Designing with Data: Using Analytics to Improve Web and Mobile Applications

Jussi Ahola

University of Tampere
School of Information Sciences
Interactive Technology
Pro gradu thesis
Supervisor: Jaakko Hakulinen
November 2014

University of Tampere
School of Information Sciences
Interactive Technology
Ahola, Jussi: Designing with Data: Using Analytics to Improve Web and Mobile Applications
Pro gradu thesis, 81 pages
November 2014

This thesis looks at the ways in which software analytics can be used to gather data on web and mobile application usage. The main goals of the study were to find out how the data collected with analytics can help in improving these types of applications and to place analytics into the group of user research methods available to an HCI researcher.

The first five chapters form the theoretical part of the thesis. These chapters discuss the methodological foundations of analytics in sociological and psychological research, place analytics into the automated data collection tradition in HCI research, chart the technical and strategic details of how the data is best collected, and compare the strengths and limitations of analytics data with those of other user research methods.

The research part of the thesis is based on work done on three applications, two of which were mobile applications and one a web application. These three applications were treated as case studies that exemplify the ways in which analytics can be used to improve software applications.

The results showed analytics to be an extremely useful research method for an array of research questions. The collected data revealed several potential points of improvement in the studied applications. Furthermore, the low cost and good availability of different analytics solutions was found to make it a method that any HCI researcher or designer with an access to a publicly deployed application can add to their toolbox.

Keywords and phrases: analytics, behavioural data, quantitative data, web application, mobile application.

Table of contents

1. Introduction

This thesis explores the ways in which modern analytics solutions and the data collected with them can be used to improve web and mobile applications. The theoretical part of the thesis provides some background into the subject and places analytics as a research methodology into the sociological and psychological research traditions. The research part of the thesis is based on data collected from three case studies, which exemplify some of the possible approaches into how analytics data can be used in improving these applications.

Let us assume for a moment that we run a news service directed towards the general public. Similarly to many other successful services today, we want our users to be able to use the service with whatever device they happen to have and hence serve them in several digital channels: we have put plenty of effort in designing and developing a modern web application and mobile applications for all major mobile operating systems.

How do we know how many active users each of these channels has? Which navigation links from the main view are the users most likely to follow? Do the users read more articles per session with one of the mobile applications or with the web application? Which features do they use and in which order? Is the sports section more popular than the business section? How often do the users stop the purchase process for paid articles before completing it? On which step of the purchase process are they abandoning it? Which one of the two design alternatives that we have been pondering over would lead to more loyal users? And, most importantly, how could we use the answers to these questions to improve our applications?

To be able to answer questions such as these, we need detailed data on how users are interacting with our applications. There are many possible ways to collect these types of behavioural data, but the present study employs a method that has lately been receiving

plenty of attention especially in the industry: software analytics. Along with the buzz around analytics, the current interest in measuring behavioural data is also highlighted by the hype around some related and lately emerged concepts and terms such as *big data*, *web intelligence*, and *business intelligence*.

A subset of software analytics, web analytics, has been defined as the "measurement, collection, analysis, and reporting of Internet data for the purposes of understanding and optimizing Web usage" (Web Analytics Association 2008). Since the scope of the present study involves collecting data not only from the web, but also from mobile applications, a broader definition is required. As it is used in this thesis, *analytics* refers to 'the partly automated collection, storage, analysis, and reporting of human-system interaction events using a publicly available and hosted solution.' In that definition, *partly automated* refers to the fact that once suitable instrumentation is in place, 'the data is collected without any effort from the researcher and the storage and some analysis of the data are typically done automatically by the analytics software.' These analytics solutions are referred to as *hosted* as the data are saved to a database and accessed through an interface 'hosted on the analytics vendor's servers.' *Human-system interaction event* refers to 'a human behaviour directed towards a system's user interface and the system's feedback towards the human.' For the purposes of this thesis, *system* will refer to a 'web or mobile application,' as the data for the thesis was collected from these types of applications.

HCI as a field rests on a multidisciplinary foundation. Many of the methods used in HCI research and design are based on work done in the fields of human factors, engineering, psychology and sociology, for example. Analytics makes no exception to this tradition: the interaction events recorded using analytics are, of course, human behaviours, which, as the name suggests, have historically been studied in the field of behavioural sciences. Other user research methods in HCI, such as interviews, ethnographic observation, surveys, and experimental research also draw from this multidisciplinary background. With the help of this diverse toolbox drawing from several disciplines, HCI researchers today are in a better position to understand complex socio-technical systems (Shneiderman 2008, 1350). Hence one of the goals of this thesis is to place analytics in

this toolbox available to HCI researchers and designers, and to compare and contrast its strengths and weaknesses with those of other user research methods.

Because the development of the majority of web and mobile applications is commercial by nature, it is only natural that most current analytics research is carried out from the business, rather than academic, perspective. Furthermore, most of the advances in the field are occurring in the practitioner side (Jansen 2009, 2). For these reasons, the terminology used in the field is also mostly coined and defined in the industry rather than the academia: typical jargon includes terms such as *customer* and *A/B test*, whereas corresponding concepts in academic HCI research would most often be denoted by *user* and *between-subjects experimental design*. In this thesis, terminology from both sides of the divide will be employed with the goal of also linking matching terms together where appropriate.

