance in mobile computing systems. *Personal and Ubiquitous Computing*, 11(2):81–96, 2007.

[19] D. Barreau and B. A. Nardi. Finding and reminding: file organization from the desktop. *SIGCHI Bull.*, 27(3):39–43, 1995.

[20] H. Becker, A. Broder, E. Gabrilovich, V. Josifovski, and B. Pang. What happens after an ad click?: quantifying the impact of landing pages in web advertising. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 57–66, New York, NY, USA, 2009. ACM.

[21] M. Beigl, H.-W. Gellersen, and A. Schmidt. Mediacups: experience with design and use of computer-augmented everyday artefacts. *Computer Networks*, 35(4):401–409, 2001.

[22] G. Bell and P. Dourish. Yesterday's tomorrows: notes on ubiquitous computing's dominant vision. *Personal and Ubiquitous Computing*, 11(2):133–143, 2007.

[23] V. Bellotti, B. Begole, E. H. Chi, N. Ducheneaut, J. Fang, E. Isaacs, T. King, M. W. Newman, K. Partridge, B. Price, P. Rasmussen, M. Roberts, D. J. Schiano, and A. Walendowski. Activity-based serendipitous recommendations with the magitti mobile leisure guide. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1157–1166, New York, NY, USA, 2008. ACM.

[24] D. L. Ben Lulu and T. Kuflik. Functionality-based clustering using short textual description: helping users to find apps installed on their mobile device. In *Proceedings of the 2013 international conference on Intelligent user interfaces*, IUI '13, pages 297–306, New York, NY, USA, 2013. ACM.

[25] R. Benbunan-Fich, R. F. Adler, and T. Mavlanova. Measuring multitasking behavior with activity-based metrics. *ACM Trans. Comput.-Hum. Interact.*, 18(2), 2011.

[26] O. Bergman, M. Tene-Rubinstein, and J. Shalom. The use of attention resources in navigation versus search. *Personal and Ubiquitous Computing*, 17(3):583–590, 2013.

[27] G. Blasko and S. Feiner. An interaction system for watch computers using tactile guidance and bidirectional segmented strokes. In *Proceedings of the Eighth International Symposium on Wearable Computers*, ISWC '04, pages 120–123, Washington, DC, USA, 2004. IEEE Computer Society. (to appear).

[28] J. O. Blom and A. F. Monk. Theory of personalization of appearance: why users personalize their pcs and mobile phones. *Human–Computer Interaction*, 18(3):193–228, 2003.

[29] M. Böhmer and G. Bauer. The case for Context-Collaborative filtering in pervasive services environments. *In Proceedings of Workshop on Context-aware Mobile Media and Mobile Social Networks*, 2009.

[30] M. Böhmer and G. Bauer. Improving the recommendation of mobile services by interpreting the user's icon arrangement. In *MobileHCI: Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, New York, NY, USA, 2009. ACM.

[31] M. Böhmer and G. Bauer. Exploiting the icon arrangement on mobile devices as information source for context-awareness. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, MobileHCI '10, pages 195–198, New York, NY, USA, 2010. ACM.

[32] M. Böhmer, G. Bauer, and A. Krüger. Exploring the Design Space of Recommender Systems that Suggest Mobile Apps. In *Proceedings of Workshop on Context-Aware Recommender Systems*, 2010.

[33] M. Böhmer, G. Bauer, and A. Krüger. Context tags: exploiting user-given contextual cues for disambiguation. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11, pages 611–616, New York, NY, USA, 2011. ACM.

[34] M. Böhmer, G. Bauer, and W. Wicht. LBS 2.0 - enabling User-Driven provision and Context-Aware utilisation of Location-Based services. In *2008 The Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, pages 374–375. IEEE, 2008.

[35] M. Böhmer, G. Bauer, and W. Wicht. Hiding the complexity of LBS. In *LOCWEB: Proceedings of the 2nd International Workshop on Location and the Web*, pages 1–3, New York, NY, USA, 2009. ACM.

[36] M. Böhmer, L. Ganev, and A. Krüger. Appfunnel: A framework for usage-centric evaluation of recommender systems that suggest mobile applications. In *IUI: Proceedings of the 18th international conference on Intelligent user interfaces*, pages 267–276, New York, NY, USA, 2013. ACM.

[37] M. Böhmer, S. Gehring, J. Hempel, and A. Krüger. Revisiting Phone Call UIs for Multipurpose Mobile Phones. In *Proceedings of the 15th International Conference on Human Computer Interaction with Mobile Devices and Services*. ACM, 2013. (to appear).

[38] M. Böhmer, B. Hecht, J. Schöning, A. Krüger, and G. Bauer. Falling asleep with Angry Birds, Facebook and Kindle: a large scale study on mobile application usage. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11, pages 47–56, New York, NY, USA, 2011. ACM.

[39] M. Böhmer and A. Krüger. Gaming the Android OS for Improving the Design of Smartphone Launchers. In *Workshop on Informing Future Design via Large-Scale Research Methods and Big Data at MobileHCI*, 2013.

[40] M. Böhmer and A. Krüger. A study on icon arrangement by smartphone users. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 2137–2146, New York, NY, USA, 2013. ACM.

[41] M. Böhmer, C. Lander, and A. Krüger. What's in the apps for context? extending a sensor for studying app usage to informing context-awareness. In *Proceedings of 2nd International Workshop on Ubiquitous Mobile Instrumentation*. ACM, 2013. (to appear).

[42] M. Böhmer, M. Prinz, and G. Bauer. Contextualizing Mobile Applications for Context-aware Recommendation. In *Adjunct Proceedings of Pervasive*, 2010.

[43] M. Braunhofer, M. Kaminskas, and F. Ricci. Recommending music for places of interest in a mobile travel guide. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 253–256, New York, NY, USA, 2011. ACM.

[44] R. Bridle and E. McCreath. Inducing shortcuts on a mobile phone interface. In *Proceedings of the 11th international conference on Intelligent user interfaces*, IUI '06, pages 327–329, New York, NY, USA, 2006. ACM.

[45] B. Brown, M. McGregor, and E. Laurier. iPhone in vivo: video analysis of mobile device use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 1031–1040, New York, NY, USA, 2013. ACM.

[46] D. Brumby and V. Seyedi. An empirical investigation into how users adapt to mobile phone auto-locks in a multitask setting. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*, MobileHCI '12, pages 281–290, New York, NY, USA, 2012. ACM.

[47] D. P. Brumby, S. C. E. Davies, C. P. Janssen, and J. J. Grace. Fast or safe?: how performance objectives determine modality output choices while interacting on the move. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 473–482, New York, NY, USA, 2011. ACM.

[48] D. P. Brumby, D. D. Salvucci, and A. Howes. Focus on driving: how cognitive constraints shape the adaptation of strategy when dialing while driving. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1629–1638, New York, NY, USA, 2009. ACM.