With the help of three case studies, two of which concern mobile applications and one a web application, this thesis aims to study the possible ways in which analytics can be employed as a user research method in Human-Computer Interaction (HCI) research. Before analysing the actual data from these case studies, Chapters 2 to 5 lay out the theoretical framework on which the data analysis rests on: Chapter 2 discusses the methodological foundations of analytics found in the fields of sociology and psychology; Chapter 3 places analytics into the automated data collection tradition in HCI research; Chapter 4 delves into the nuances of how analytics data are collected; Chapter 5 lays out the benefits and limitations of these type of data. The research procedures and data analyses from the three case studies will be presented in Chapter 6, while the conclusion will be left to Chapter 7.

As the title of this thesis suggests, special emphasis will be placed on the notion of how the research outcomes can be used in improving the applications. *Improvement* is a subjective term that means different things for different people as regards different applications: For the companies behind these applications, it can mean raising the level of user engagement or the number of purchases that are made in the application, which would lead to more revenue for the company. For the users of the applications, improvement can mean more efficient use as manifested in shorter task completion

times or raising the level of enjoyment, which could be measured in more use of the application. Ideally, improvement for one group means improvement for the other, too. The subjective nature of the term *improvement*, however, will be turned into more objective by defining what is meant with it separately for each of the case studies, which then allows for more concrete operationalisations of the concept. The more specific research questions that drove each of case studies will also be defined separately for each of them. The case studies were selected from client cases and personnel's hobby projects done at the software development company where the present author works as a User Experience Designer.

When embarking on a research project on the use of analytics in improving web and mobile applications, one is easily tempted to approach the subject by providing a how-to guide to the use of a specific analytics tool or, at most, a limited set of those tools. The field of analytics at the moment is, however, a highly dynamic one. Because of the large-scale interest in analytics and the dynamicity of the field, new tools for measuring user behaviour in digital environments seem to be appearing by the week. For this reason, a how-to approach to the use of a specific analytics tool would likely become outdated in a matter of years, if not months (Jansen 2009, vii).

Though tools and methods are changing fast, the principles behind recording and analysing behavioural data in digital environments are, if not ever-lasting, at least more enduring. By understanding these principles, the researcher will remain in a better position to understand and make use of the ever-changing field of methods. This difference between principles and methods was eloquently worded by the 19th century essayist Ralph Waldo Emerson:

> As to methods there may be a million and then some, but principles are few. The man who grasps principles can successfully select his own methods. The man who tries methods, ignoring principles, is sure to have trouble. (Emerson, source unknown)

With this emphasis on the principles rather than methods and tools, my hope is that this thesis will provide some enduring value that stretches beyond the release of yet another analytics tool or an update to any of the existing tools.

2. Methodological foundations

This chapter presents the theoretical foundations of using behavioural trace data in HCI research and places analytics as a research method into the psychological and sociological research traditions, in which these types of behavioural data have been used. First, I will discuss the notion of a *behaviour* as it used in this thesis. Secondly, the concept of *trace data* as evidence of past human behaviours in physical settings will be introduced. Finally, the concept of trace data will be extended from physical settings to digital settings.

2.1. Defining behaviours

Using observations of human or animal behaviour as scientific data stem from a psychological research approach called *behaviourism*, which was advanced in the 19th and 20th centuries by such prominent names as Ivan Pavlov and B. F. Skinner. Behaviourism stresses the outward behaviours of organisms as a source of scientific evidence (Skinner 1953). With the emphasis on the concept of a *behaviour*, this construct demands for a more precise definition as it is used in the present thesis.

For the purposes of analytics research, Jansen defines a *behaviour* as ”an observable activity of a person, animal, team, organization, or system,” but also goes on to state that such a broad definition renders the term somewhat overloaded. To flesh out the term in more detail, three separate categories for behaviours are offered:

- Behaviours are something that can be detected and, therefore, recorded
- Behaviours are an action or a specific goal-driven event with some purpose other than the specific action that is observable
- Behaviours are reactive responses to environmental stimuli (Jansen 2009, 9).

A behaviour can hence be understood as an event in terms of its behavioural characteristics (Sellars 1963, 22); one behaviour can be distinguished from another by comparing their behavioural components. Besides behaviours, there are, however, two other types of variables that a study relying on behavioural data should address: contexts and subjects (Jansen 2009, 9). If we put behaviours in a central role among these three types of variables, we can see both the contexts and subjects as parameters to the observed behaviour: an observable *behaviour* of the *subject* of interest in a specific *context*. As we will see, analytics records not only the behaviours, but also contextual and subject-related data on those behaviours.

Several scholars who have studied users' behaviours with interactive systems have built taxonomies of behavioural patterns. Hargittai (2004) provides a taxonomy of web browsing behaviour that includes categories such as directly accessing a URL, use of browser features, and use of search engines, while Jansen and McNeese (2005) classified behaviours related to online search with the help of a taxonomy, part of which is presented in Table 1:

Behaviour	Description
Relevance action	Interaction such as print, save, bookmark, or copy
Relevance Action: Bookmark	User bookmarked a relevant document
Relevance Action: Copy Paste	User copy-pasted all of, a portion of, or the URL to a relevant document
Relevance Action: Print	User printed a relevant document
Relevance Action: Save	User saved a relevant document

Table 1. Part of a taxonomy of behaviours related to online search behaviours (Jansen and McNeese 2005, 1494).

These taxonomies can be helpful in classifying and coding behaviours as long as the categories are discrete and do not overlap with each other.

The spectrum of behaviours that are of interest to HCI researchers varies in their context and subject parameters. In their spectrum of HCI events, Fisher and Sanderson (1994, 260) list potentially interesting behaviours from single UI events, which last a few milliseconds, to project events, which might last years and involve several users. To be able to decipher these longer behavioural patterns that involve several users from low-level user-system interaction events, some abstraction is required. Hilbert and Redmiles present the following ladder of abstraction levels that can be formed from low-level behaviours:

Goal/Problem-Related
(e.g., placing an order)

Domain/Task-Related
(e.g., providing address information)

Abstract Interaction Level
(e.g., providing values in input fields)

UI Events
(e.g., shifts in input focus, key events)

Input Device Events
(e.g., hardware-generated key or mouse interrupts)

Physical Events
(e.g., fingers pressing keys or hand moving mouse)

Figure 1. Levels of abstraction in user behaviours, adapted from Hilbert and Redmiles 2000, 394)

Physical behaviours lay on the bottom level while goal and problem-related behaviours are several levels of abstraction upwards on the top of the ladder. By abstracting and aggregating the low-level behaviours into larger chunks, behavioural data can be used to study even multi-year project events, as described by Fisher and Sanderson (1994, 260).

2.2. Trace data

To be able to study human behaviours, we need some way of gathering evidence of which behaviours have taken place. Behavioural data gathered after the behaviours in question have taken place have been termed as *trace data*, a term that points to traces that are left behind from past human actions (Jansen 2009, 2). The historical basis of research based on trace data can be assigned to the sociological tradition. In sociological research, these traces have traditionally been of physical nature: as people go about in their daily lives, they often leave behind traces to the physical surroundings in which they perform different actions. These traces can appear in the form of wear on some objects, marks, trash, or as reductions in the quantity of some materials. The traces can then be treated and studied as evidence of behaviours that have taken place. In contrast to data produced by methods such as questionnaires or interviews, trace data in a physical setting are not intentionally formed by the test subjects to function as a basis of comparison and research, but rather available for more opportunistic exploration after the behaviour under study has taken place (Webb et al. 2000, 36).

Trace data can be classified into two broad categories: erosion and accretion data (Webb et al. 2000, 36). Erosion data measure the wear on some already existing material, such as the natural wear caused to the floor material of a public building, whereas accretion data measure the increase of some material, such as the amount of trash that is left behind after a public event. In the examples mentioned above, both the erosion caused to the floor and the accretion of trash are examples of trace data that have been formed naturally, meaning that the researcher has not made any changes to the environment under study or intervened in the data collection process in any way.

As opposed to natural trace data, erosion and accretion data can also be collected in a more controlled fashion: the researcher can intervene to make some changes to the environment in order to facilitate the build-up of relevant data. These approaches have been termed as controlled erosion measures and controlled accretion measures (Webb et al. 2000, 43-44). For example, if the floor material of the public building wears off in a way that is unsuitable the purposes of the research, ideally the floor material could be changed into something that speeds up the accumulation of the traces, makes a distinction between traces produced at different times of the day, or makes the traces

produced by different individuals stand out better. However, if the experimenter intervenes in the data collection in this way, care should be taken that few of the most significant benefits of using trace data, its unobtrusiveness and non-reactiveness, are not compromised. According to Webb et al. (2000, 43), the subjects should not be permitted to become aware of the researcher intervention when controlled erosion or accretion measures are used to collect data; this, however, raises some ethical issues, which will be discussed in more detail in Section 5.4.

If the behaviour that has taken place in a physical setting can be investigated by studying the traces that have been formed into that setting, how does one go about in studying behaviour that takes place when a user is interacting with a web or mobile application? This issue will be explored in the next section.

2.3. Instrumenting – collecting trace data in digital settings

A vast array of methods to study user behaviour has been applied in the history of HCI research: ethnography, user observation, and laboratory-based usability studies, among many other methods, can be used to gather this sort of data. These approaches, however, have their drawbacks. As Lazar et al. (2010, 308) note, "[t]iming user task completion with a stopwatch, furiously writing notes describing user interactions with software systems, coding notes from ethnographic observations, and many other tasks are laborious, time-consuming, and often, – as a result – error-prone." Furthermore, the data gathered with these methods cannot be described as trace data, as they are not formed directly by the users themselves, but rather by the researcher who, in one way or another, is observing the user.