[49] A. Brush and K. Inkpen. Yours, mine and ours? sharing and use of technology in domestic environments. In J. Krumm, G. D. Abowd, A. Seneviratne, and T. Strang, editors, *UbiComp 2007: Ubiquitous Computing*, vol-

ume 4717 of *Lecture Notes in Computer Science*, chapter 7, pages 109–126. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[50] A. Budde and F. Michahelles. Product empire — serious play with barcodes. In *Proceedings of Internet of Things*, IOT 2010, pages 1–7. IEEE, 2010.

[51] P. Buddharaju, Y. Fujiki, I. Pavlidis, and E. Akleman. A novel way to conduct human studies and do some good. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '10, pages 4699–4702, New York, NY, USA, 2010. ACM.

[52] J. Budzik and K. J. Hammond. User interactions with everyday applications as context for just-in-time information access. In *Proceedings of the 5th international conference on Intelligent user interfaces*, IUI '00, pages 44–51, New York, NY, USA, 2000. ACM.

[53] E. Buie and M. Blythe. Spirituality: there's an app for that! (but not a lot of research). In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, pages 2315–2324, New York, NY, USA, 2013. ACM.

[54] S. Butt and J. G. Phillips. Personality and self reported mobile phone use. *Comput. Hum. Behav.*, 24(2):346–360, 2008.

[55] P. G. Campos, I. Cantador, and F. Díez. Exploiting time contexts in collaborative filtering: an item splitting approach. In *Proceedings of the 3rd Workshop on Context-awareness in Retrieval and Recommendation*, CaRR '13, pages 3–6, New York, NY, USA, 2013. ACM.

[56] J. Cauchard, M. Löchtefeld, M. Fraser, A. Krüger, and S. Subramanian. m+pSpaces: virtual workspaces in the spatially-aware mobile environment. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*, MobileHCI '12, pages 171–180, New York, NY, USA, 2012. ACM.

[57] A. Charland and B. Leroux. Mobile application development: web vs. native. *Commun. ACM*, 54(5):49–53, 2011.

[58] X. A. Chen. Body-centric interaction with mobile devices. In *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction*, TEI '12, pages 385–386, New York, NY, USA, 2012. ACM.

[59] G. Chittaranjan, J. Blom, and D. Gatica-Perez. Who's who with Big-Five: Analyzing and classifying personality traits with smartphones. In *2011 15th Annual International Symposium on Wearable Computers*, pages 29–36. IEEE, 2011.

[60] G. Chittaranjan, J. Blom, and D. G. Perez. Mining large-scale smartphone data for personality studies. *Personal Ubiquitous Comput.*, 17(3):433–450, 2013.

[61] T. Choudhury, G. Borriello, S. Consolvo, D. Haehnel, B. Harrison, B. Hemingway, J. Hightower, P. Klasnja, K. Koscher, A. LaMarca, J. A. Landay, L. LeGrand, J. Lester, A. Rahimi, A. Rea, and D. Wyatt. The mobile sensing platform: An embedded activity recognition system. *IEEE Pervasive Computing*, 7(2):32–41, 2008.

[62] A. Y. K. Chua, R. S. Balkunje, and D. H. L. Goh. Fulfilling mobile information needs: a study on the use of mobile phones. In *Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication*, ICUIMC '11, page Article No. 92, New York, NY, USA, 2011. ACM.

[63] K. Church, M. Cherubini, J. Neumann, and N. Oliver. Understanding mobile information needs on a large-scale: tools, experiences and challenges. In *Proceedings of the 2nd international workshop on Research in the large*, LARGE '11, pages 1–4, New York, NY, USA, 2011. ACM.

[64] K. Church, A. Cousin, and N. Oliver. I wanted to settle a bet!: understanding why and how people use mobile search in social settings. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*, MobileHCI '12, pages 393–402, New York, NY, USA, 2012. ACM.

[65] K. Church and N. Oliver. Understanding mobile web and mobile search use in today's dynamic mobile landscape. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11, pages 67–76, New York, NY, USA, 2011. ACM.

[66] K. Church and B. Smyth. Understanding mobile information needs. In *Proceedings of the 10th international conference on Human computer interaction with mobile devices and services*, MobileHCI, pages 493–494, New York, NY, USA, 2008. ACM.

[67] K. Church and B. Smyth. Understanding the intent behind mobile information needs. In *Proceedings of the 14th international conference on Intelligent user interfaces*, IUI, pages 247–256, New York, NY, USA, 2009. ACM.

[68] K. Church, B. Smyth, K. Bradley, and P. Cotter. A large scale study of european mobile search behaviour. In *Proceedings of the 10th international conference on Human computer interaction with mobile devices and services*, MobileHCI '08, pages 13–22, New York, NY, USA, 2008. ACM.

[69] S. Clinch, N. Davies, T. Kubitza, and A. Schmidt. Designing application stores for public display networks. In *Proceedings of the 2012 International Symposium on Pervasive Displays*, PerDis '12, New York, NY, USA, 2012. ACM.

[70] A. Cockburn, C. Gutwin, and S. Greenberg. A predictive model of menu performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 627–636, New York, NY, USA, 2007. ACM.

[71] P. Coulton and W. Bamford. Experimenting through mobile 'apps' and 'app stores'. *Int. J. Mob. Hum. Comput. Interact.*, 3(4):55–70, 2011.

[72] H. Cramer, M. Rost, N. Belloni, F. Bentley, and D. Chincholle. Research in the large. using app stores, markets, and other wide distribution channels in ubicomp research. In *Proceedings of the 12th ACM international conference adjunct papers on Ubiquitous computing - Adjunct*, Ubicomp '10 Adjunct, pages 511–514, New York, NY, USA, 2010. ACM.

[73] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 39–46, New York, NY, USA, 2010. ACM.

[74] Y. Cui and V. Roto. How people use the web on mobile devices. In *Proc.eeding of the 17th international conference on World Wide Web*, WWW, pages 905–914, New York, NY, USA, 2008. ACM.

[75] L. Dabbish, G. Mark, and V. M. González. Why do i keep interrupting myself?: environment, habit and self-interruption. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 3127–3130, New York, NY, USA, 2011. ACM.

[76] C. Davidsson and S. Moritz. Utilizing implicit feedback and context to recommend mobile applications from first use. In *Proceedings of the 2011 Workshop on Context-awareness in Retrieval and Recommendation*, CaRR '11, pages 19–22, New York, NY, USA, 2011. ACM.

[77] T. Davies and A. Beeharee. The case of the missed icon: change blindness on mobile devices. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, CHI '12, pages 1451–1460, New York, NY, USA, 2012. ACM.

[78] E. S. De Guzman, M. Sharmin, and B. P. Bailey. Should i call now? understanding what context is considered when deciding whether to initiate remote communication via mobile devices. In *Proceedings of Graphics Interface 2007*, GI '07, pages 143–150, New York, NY, USA, 2007. ACM.