In physical settings the subjects form the traces directly into the environment when they interact with it. In the field of HCI, however, our primary interest is in interactions that take place in digital settings. In digital settings, the traces must be of digital nature, too, and might not be formed directly by the subjects themselves, but rather by some sort of a tool designed to record the traces (Jansen 2009, 13). As concerns the division of trace data into the two broad categories of erosion and accretion, the data collected from user-

system interactions clearly fall under the accretion branch: luckily the systems that we study with the help of analytics do not wear off as users interact with them, but rather new material accretes.

As was noted above, trace data in a physical setting can often be studied opportunistically after the behaviour under study has taken place; when new interesting research questions arise, it may be possible to study these in retrospect by using the physical trace data that have been formed without any researcher intervention. In digital settings, however, no trace data can be studied fully opportunistically in retrospect since some preparation is always required beforehand to capture the data; in digital environments, traces do not form naturally. In order to gather any digital trace data, the environment needs to be altered by integrating it with some method of collecting the data. The practice of augmenting software with a method of collecting digital trace data is referred to as *instrumenting* (Lazar 2010, 321). As Münch et al (2014) note, software applications instrumented with these data collection tools capture user-system interactions and send data on them to a database for storage and later access.

The practice of instrumenting applications places analytics, along with all other methods of collecting digital trace data, under the label of controlled accretion measures: a researcher or a developer intervenes in the data collection process to influence which data accrete and how they accrete. As was mentioned above, researcher intervention can jeopardize the benefit of non-reactiveness of trace data by making the subjects aware that the traces of their behaviour are being observed. In the case of analytics, however, the effects of researcher intervention can be minimised, as in most cases the alterations to the applications under study are effectively invisible.

An analogy between some digital trace data and opportunistically available natural trace data, however, is not too far-fetched. The servers on which applications are running collect some log data for the purposes of application testing and development in any case: these raw log data may be available for design research purposes also, but their usefulness for this purpose varies. Most web and mobile analytics tools also collect some trace data with very little instrumenting by the researcher: often a few lines of code added to the application source code are enough to capture these. When collected

with analytics solutions that are based on page tagging, the subject of this thesis, these data often include metrics on the number of sessions, number of page views, number of users, and the length of use times, among many others. Through the analogy, these foundational metrics could be considered data that is available for somewhat opportunistic exploration with very little alteration to the environment. For more refined metrics, a deeper integration to collect data from different user-system interaction events is needed. The difference between foundational metrics and more refined metrics, as well as the technical quirks of instrumenting software with analytics tools, will be explored in more detail in Sections 4.1 and 4.2.

As has been discussed in this chapter, conceptually the research based on trace data collected with the help of modern analytics solutions follows a research tradition with its roots in psychology and sociology. The other historical consideration, however, can be found in some automated data collection methods prevalent in earlier HCI research. In order to place the methodology used in this thesis into the automated data collection tradition, some these automated methods will be introduced next.

3. Historical roots of analytics – automated data collection in HCI

This chapter introduces some automated data collection methods used in HCI research and places analytics into the automated data collection tradition. As Lazar et al. (2010, 308) note, "[t]he very computers that are the subject of our research are also powerful data collection tools." This resource has been in used in many creative ways to efficiently collect traces from user-system interactions.

3.1. Server log file analysis

As concerns trace data collected from web pages, modern web analytics solutions were preceded by server log file analysis (also called *transaction log analysis*, TLA). Whenever a client, i.e. a user accessing the server through an interface, makes a request to the web server, its access log is updated with a new entry. From the HCI research perspective, useful information in a log file entry can include the IP address of the device that made the request, timestamp, type of the HTTP method that made the request (often GET or POST), resource that was requested, referring page, along with the make and version of the browser that made the request. Most web servers use one of the standardized web log formats: few of the most popular standardized formats are the NCSA Common Log, NCSA Combined Log, NCSA Separate Log, and W3C Extended Log (Jansen 2009, 25). An example of the NCSA Combined Log format entry is given in Figure 2:

```
94.123.123.12 - - [29/May/2014:04:41:05 -0700] "GET /about.html HTTP/1.1" 200
11642 "http://www.example.com/ " "Mozilla/5.0 (Macintosh; Intel Mac OS X
10_9_2) AppleWebKit/537.75.14 (KHTML, like Gecko) Version/7.0.3
Safari/537.75.14"
```

Figure 2. Example of a NCSA Combined Log format produced by an Apache HTTP server (Apache Software Foundation).

These standardized log entry formats can often be customized to include some custom fields, which can be useful in extending the use of server log files to specific research needs (Lazar et al. 2010, 310). With access to the log files, the researcher can mine textual data from them to be transformed and analysed with the help of a spreadsheet application, for example. Freely or commercially available software for the analysis and visualization of log file data include tools such as AWStats and Sawmill.

Data collected from server log files have been used, for instance, to build website navigation maps, to study issues related to website usability, and to empirically capture task completion times of link selections (Lazar et al., 311-313). As concerns web applications, log file analysis has some benefits that analytics solutions based on page tagging do not have: server log files do not rely on JavaScript and cookies to work. Even if a user has disabled JavaScript from their browser and denied cookies from being saved, a request to the server is always logged.

However, the shortcomings of log files in analysing modern web application usage surpass their benefits. Firstly, for many research questions the fidelity of the data that can be obtained from log file analysis is simply not high enough. Furthermore, some technical changes in the way that highly interactive web applications are built today have rendered log file data less useful for the study of user-system interactions. Many of the earlier web applications were simple collections of page layouts that used the request/response paradigm of the HTTP protocol to move between pages and to populate them with the content that the client requested; the server was contacted whenever the user interacted with the application and hence a trace of the interaction was left to the server log files. In the development of today's highly dynamic web applications with sophisticated interfaces, however, there is a tendency to move much of the interaction to the client side (Atterer et al. 2006, 203). On the web, these applications are to a large extent JavaScript-based and make requests to the server only when there is a need to save or load data. Modern techniques and architectures such as AJAX (Asynchronous JavaScript and XML) and SPA (Single-Page Application), which aim for a more fluid experience for the user, are based on the premise that as much of the source code as possible is retrieved from the server with a single load and much of the application logic and interactions are shifted from the server to the client. When a

user-system interaction takes place entirely on the client, the server is not contacted at all and a user behaviour analysis based on server log files will fail. Leiva and Vivó (2013, 2) note that server logs are enough to quantify some aspects of web browsing behaviour, but higher-fidelity research into user-system interactions also requires studying the client side.

3.2. Instrumented and custom-built software

Observing people using complex software, such as word processing and spreadsheet applications with hundreds of possible button, menu, and keyboard shortcut selections can be a daunting task. Questions such as which of the selections are most used and how possible redesigns would affect the usage may be impossible to answer with qualitative user observation data, which, for practical reasons, often has a limited sample size (Lazar 2010, 321).

To collect data to answer such questions, researchers and developers working on complex applications have built custom data collection tools into those applications; the application is collecting data on its own use to a database maintained by the application developers themselves. Applications furnished with these custom data recording tools are known as *instrumented software*. With the help of an instrumented version of a given application, traces of all user-system interactions that are of interest can be stored into a log file or a database maintained by the developers of the application. Though conceptually the notion of self-recording software is extendable to web and mobile applications and it is not, in fact, too far from modern analytics solutions, in the literature the term *instrumented software* seems to refer especially to desktop applications: Harris (2005) describes a research effort with an instrumented version of Microsoft Office suite, while Terry et al. (2008) used an instrumented version of the popular open-source image manipulation application GIMP. Data from the former were used to inform the design decisions that went into the release of a new version of the application suite, while data from the latter was used to detect and fix usability issues that might not have been detected without them.

Besides instrumented versions of commercial or open-source software products, researchers have built instrumented software solutions with their only purpose being that of running a scientific experiment. These efforts do not aim at studying the use of a specific application, but rather at shedding light on some more general characteristics of the interaction between humans and technology. The well-known concept of Fitts' law (Fitts 1954), for instance, has been studied using custom-built software that tracks selection times for targets of varying size and distance from each other. The accuracy of such software in recording the selection times far surpasses that which any human could record manually.

Whereas building an instrumented application from the scratch requires plenty of technical expertise and resources, commercial analytics solutions are easier to set up and do not require as much developer resources. The notion of instrumenting as adding user-system interaction recording capabilities into an application, however, extends well into commercial analytics solutions too.

3.3. Using web proxies to record interaction data

Though originally web proxies were designed to improve bandwidth usage and web browsing experience inside an organisation's network, they have been used in creative ways for HCI research purposes. A web proxy functions between the client and the server: it receives all the request that the client makes, passes them on to the server, receives the server's responses, and passes them on to the client. What is important for HCI research purposes is that the proxy can also modify, first, the client's requests before they are passed on to the server and, second, the server's responses before they are passed back to the client.

UsaProxy (Atterer et al. 2006) is a proxy-based approach to collect detailed user-system interaction data from the web. All HTML data that is passed from the server to the client is modified with additional JavaScript code that tracks all interactions with the Document Object Model (DOM) elements on the webpage. The JavaScript then sends the trace data that is captured on these interactions to the proxy server, which stores it

into a database for further processing. With this approach, a variety of different low-level user interactions can be recorded: for instance window resizes, mouse clicks, mouse hovers, mouse movements, page scrolls, and keystrokes can all be recorded along with the cursor coordinates on which these interactions took place and with mappings of these coordinates to the DOM elements on the webpage (Atterer et al. 2006, 208).

Web proxies provide a powerful way to collect data not from a single web site or web application, but from all of the sites and applications that a user who is connected to the web via the proxy visits. Proxy-based data collection is, then, user-centric: whereas analytics solutions based on page-tagging and all the methods described in this chapter focus on a specific site or application, and could hence be called application-centric, a proxy focuses on a single user or a restricted set of users and collects data from all the web sites and applications they access. This can be either a good thing or a bad thing depending on the type of the research project: a proxy-based approach cannot provide great results if the goal is to learn about user interactions on a specific application, but might work well if the goal is to learn about the behaviour of a set of users more generally. There are, however, some practical issues with this approach. Most importantly, collecting data truly from the wild with a proxy is difficult: the user who is tracked must be either physically connected to a network from which all traffic to the Internet is passed through a proxy or they must willingly connect to a proxy server before connecting to the Internet. The user knowing about them being tracked can introduce some evaluation artefacts into the research.