[79] R. Demumieux and P. Losquin. Gather customer's real usage on mobile phones. In *Proc.eedings of the 7th international conference on Human computer interaction with mobile devices and services*, MobileHCI, pages 267–270, New York, NY, USA, 2005. ACM.

[80] A. K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7–7, 2001.

[81] A. K. Dey, D. Ferreira, and V. Kostakos. Lessons learned from Large-Scale user studies: Using android market as a source of data. *Int. J. Mob. Hum. Comput. Interact.*, 4(3):28–43, 2012.

[82] A. K. Dey and J. Mankoff. Designing mediation for context-aware applications. *ACM Trans. Comput.-Hum. Interact.*, 12(1):53–80, 2005.

[83] A. K. Dey, K. Wac, D. Ferreira, K. Tassini, J. H. Hong, and J. Ramos. Getting closer: an empirical investigation of the proximity of user to their smart phones. In *Proceedings of the 13th international conference on Ubiquitous computing*, UbiComp '11, pages 163–172, New York, NY, USA, 2011. ACM.

[84] T. M. Do and D. Gatica-Perez. Where and what: Using smartphones to predict next locations and applications in daily life. *Pervasive and Mobile Computing*, (to appear), 2013.

[85] T. M. T. Do, J. Blom, and D. G. Perez. Smartphone usage in the wild: a large-scale analysis of applications and context. In *Proceedings of the 13th international conference on multimodal interfaces*, ICMI '11, pages 353–360, New York, NY, USA, 2011. ACM.

[86] T. M. T. Do and D. G. Perez. By their apps you shall understand them: mining large-scale patterns of mobile phone usage. In *Proceedings of the 9th International Conference on Mobile and Ubiquitous Multimedia*, MUM '10, New York, NY, USA, 2010. ACM.

[87] J. Dostal, P. O. Kristensson, and A. Quigley. Subtle gaze-dependent techniques for visualising display changes in multi-display environments. In *Proceedings of the 2013 international conference on Intelligent user interfaces*, IUI '13, pages 137–148, New York, NY, USA, 2013. ACM.

[88] P. Dourish. What we talk about when we talk about context. *Personal Ubiquitous Comput.*, 8(1):19–30, 2004.

[89] N. Eagle and A. S. Pentland. Reality mining: sensing complex social systems. *Personal Ubiquitous Comput.*, 10(4):255–268, 2006.

[90] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin. A first look at traffic on smartphones. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, IMC '10, pages 281–287, New York, NY, USA, 2010. ACM.

[91] H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan, and D. Estrin. Diversity in smartphone usage. In *Proceedings of the 8th interna-*

tional conference on Mobile systems, applications, and services, MobiSys '10, pages 179–194, New York, NY, USA, 2010. ACM.

[92] S. Feiner, B. MacIntyre, T. Hollerer, and A. Webster. A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment. In *Proceedings of the 1st IEEE International Symposium on Wearable Computers*, ISWC '97, page 74, Washington, DC, USA, 1997. IEEE Computer Society.

[93] D. Ferreira, A. K. Dey, and V. Kostakos. Understanding human-smartphone concerns: a study of battery life. In *Proceedings of the 9th international conference on Pervasive computing*, Pervasive'11, pages 19–33, Berlin, Heidelberg, 2011. Springer-Verlag.

[94] L. Findlater and J. McGrenere. A comparison of static, adaptive, and adaptable menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 89–96, New York, NY, USA, 2004. ACM.

[95] L. Findlater and J. McGrenere. Impact of screen size on performance, awareness, and user satisfaction with adaptive graphical user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1247–1256, New York, NY, USA, 2008. ACM.

[96] J. E. Fischer, C. Greenhalgh, and S. Benford. Investigating episodes of mobile phone activity as indicators of opportune moments to deliver notifications. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11, pages 181–190, New York, NY, USA, 2011. ACM.

[97] J. E. Fischer, N. Yee, V. Bellotti, N. Good, S. Benford, and C. Greenhalgh. Effects of content and time of delivery on receptivity to mobile interruptions. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, MobileHCI '10, pages 103–112, New York, NY, USA, 2010. ACM.

[98] J. A. Flanagan. Unsupervised clustering of context data and learning user requirements for a mobile device. In A. Dey, B. Kokinov, D. Leake, and R. Turner, editors, *Modeling and Using Context*, volume 3554 of *Lecture Notes in Computer Science*, pages 155–168. Springer Berlin Heidelberg, 2005.

[99] J. A. Flanagan. An unsupervised learning paradigm for Peer-to-Peer labeling and naming of locations and contexts. In M. Hazas, J. Krumm, and T. Strang, editors, *Location- and Context-Awareness*, volume 3987 of *Lecture Notes in Computer Science*, pages 204–221. Springer, Berlin, Heidelberg, 2006.

[100] J. Froehlich, M. Y. Chen, S. Consolvo, B. Harrison, and J. A. Landay. MyExperience: a system for in situ tracing and capturing of user feedback on mobile phones. In *Proc.eedings of the 5th international conference on Mobile systems, applications and services*, MobiSys, pages 57–70, New York, NY, USA, 2007. ACM.

[101] Y. Fukazawa, M. Hara, M. Onogi, and H. Ueno. Automatic mobile menu customization based on user operation history. In *MobileHCI: Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 1–4, New York, NY, USA, 2009. ACM.

[102] I. Giannopoulos, P. Kiefer, and M. Raubal. GeoGazemarks: providing gaze history for the orientation on small display maps. In *Proceedings of the 14th ACM international conference on Multimodal interaction*, ICMI '12, pages 165–172, New York, NY, USA, 2012. ACM.

[103] A. Girardello and F. Michahelles. AppAware: which mobile applications are hot? In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, MobileHCI '10, pages 431–434, New York, NY, USA, 2010. ACM.

[104] A. Girardello and F. Michahelles. Explicit and Implicit Ratings for Mobile Applications. In *Proceedings of INFORMATIK 2010*, pages 606–612, Bonn, 2010. Gesellschaft für Informatik.

[105] M. Gorgoglione, U. Panniello, and A. Tuzhilin. The effect of context-aware recommendations on customer purchasing behavior and trust. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 85–92, New York, NY, USA, 2011. ACM.

[106] S. A. Grandhi, R. Schuler, and Q. G. Jones. Telling calls: facilitating mobile phone conversation grounding and management. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2153–2162, New York, NY, USA, 2011. ACM.

[107] S. A. Grandhi, R. P. Schuler, and Q. Jones. Telling calls: making informed call handling decisions. In *Proceedings of the 8th ACM Conference on Designing Interactive Systems*, DIS '10, pages 43–46, New York, NY, USA, 2010. ACM.

[108] S. Gustafson, C. Holz, and P. Baudisch. Imaginary phone: learning imaginary interfaces by transferring spatial memory from a familiar device. In *Proc. of UIST*, pages 283–292, New York, NY, USA, 2011. ACM.

[109] J. Häkkilä and C. Chatfield. Personal customisation of mobile phones: a case study. In *Proceedings of the 4th Nordic conference on Human-computer interaction: changing roles*, NordiCHI '06, pages 409–412, New York, NY, USA, 2006. ACM.

[110] E. Harmon and M. Mazmanian. Stories of the smartphone in everyday discourse: conflict, tension & instability. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 1051–1060, New York, NY, USA, 2013. ACM.

[111] R. Harr and V. Kaptelinin. Interrupting or not: exploring the effect of social context on interrupters' decision making. In *Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design*, NordiCHI '12, pages 707–710, New York, NY, USA, 2012. ACM.

[112] C. Hayes, P. Massa, P. Cunningham, P. Avesani, and P. Cunningham. An on-line evaluation framework for recommender systems. In *Workshop on Personalization and Recommendation in E-Commerce*, 2002.

[113] T. Henderson and F. B. Abdesslem. Scaling measurement experiments to planet-scale: ethical, regulatory and cultural considerations. In *Proceedings of the 1st ACM International Workshop on Hot Topics of Planet-Scale Mobility Measurements*, HotPlanet '09, pages 1–5, New York, NY, USA, 2009. ACM.

[114] N. Henze and S. Boll. Push the study to the app store: evaluating off-screen visualizations for maps in the android market. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, MobileHCI '10, pages 373–374, New York, NY, USA, 2010. ACM.

[115] N. Henze and S. Boll. Release your app on sunday eve: finding the best time to deploy apps. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11, pages 581–586, New York, NY, USA, 2011. ACM.

[116] N. Henze and M. Pielot. App stores: external validity for mobile HCI. *interactions*, 20(2):33–38, 2013.

[117] N. Henze, M. Pielot, B. Poppinga, T. Schinke, and S. Boll. My app is an experiment: Experience from user studies in mobile app stores. *International Journal of Mobile Human Computer Interaction*, 3(4):71–91, 2011.

[118] N. Henze, B. Poppinga, and S. Boll. Experiments in the wild: public evaluation of off-screen visualizations in the android market. In *Proc.eedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, NordiCHI, pages 675–678, New York, NY, USA, 2010. ACM.

[119] N. Henze, E. Rukzio, and S. Boll. 100,000,000 taps: analysis and improvement of touch performance in the large. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11, pages 133–142, New York, NY, USA, 2011. ACM.

[120] J. L. Herlocker, J. A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, CSCW '00, pages 241–250, New York, NY, USA, 2000. ACM.

[121] D. M. Hilbert and D. F. Redmiles. An approach to large-scale collection of application usage data over the internet. In *Proceedings of the 20th international conference on Software engineering*, ICSE '98, pages 136–145, Washington, DC, USA, 1998. IEEE Computer Society.

[122] A. M. Hinze, C. Chang, and D. M. Nichols. Contextual queries express mobile information needs. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, MobileHCI '10, pages 327–336, New York, NY, USA, 2010. ACM.

[123] J. Ho and S. S. Intille. Using context-aware computing to reduce the perceived burden of interruptions from mobile devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pages 909–918, New York, NY, USA, 2005. ACM.

[124] A. Holzer and J. Ondrus. Mobile application market: A developer's perspective. *Telematics and Informatics*, 28(1):22–31, 2011.

[125] F.-Y. Hong, S.-I. Chiu, and D.-H. Huang. A model of the relationship between psychological characteristics, mobile phone addiction and use of mobile phones by taiwanese university female students. *Computers in Human Behavior*, 28(6):2152–2159, 2012.

[126] J. Huhtala, J. Mäntyjärvi, A. Ahtinen, L. Ventä, and M. Isomursu. Animated transitions for adaptive small size mobile menus. In *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part I*, volume 5726 of *INTERACT '09*, pages 772–781, Berlin, Heidelberg, 2009. Springer-Verlag.

[127] S. T. Iqbal and B. P. Bailey. Leveraging characteristics of task structure to predict the cost of interruption. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 741–750, New York, NY, USA, 2006. ACM.

[128] S. T. Iqbal and E. Horvitz. Disruption and recovery of computing tasks: field study, analysis, and directions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 677–686, New York, NY, USA, 2007. ACM.

[129] S. T. Iqbal, E. Horvitz, Y. C. Ju, and E. Mathews. Hang on a sec!: effects of proactive mediation of phone conversations while driving. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 463–472, New York, NY, USA, 2011. ACM.

[130] S. T. Iqbal, Y. C. Ju, and E. Horvitz. Cars, calls, and cognition: investigating driving and divided attention. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1281–1290, New York, NY, USA, 2010. ACM.

[131] H. Ishii, A. Mazalek, and J. Lee. Bottles as a minimal interface to access digital information. In *CHI '01 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '01, pages 187–188, New York, NY, USA, 2001. ACM.

[132] D. Jannach and K. Hegelich. A case study on the effectiveness of recommendations in the mobile internet. In *Proc.eedings of the third ACM conference on Recommender systems*, RecSys, pages 205–208, New York, NY, USA, 2009. ACM.

[133] S. Jenson. Mobile apps and the approaching zombie apocalypse. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, MobileHCI '10, pages 5–6, New York, NY, USA, 2010. ACM.

[134] J. Jin and L. A. Dabbish. Self-interruption on the computer: a typology of discretionary task interleaving. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1799–1808, New York, NY, USA, 2009. ACM.

[135] J. Johnson. *Designing with the mind in mind.* Morgan Kaufman, 2010.

[136] P. Johnson. Usability and mobility; interactions on the move. In *First workshop on human computer interaction with mobile devices*, 1998.

[137] E. Kaasinen. User needs for location-aware mobile services. *Personal and Ubiquitous Computing*, 7(1):70–79, 2003.

[138] D. Kamisaka, S. Muramatsu, H. Yokoyama, and T. Iwamoto. Operation prediction for Context-Aware user interfaces of mobile phones. In *Proceedings of Ninth Annual International Symposium on Applications and the Internet*, SAINT '09, pages 16–22. IEEE, 2009.

[139] A. Karatzoglou, L. Baltrunas, and M. Böhmer. Collaborative context-aware preference learning. In *NIPS Workshop on Choice Models and Preference Learning*, 2011.

[140] A. Karatzoglou, L. Baltrunas, K. Church, and M. Böhmer. Climbing the app wall: enabling mobile app discovery through context-aware recommendations. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, CIKM '12, pages 2527–2530, New York, NY, USA, 2012. ACM.

[141] H. Karjaluoto, J. Karvonen, M. Kesti, T. Koivumäki, M. Manninen, J. Pakola, A. Ristola, and J. Salo. Factors affecting consumer choice of mobile phones: Two studies from finland. *Journal of Euromarketing*, 14(3):59–82, 2005.

[142] A. K. Karlson, S. T. Iqbal, B. Meyers, G. Ramos, K. Lee, and J. C. Tang. Mobile taskflow in context: a screenshot study of smartphone usage. In *Proc. of CHI '10*, CHI '10, pages 2009–2018, New York, NY, USA, 2010. ACM.