Having presented the behavioural and sociological foundations of using trace data in HCI research and discussed the automated data collection tradition on which analytics research rests on, I will now turn to the nuances of how this data is collected, what are its main benefits and limitations, and the methodology surrounding it in more detail.

4. Data collection for analytics – how analytics works?

This chapter introduces the technical details of how analytics data is collected from web and mobile applications and charts some strategies for conducting an analytics research project.

4.1. Web application analytics

As regards web application analytics, a sub-section of all analytics, the methodology used in this thesis is based on what is known as *page tagging*. These page tagging analytics tools are commercially or freely available and act as hosted solutions: the data that is recorded from any user-system interactions are sent to the analytics vendor's servers for storing and further processing. These data, often along with some charts and graphs generated from them, can then be accessed through an interface provided by the analytics vendor.

The page tagging approach relies on JavaScript and cookies to work. After performing all the necessary administrative actions of setting up an account on the analytics vendor's site, the developer of the application adds a snippet of JavaScript code into all of the HTML pages on which user interactions are to be tracked; practice varies, but most analytics vendors recommend this code to be added inside the head element of the HTML document. An example of Google Analytics' tracking code is shown in Figure 3:

```
<script>

     (function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function
     (){
     (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new
     Date();a=s.createElement(o),
     m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefor
     e(a,m)
     })(window,document,'script','//www.google-
     analytics.com/analytics.js','ga');

     ga('create', 'UA-43783099-1', 'jussiahola.com');
     ga('send', 'pageview');

</script>
```

Figure 3. Example of a Google Analytics JavaScript tracking code (Google 2014b).

This tracking code then sends information on visits to the site to a database located on a server hosted by the analytics vendor. The data that is sent to the database with this simple integration of the tracking code are often called *foundational metrics* (Jansen 2009, 35) and typically they include information on:

- Which HTML page the user accessed
- When it was accessed
- The referring URL, i.e. the HTML page where the user came from (links or search engine referrals)
- IP address
- Technical details on the hardware and software of the client, such as the make and version of the browser, operating system, and screen resolution (Beasley 2013, 26).

If the user is accessing the web application on a mobile device such as a smartphone or a tablet, some mobile-specific metrics may also collected as part of foundational metrics. Typically these include:

- Device name
- Device type
- Carrier network (Dykes 2013)

Furthermore, the GPS sensors prevalent in modern mobile devices can provide more accurate location data than is available through IP address analysis (Dykes 2013).

The JavaScript code also stores a persistent cookie in the web browser's memory. The cookie is used to identify whether data from a given user has been recorded before or not; many analytics vendors classify these users as *returning visitors* and *new visitors*, respectively. Furthermore, as the user interacts with the web application during a use episode, the cookie is used to group the recorded interaction events as belonging to the same use episode; in analytics jargon the use episode is often denoted by the word *visit*. When no interaction events are recorded from the same user during a certain period of time, the analytics vendor will regard the use episode as having ended and groups the data accordingly in its reports.

Figure 4 shows how some of the foundational metrics are reported in Google Analytics' reporting interface:

Figure 4. Example of some foundational web application metrics as they are reported in Google Analytics' reporting interface.

However, the foundational metrics recorded with this simple addition of a JavaScript code snippet inside the head element of a HTML page are not much better than what a careful server log file analysis can tell; this approach provides only page-to-page tracking, meaning that data only from one type of interaction events, movements from an HTML page to another, are recorded. As has been noted above, this is problematic for tracking user interactions in dynamic web applications in which much of the

interaction takes place on the client side and most interactions do not result in a new HTML page load. An interaction event can be defined as almost anything that a user does with a system and hence the whole spectrum of possible interaction events encompasses much more than just those events that result in movement between web pages. Interactions with any Flash elements and dynamic HTML elements, for instance, do not necessarily result in page loads and hence are not captured with page-to-page analytics.

With the page-tagging approach the way around this issue is to add tracking to those elements on a web page with which users can interact and which do not result in new page loads. Often called *in-page tracking* (Beasley 2013, 187), with most analytics vendors this happens by instrumenting the desired elements with additional JavaScript code which records the interactions and sends data on them to the analytics vendor's server whenever the element is interacted with. This allows the researcher to collect data on interactions with all DOM elements on a webpage regardless of whether interacting with them results in a new page load or not. Figure 5 shows an example of an HTML *button* element instrumented with analytics tracking:

```
<button class="play_video" onClick="_gaq.push(['_trackEvent', 'Videos',
'Play', 'Cat Video']);">Play</button>
```

Figure 5. Example of a Google Analytics JavaScript in-page event tracking code attached to an HTML *button* element (Google 2014a).