[143] S. Karpischek, F. Michahelles, and E. Fleisch. my2cents: enabling research on consumer-product interaction. *Personal Ubiquitous Comput.*, 16(6):613–622, 2012.

[144] J. H. Kim and K. P. Lee. Culturally adapted mobile phone interface design: correlation between categorization style and menu structure. In *Proceedings of the 9th international conference on Human computer interaction with mobile devices and services*, MobileHCI '07, pages 379–382, New York, NY, USA, 2007. ACM.

[145] A. L. S. King, A. M. Valença, A. C. O. Silva, T. Baczynski, M. R. Carvalho, and A. E. Nardi. Nomophobia: Dependency on virtual environments or social phobia? *Computers in Human Behavior*, 29(1):140–144, 2013.

[146] N. Kiukkonen, B. J., O. Dousse, D. Gatica-Perez, and L. J. Towards rich mobile phone datasets: Lausanne data collection campaign. In *Proceedings of International Conference on Pervasive Services*, ICPS 2010. ACM, 2010.

[147] J. Kjeldskov and J. Paay. Indexicality: Understanding mobile human-computer interaction in context. *ACM Trans. Comput.-Hum. Interact.*, 17(4), 2010.

[148] B. Knijnenburg, M. Willemsen, Z. Gantner, H. Soncu, and C. Newell. Explaining the user experience of recommender systems. *User Modeling and User-Adapted Interaction*, 22(4-5):441–504, 2012.

[149] J. Knittel, A. S. Shirazi, N. Henze, and A. Schmidt. Utilizing contextual information for mobile communication. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, pages 1371–1376, New York, NY, USA, 2013. ACM.

[150] J. A. Konstan and J. Riedl. Recommender systems: from algorithms to user experience. *User Modeling and User-Adapted Interaction*, 22(1):101–123, 2012.

[151] M. Kranz, L. Murmann, and F. Michahelles. Research in the large: Challenges for large-scale mobile application research – a case study about nfc adoption using gamification via an app store. *International Journal of Mobile Human Computer Interaction*, 5(1):45–61, 2013.

[152] C. Kray and M. Rohs. Swiss Army Knife meets Camera Phone: Tool Selection and Interaction using Visual Markers. In *Proc. Joint Workshops MIRW and MGuides*, 2007.

[153] S. Kujala and T. M. Shatz. Emotions, experiences and usability in real-life mobile phone use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 1061–1070, New York, NY, USA, 2013. ACM.

[154] X. Lang, E. Oreglia, and S. Thomas. Social practices and mobile phone use of young migrant workers. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, MobileHCI '10, pages 59–62, New York, NY, USA, 2010. ACM.

[155] I. Lee, J. Kim, and J. Kim. Use contexts for the mobile internet: A longitudinal study monitoring actual use of mobile internet services. *International Journal of Human-Computer Interaction*, 18(3):269–292, 2005.

[156] J. S. Lee and J. C. Lee. Context awareness by case-based reasoning in a music recommendation system. In *Proceedings of the 4th international conference on Ubiquitous computing systems*, UCS'07, pages 45–58, Berlin, Heidelberg, 2007. Springer.

[157] L. Leiva, M. Böhmer, S. Gehring, and A. Krüger. Back to the app: the costs of mobile application interruptions. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*, MobileHCI '12, pages 291–294, New York, NY, USA, 2012. ACM.

[158] D. Lottridge and M. Chignell. Driving under the influence of phones: The importance of cognitive ability and cognitive style on Interruption-Related performance. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 51(20):1393–1397, 2007.

[159] Q. H. Mahmoud and P. Popowicz. Toward a framework for the discovery and acquisition of mobile applications. In *2010 Ninth International Conference on Mobile Business and 2010 Ninth Global Mobility Roundtable (ICMB-GMR)*, pages 58–65. IEEE, 2010.

[160] N. Mallat, M. Rossi, V. K. Tuunainen, and A. Öörni. The impact of use context on mobile services acceptance: The case of mobile ticketing. *Information & Management*, 46(3):190–195, 2009.

[161] S. Marathe and S. S. Sundar. What drives customization?: control or identity? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 781–790, New York, NY, USA, 2011. ACM.

[162] G. Mark, V. M. Gonzalez, and J. Harris. No task left behind?: examining the nature of fragmented work. In *Proceedings of the SIGCHI Conference*

on Human Factors in Computing Systems, CHI '05, pages 321–330, New York, NY, USA, 2005. ACM.

[163] S. Matsui and S. Yamada. Genetic algorithm can optimize hierarchical menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1385–1388, New York, NY, USA, 2008. ACM.

[164] J. McDonald, T. Dayton, and D. McDonald. Adapting menu layout to tasks. *International Journal of Man-Machine Studies*, 28(4):417—435, 1988.

[165] D. McMillan, A. Morrison, O. Brown, M. Hall, and M. Chalmers. Further into the wild: Running worldwide trials of mobile systems. In P. Floréen, A. Krüger, and M. Spasojevic, editors, *Pervasive Computing*, volume 6030 of *Lecture Notes in Computer Science*, chapter 13, pages 210–227–227. Springer, Berlin, Heidelberg, 2010.

[166] D. McMillan, A. Morrison, and M. Chalmers. Categorised ethical guidelines for large scale mobile HCI. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 1853–1862, New York, NY, USA, 2013. ACM.

[167] D. C. McMillan. *Mass participation user trials*. PhD thesis, University of Glasgow, 2012.

[168] S. M. McNee, J. Riedl, and J. A. Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '06, pages 1097–1101, New York, NY, USA, 2006. ACM.

[169] A. Mehrotra, A. Nguyen, J. Blumenstock, and V. Mohan. Differences in phone use between men and women: quantitative evidence from rwanda. In *Proceedings of the Fifth International Conference on Information and Communication Technologies and Development*, ICTD '12, pages 297–306, New York, NY, USA, 2012. ACM.

[170] E. Miluzzo, N. D. Lane, H. Lu, and A. T. Campbell. Research in the app store era: Experiences from the cenceme app deployment on the iphone. In *Workshop at UbiComp 2010 on Research in the Large*, 2010.

[171] A. Möller, M. Kranz, B. Schmid, L. Roalter, and S. Diewald. Investigating self-reporting behavior in long-term studies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 2931–2940, New York, NY, USA, 2013. ACM.

[172] A. Morrison, O. Brown, D. McMillan, and M. Chalmers. Informed consent and users' attitudes to logging in large scale trials. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '11, pages 1501–1506, New York, NY, USA, 2011. ACM.

[173] A. Morrison, D. McMillan, S. Reeves, S. Sherwood, and M. Chalmers. A hybrid mass participation approach to mobile software trials. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 1311–1320, New York, NY, USA, 2012. ACM.

[174] J. Müller. *Context Adaptive Digital Signage In Transitional Spaces*. PhD thesis, University of Münster, 2008.

[175] S. Nylander, J. Faadal, and S. Mottaghy. Couch mobility: the cell phone's most important feature at home is mobility. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '12, pages 1973–1978, New York, NY, USA, 2012. ACM.

[176] S. Nylander, T. Lundquist, and A. Brännström. At home and with computer access: why and where people use cell phones to access the internet. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1639–1642, New York, NY, USA, 2009. ACM.

[177] A. Oulasvirta. Finding meaningful uses for context-aware technologies: the humanistic research strategy. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 247–254, New York, NY, USA, 2004. ACM.

[178] A. Oulasvirta. Rethinking experimental designs for field evaluations. *Pervasive Computing, IEEE*, 11(4):60–67, 2012.

[179] A. Oulasvirta, T. Rattenbury, L. Ma, and E. Raita. Habits make smartphone use more pervasive. *Personal Ubiquitous Comput.*, 16(1):105–114, 2012.

[180] A. Oulasvirta, A. Reichel, W. Li, Y. Zhang, M. Bachynskyi, K. Vertanen, and P. O. Kristensson. Improving two-thumb text entry on touchscreen devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 2765–2774, New York, NY, USA, 2013. ACM.

[181] A. Oulasvirta, S. Tamminen, V. Roto, and J. Kuorelahti. Interaction in 4-second bursts: the fragmented nature of attentional resources in mobile HCI. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pages 919–928, New York, NY, USA, 2005. ACM.

[182] W. Pan, N. Aharony, and A. Pentland. Composite social network for predicting mobile apps installation. In *AAAI'11*, pages –1–1, 2011.

[183] A. Parate, M. Böhmer, D. Chu, D. Ganesan, and B. M. Marlin. Practical prediction, prefetch, and prelaunch for faster access to applications on mobile phones. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, New York, NY, USA, 2013. ACM. (to appear).

[184] P. Parhi, A. K. Karlson, and B. B. Bederson. Target size study for one-handed thumb use on small touchscreen devices. In *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, MobileHCI '06, pages 203–210, New York, NY, USA, 2006. ACM.

[185] K. Partridge and B. Price. Enhancing mobile recommender systems with activity inference. In G.-J. Houben, G. McCalla, F. Pianesi, and M. Zancanaro, editors, *User Modeling, Adaptation, and Personalization*, volume 5535, chapter 29, pages 307–318. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[186] J. Pasquero, S. J. Stobbe, and N. Stonehouse. A haptic wristwatch for eyes-free interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 3257–3266, New York, NY, USA, 2011. ACM.

[187] M. Pielot, N. Henze, and S. Boll. Experiments in app stores - how to ask users for their consent? In *CHI '11 Workshop on Ethics, Logs and Videotape: Ethics in Large Scale Trials & User Generated Content*, 2011.

[188] I. Podnar, M. Hauswirth, and M. Jazayeri. Mobile push: Delivering content to mobile users. In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, ICDCSW '02, pages 563–570, Washington, DC, USA, 2002. IEEE Computer Society.

[189] B. Poppinga, H. Cramer, M. Böhmer, A. Morrison, F. Bentley, N. Henze, M. Rost, and F. Michahelles. Research in the large 3.0: app stores, wide distribution, and big data in MobileHCI research. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services companion*, MobileHCI '12, pages 241–244, New York, NY, USA, 2012. ACM.

[190] P. Pu, L. Chen, and R. Hu. A user-centric evaluation framework for recommender systems. In *Proc. RecSys '11*, pages 157–164, 2011.

[191] M. Raento, A. Oulasvirta, and N. Eagle. Smartphones: An emerging tool for social scientists. *Sociological Methods & Research*, 37(3):426–454, 2009.

[192] M. Raento, A. Oulasvirta, R. Petit, and H. Toivonen. ContextPhone: A prototyping platform for Context-Aware mobile applications. *IEEE Pervasive Computing*, 4(2):51–59, 2005.

[193] M. T. Raghunath and C. Narayanaswami. User interfaces for applications on a wrist watch. *Personal Ubiquitous Comput.*, 6(1):17–30, 2002.

[194] A. Rahmati, C. Shepard, C. Tossell, M. Dong, Z. Wang, L. Zhong, and P. T. Kortum. Tales of 34 iphone users: How they change and why they are different. Technical report tr-2011-0624, Rice University, 2011.

[195] A. Rahmati, C. Tossell, C. Shepard, P. Kortum, and L. Zhong. Exploring iPhone usage: the influence of socioeconomic differences on smartphone adoption, usage and usability. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*, MobileHCI '12, pages 11–20, New York, NY, USA, 2012. ACM.

[196] A. Rahmati and L. Zhong. Studying smartphone usage: Lessons from a Four- field study. *IEEE Transactions on Mobile Computing*, 12(7):1417–1427, 2013.

[197] R. M. Ratwani, A. E. Andrews, J. D. Sousk, and J. G. Trafton. The effect of interruption modality on primary task resumption. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 52(4):393–397, 2008.

[198] P. Ravasio, S. G. Schär, and H. Krueger. In pursuit of desktop evolution: User problems and practices with modern desktop systems. *ACM ToCHI*, 11(2):156–180, 2004.

[199] R. Rawassizadeh, M. Tomitsch, K. Wac, and Tjoa. UbiqLog: a generic mobile phone-based life-log framework. *Personal and Ubiquitous Computing*, 17(4):621–637, 2013.

[200] D. A. Redelmeier and R. J. Tibshirani. Association between cellular-telephone calls and motor vehicle collisions. *New England Journal of Medicine*, 336(7):453–458, 1997.

[201] J. Rekimoto, Y. Ayatsuka, and K. Hayashi. Augment-able reality: Situated communication through physical and digital spaces. In *Proceedings of the 2nd IEEE International Symposium on Wearable Computers*, ISWC '98, pages 68–75, Washington, DC, USA, 1998. IEEE.

[202] P. Resnick and H. R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, 1997.

[203] D. Riboni and C. Bettini. COSAR: hybrid reasoning for context-aware activity recognition. *Personal Ubiquitous Comput.*, 15(3):271–289, 2011.

[204] R. Riccamboni, A. Mereu, and C. Boscarol. Keys to nature: A test on the iphone market. In *Tools for Identifying Biodiversity: Progress and Problems*, pages 445–450. Edizioni Università di Trieste, 2010.

[205] F. Ricci. Mobile Recommender Systems. *International Journal of Information Technology and Tourism*, 12(3):205–231, 2011.

[206] F. Ricci, L. Rokach, and B. Shapira. Introduction to recommender systems handbook. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, chapter 1, pages 1–35. Springer US, Boston, MA, 2011.

[207] Y. Rogers. *HCI Theory: Classical, Modern, and Contemporary (Synthesis Lectures on Human-Centered Informatics)*. Morgan & Claypool Publishers, 1 edition, 2012.

[208] Y. Rogers, H. Sharp, and J. Preece. *Interaction Design: Beyond Human - Computer Interaction*. Wiley, 3 edition, 2011.

[209] R. Rosales, H. Cheng, and E. Manavoglu. Post-click conversion modeling and analysis for non-guaranteed delivery display advertising. In *Proc. WSDM '12*, pages 293–302, 2012.

[210] A. Said. Identifying and utilizing contextual data in hybrid recommender systems. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 365–368, New York, NY, USA, 2010. ACM.

[211] D. Salber, A. K. Dey, and G. D. Abowd. The context toolkit: aiding the development of context-enabled applications. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, CHI '99, pages 434–441, New York, NY, USA, 1999. ACM.

[212] A. Salovaara, A. Lindqvist, T. Hasu, and J. Häkkilä. The phone rings but the user doesn't answer: unavailability in mobile communication. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11, pages 503–512, New York, NY, USA, 2011. ACM.

[213] D. D. Salvucci. On reconstruction of task context after interruption. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 89–92, New York, NY, USA, 2010. ACM.

[214] D. D. Salvucci and P. Bogunovich. Multitasking and monotasking: the effects of mental workload on deferred task interruptions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 85–88, New York, NY, USA, 2010. ACM.

[215] M. Satyanarayanan. Swiss army knife or wallet? *IEEE Pervasive Computing*, 4(2):2–3, 2005.

[216] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, pages 253–260, New York, NY, USA, 2002. ACM.

[217] B. N. Schilit, D. M. Hilbert, and J. Trevor. Context-aware communication. *Wireless Communications, IEEE*, 9(5):46–54, 2002.

[218] B. N. Schilit and M. M. Theimer. Disseminating active map information to mobile hosts. *IEEE Network*, 8(5):22–32, 1994.

[219] A. Schmidt. Implicit human computer interaction through context. *Personal and Ubiquitous Computing*, 4(2-3):191–199, 2000.

[220] A. Schmidt, M. Beigl, and Hans-W. There is more to context than location. *Computers and Graphics*, 23(6):893–901, 1999.

[221] A. Schmidt, A. Takaluoma, and J. Mäntyjärvi. Context-Aware telephony over WAP. *Personal Ubiquitous Comput.*, 4(4):225–229, 2000.

[222] M. Schneider and S. Kiesler. Calling while driving: effects of providing remote traffic context. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pages 561–569, New York, NY, USA, 2005. ACM.

[223] J. Schöning. *Advanced User Interfaces for Spatial Information*. PhD thesis, Saarland University, 2010.

[224] A. Sears and B. Shneiderman. Split menus: effectively using selection frequency to organize menus. *ACM ToCHI*, 1(1):27–51, 1994.

[225] W. R. Shadish, T. D. Cook, and D. T. Campbell. *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*. Cengage Learning, 2 edition, 2001.

[226] C. Shepard, A. Rahmati, C. Tossell, L. Zhong, and P. Kortum. LiveLab: measuring wireless networks and smartphone users in the field. *SIGMETRICS Perform. Eval. Rev.*, 38(3):15–20, 2011.

[227] K. Shi and K. Ali. GetJar mobile application recommendations with very sparse datasets. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 204–212, New York, NY, USA, 2012. ACM.

[228] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, A. Hanjalic, and N. Oliver. TFMAP: optimizing MAP for top-n context-aware recommendation. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '12, pages 155–164, New York, NY, USA, 2012. ACM.

[229] C. Shin, J. H. Hong, and A. K. Dey. Understanding and prediction of mobile application usage for smart phones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pages 173–182, New York, NY, USA, 2012. ACM.

[230] F. M. Shipman, C. C. Marshall, and T. P. Moran. Finding and using implicit structure in human-organized spatial layouts of information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 346–353, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[231] A. S. Shirazi, M. Rohs, R. Schleicher, S. Kratz, A. Müller, and A. Schmidt. Real-time nonverbal opinion sharing through mobile phones during sports events. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 307–310, New York, NY, USA, 2011. ACM.

[232] T. Sohn, K. A. Li, W. G. Griswold, and J. D. Hollan. A diary study of mobile information needs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 433–442, New York, NY, USA, 2008. ACM.

[233] C. Speier, J. S. Valacich, and I. Vessey. The influence of task interruption on individual decision making: An information overload perspective. *Decision Sciences*, 30(2):337–360, 1999.

[234] R. St. Amant, T. E. Horton, and F. E. Ritter. Model-based evaluation of cell phone menu interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 343–350, New York, NY, USA, 2004. ACM.

[235] K. Stamm, S. I. Ahamed, P. Madiraju, and S. Zulkernain. Mobile intelligent interruptions management (MIIM): a context aware unavailability system. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 599–600, New York, NY, USA, 2010. ACM.

[236] T. Starner, S. Mann, B. Rhodes, J. Levine, J. Healey, D. Kirsch, R. W. Picard, and A. Pentland. Augmented reality through wearable computing. *Presence: Teleoperators and Virtual Environments*, 6(4):386–398, 1997.

[237] D. L. Strayer, F. A. Drews, and W. A. Johnston. Cell phone-induced failures of visual attention during simulated driving. *Journal of experimental psychology. Applied*, 9(1):23–32, 2003.

[238] D. L. Strayer and W. A. Johnston. Driven to distraction: Dual-Task studies of simulated driving and conversing on a cellular telephone. *Psychological Science*, 12(6):462–466, 2001.

[239] S. Suzuki, V. Bellotti, N. Yee, B. E. John, Y. Nakao, T. Asahi, and S. Fukuzumi. Variation in importance of time-on-task with familiarity with mobile phone models. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2551–2554, New York, NY, USA, 2011. ACM.

[240] K. Swearingen and R. Sinha. Beyond algorithms: An HCI perspective on recommender systems. *ACM SIGIR 2001 Workshop on Recommender Systems*, 2001.

[241] J. Teevan and A. Hehmeyer. Understanding how the projection of availability state impacts the reception incoming communication. In *Proceedings of*

the 2013 conference on Computer supported cooperative work, CSCW '13, pages 753–758, New York, NY, USA, 2013. ACM.

[242] J. Teevan, A. Karlson, S. Amini, A. J. B. Brush, and J. Krumm. Understanding the importance of location, time, and people in mobile local search behavior. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11, pages 77–80, New York, NY, USA, 2011. ACM.

[243] H. ter Hofte. Xensible interruptions from your mobile phone. In *Proceedings of the 9th international conference on Human computer interaction with mobile devices and services*, MobileHCI '07, pages 178–181, New York, NY, USA, 2007. ACM.

[244] W. F. Tichy. Multitasking. In *Encyclopedia of Computer Science*, pages 1210–1211. John Wiley and Sons Ltd., Chichester, UK, 2003.

[245] C. C. Tossell, P. Kortum, A. Rahmati, C. Shepard, and L. Zhong. Characterizing web use on smartphones. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, CHI '12, pages 2769–2778, New York, NY, USA, 2012. ACM.

[246] C. C. Tossell, P. Kortum, C. Shepard, A. Rahmati, and L. Zhong. An empirical analysis of smartphone personalisation: measurement and user variability. *Behaviour & Information Technology*, 31(10):995–1010, 2012.

[247] C. C. Tossell, P. Kortum, C. W. Shepard, A. Rahmati, and L. Zhong. Getting real: a naturalistic methodology for using smartphones to collect mediated communications. *Advances in Human-Computer Interaction*, 2012. *http://www.hindawi.com/journals/ahci/2012/815972/*, last accessed 10.07.2013.

[248] J. G. Trafton, E. M. Altmann, D. P. Brock, and F. E. Mintz. Preparing to resume an interrupted task: effects of prospective goal encoding and retrospective rehearsal. *International Journal of Human-Computer Studies*, 58(5):583–603, 2003.

[249] M. van Setten, S. Pokraev, and J. Koolwaaij. Context-Aware recommendations in the mobile tourist application COMPASS. In P. Bra and W. Nejdl, editors, *Adaptive Hypermedia and Adaptive Web-Based Systems*, volume 3137 of *Lecture Notes in Computer Science*, chapter 27, pages 235–244. Springer, Berlin, Heidelberg, 2004.

[250] P. O. S. Vaz De Melo, L. Akoglu, C. Faloutsos, and A. A. F. Loureiro. Surprising patterns for the call duration distribution of mobile phone users. In *Proceedings of the 2010 European conference on Machine learning and knowledge discovery in databases: Part III*, ECML PKDD'10, pages 354–369, Berlin, Heidelberg, 2010. Springer-Verlag.

[251] H. Verkasalo. Contextual patterns in mobile service usage. *Personal and Ubiquitous Computing*, 13(5):331–342, 2009.

[252] H. Verkasalo, C. L. Nicolás, F. J. Molina Castillo, and H. Bouwman. Analysis of users and non-users of smartphone applications. *Telemat. Inf.*, 27(3):242–255, 2010.

[253] A. Vetek, J. A. Flanagan, A. Colley, and T. Keränen. SmartActions: Context-Aware mobile phone shortcuts. In T. Gross, J. Gulliksen, P. Kotzé, L. Oestreicher, P. Palanque, R. Prates, and M. Winckler, editors, *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part I*, volume 5726 of *INTERACT '09*, pages 796–799, Berlin, Heidelberg, 2009. Springer-Verlag.

[254] F. von Reischach, F. Michahelles, and A. Schmidt. The design space of ubiquitous product recommendation systems. In *MUM: Proceedings of the 8th International Conference on Mobile and Ubiquitous Multimedia*, pages 1–10, New York, NY, USA, 2009. ACM.

[255] S. von Watzdorf and F. Michahelles. Accuracy of positioning data on smartphones. In *Proceedings of the 3rd International Workshop on Location and the Web*, LocWeb '10, New York, NY, USA, 2010. ACM.

[256] D. T. Wagner, A. Rice, and A. R. Beresford. Device Analyzer: Large-scale mobile data collection. In *Workshop on Big Data Analytics*, 2013.

[257] G. Wang. Designing smule's ocarina: The iphone's magic flute. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 303—307. *http://www.nime.org/archive/?mode=ylist&y=2009*, last accessed 08.07.2013, 2009.

[258] R. Want. You are your cell phone. *Pervasive Computing, IEEE*, 7(2):2–4, 2008.

[259] R. Want. When cell phones become computers. *IEEE Pervasive Computing*, 8(2):2–5, 2009.

[260] R. Want, A. Hopper, V. F. Ao, and J. Gibbons. The active badge location system. *ACM Trans. Inf. Syst.*, 10(1):91–102, 1992.

[261] M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):66–75, 1991.

[262] W. Woerndl, C. Schueller, and R. Wojtech. A hybrid recommender system for context-aware recommendations of mobile applications. In *2007 IEEE 23rd International Conference on Data Engineering Workshop*, ICDEW '07, pages 871–878, Washington, DC, USA, 2007. IEEE.

[263] Q. Xu, J. Erman, A. Gerber, Z. Mao, J. Pang, and S. Venkataraman. Identifying diverse usage behaviors of smartphone apps. In *Proceedings of the 2011*

ACM SIGCOMM conference on Internet measurement conference, IMC '11, pages 329–344, New York, NY, USA, 2011. ACM.

[264] B. Yan and G. Chen. AppJoy: personalized mobile application discovery. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, MobiSys '11, pages 113–126, New York, NY, USA, 2011. ACM.

[265] T. Yan, D. Chu, D. Ganesan, A. Kansal, and J. Liu. Fast app launching for mobile devices using predictive user context. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, MobiSys '12, pages 113–126, New York, NY, USA, 2012. ACM.

[266] P. Yin, P. Luo, W. C. Lee, and M. Wang. App recommendation: a contest between satisfaction and temptation. In *Proceedings of the sixth ACM international conference on Web search and data mining*, WSDM '13, pages 395–404, New York, NY, USA, 2013. ACM.

[267] S. Zhai, P. O. Kristensson, P. Gong, M. Greiner, S. A. Peng, L. M. Liu, and A. Dunnigan. Shapewriter on the iphone: from the laboratory to the real world. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '09, pages 2667–2670, New York, NY, USA, 2009. ACM.

[268] C. Zhang, X. Ding, G. Chen, K. Huang, X. Ma, and B. Yan. Nihao: A predictive smartphone application launcher. In D. Uhler, K. Mehta, and J. Wong, editors, *Mobile Computing, Applications, and Services*, volume 110 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 294–313. Springer, Berlin Heidelberg, 2013.

[269] J. Zhuang, T. Mei, S. C. H. Hoi, Y. Q. Xu, and S. Li. When recommendation meets mobile: contextual and personalized recommendation on the go. In *Proceedings of the 13th international conference on Ubiquitous computing*, UbiComp '11, pages 153–162, New York, NY, USA, 2011. ACM.

[270] M. Ziefle and S. Bay. Mental models of a cellular phone menu. comparing older and younger novice users. In S. Brewster and M. Dunlop, editors, *Mobile Human-Computer Interaction - MobileHCI 2004*, volume 3160 of *Lecture Notes in Computer Science*, chapter 3, pages 25–37. Springer, Berlin, Heidelberg, 2004.

[271] M. Ziefle, U. Schroeder, J. Strenk, and T. Michel. How younger and older adults master the usage of hyperlinks in small screen devices. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '07, pages 307–316, New York, NY, USA, 2007. ACM.

[272] A. Zimmermann, A. Lorenz, and R. Oppermann. An operational definition of context. In B. Kokinov, D. C. Richardson, T. R. Roth-Berghofer, and

L. Vieu, editors, *Proc. CONTEXT*, volume 4635 of *Lecture Notes in Computer Science*, pages 558–571, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[273] I. Zukerman and D. W. Albrecht. Predictive statistical models for user modeling. *User Modeling and User-Adapted Interaction*, 11(1):5–18, 2001.

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

Matthias Böhmer
Saarbrücken, den 16. Juli 2013