A research project into how a video player embedded on a website is interacted with makes a good example of how in-page tracking can be made use of: For commercial reasons, the developers of the site might be interested in which videos were played, if they were watched into completion, and when they were stopped (if not watched to completion). Instrumenting the appropriate interaction events, such as the users clicking the play, pause, and stop buttons, along with the event of the video running to the end, can provide answers to these questions (Beasley 2013, 192).

4.2. Mobile application analytics

As regards mobile applications, it is first important to make a terminological distinction between *mobile web* and *native mobile applications*: *Mobile web* quite simply refers to websites that are accessed using a web browser on a mobile device such as a smartphone or a tablet. As the support for JavaScript and cookies on these devices is nowadays widespread, tracking web usage on mobile devices falls into the JavaScript page tagging approach outlined in the previous section. *Native mobile application*, however, refers to an application that is typically written in a programming language such as C, Objective-C, or Java and which the users normally download and install into their mobile devices from an application store maintained by operating system manufacturers such as Google, Apple, and Microsoft. Analytics related to these native mobile applications is the topic of this section.

Conceptually mobile application analytics do not differ from web application analytics in any fundamental aspect: they are also hosted solutions and, with appropriate instrumentation, collect behavioural trace data in the form of user-system interactions, such as movements between different sections of the application, button taps, and swipes on a touchscreen. Similar to web analytics, foundational metrics such as the number of users, number of sessions, session lengths, and rough geographical location of users can be obtained with relatively simple analytics instrumentation, whereas tracking more fine-grained data on screen views and button taps, for example, requires instrumenting the corresponding interface events. The terminology related to similar concepts can, however, differ: As mobile applications do not have pages in the same sense as websites do, navigation between different sections of the application is denoted by *screen views* as opposed to *page views* and hence the more fine-grained tracking within views could also be termed *in-view tracking* as opposed to *in-page tracking*. In the same sense mobile application use episodes are often termed *sessions* in contrast to the term *visits* in web analytics (Dykes 2013). Some details are also different, such as unique users being identified using the device ID as opposed to cookies in web analytics, and mobile application analytics not being limited by browser-related security restrictions.

Figure 6 shows how some of the foundational metrics are reported in Flurry Analytics' reporting interface:

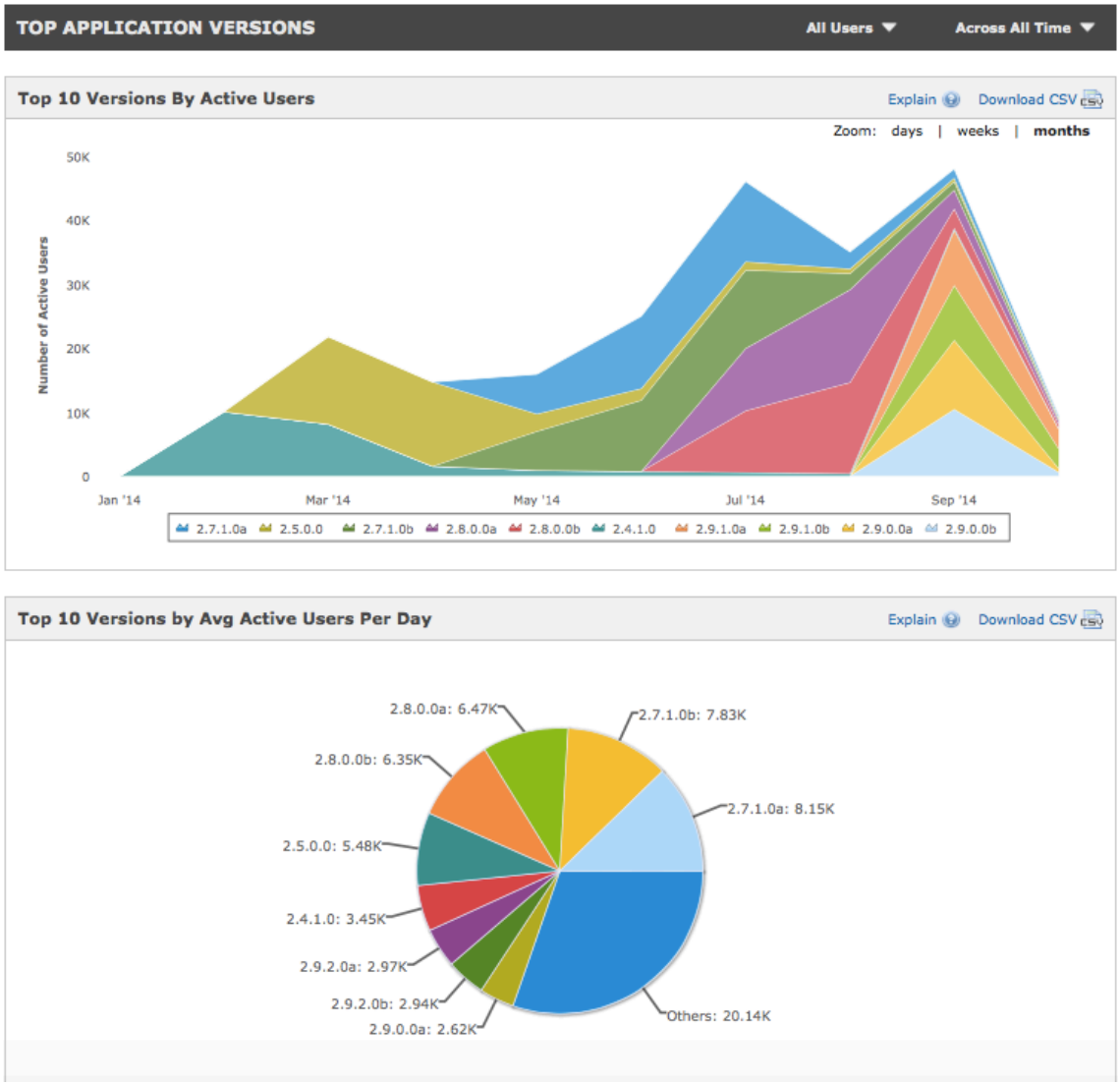

Figure 6. Example of some foundational mobile application metrics as they are reported in Flurry's reporting interface.

Whereas from a technological perspective web analytics relies primarily on JavaScript and cookies to collect interaction data, mobile application analytics are built on an altogether different technological framework. Analytics vendors have built pre-written libraries of code for different mobile device operating systems in their respective native programming languages; application developers can then include these libraries, or software development kits (SDK's) into the source code of their applications and use

the pre-written code in the SDK to record interaction events and send data on them to the vendor's servers for processing and later access. Figure 7 gives an example of how the Windows Phone SDK from Flurry Analytics can be used to record interaction data:

```
public void PlayVideo() {
     FlurryWP7SDK.Api.LogEvent("play_video");
     //Perform actions needed to play video
}
```

Figure 7. Example of a Flurry Analytics event tracking code that the developer can call after attaching the Windows Phone SDK into their WP application (Flurry 2012).

Despite these slight differences between web analytics and mobile application analytics, mobile application analytics are less about a whole new concept and more of applying the same concept to a new environment (Beasley 2013, 228).

4.3. Analytics research approaches – from structured to unstructured

The data gathered with analytics can be used to answer a variety of different questions, some of which are highly specific: How many times was this section of the application accessed? How many unique users pressed this button? What is the percentage of users using the application one week after installing it? These specific questions can also be hypothesis-driven: they can function as a part of research into the usage of an application or as a part of an experiment aiming at measuring the changes brought about by some design alterations, for example.

On the other hand, analytics data can be used for more open-ended exploration of how users interact with an application. An open-ended exploration might simply mean going through the analytics reports, pursuing the logic of how certain numbers were formed, and attempting to spot interesting trends and emerging patterns from the data.

The former of these approaches equals a highly structured effort to address a well-defined question with a concrete answer, whereas the latter is an unstructured and

perhaps even an unfocused traverse in the data with no attempt at an answer to any specific question. Beasley (2013, 14) suggests that these two extremes are best thought of as the ends of a continuum:

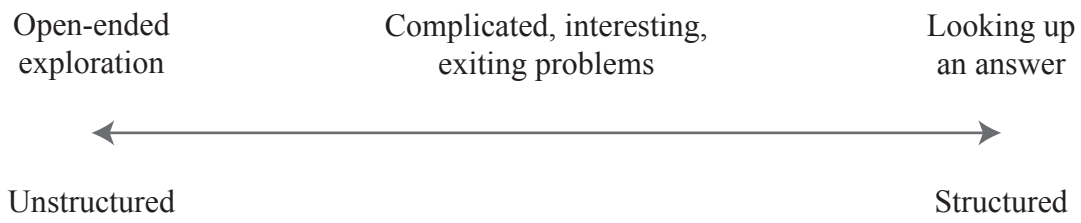

Figure 8. The range of analytics research from completely unstructured to highly structured. Adapted from Beasley (2013, 14).

Harris (2005) describes a data collection effort using an instrumented version of Microsoft Office application suite which was more of an open-ended exploration than a hypothesis-driven study or experiment: "[i]n short, we collect anything we think might be interesting and useful as long as it doesn't compromise a user's privacy." Some of the pitfalls that this approach can have manifested themselves in the post-processing of the data that was collected: as much as 70% of all the data points that were gathered from about 1.3 billion use sessions were discarded. As Lazar et al. (2010, 324) note, gathering data from all possible interactions that can take place inside an application can lead to such a large set of data that gaining insight from it may become difficult.

Another problem with instrumenting all possible user-system interaction events is that this approach requires plenty of instrumentation work and in some cases specific technical expertise. However, the positive side of open-ended exploration is that large sets of data collected from a vast array of interaction events may reveal patterns that would not have been noticed had the focus been only on some specific questions (Lazar et al. 2010, 324). With the effort described by Harris (2005), the team in charge of the Microsoft Office redesign were able to gain valuable insight into the usage patterns of the application suite and base design decisions not on guesswork, but rather on data of how real users interacted with the applications. As an example of these insights, before the data was available, the team was considering to remove the Paste button from the toolbars of the Office suite; their hunches were that only a small number of users were

using the toolbar button as more efficient keyboard shortcuts and context menus were available for the Paste command. What the data revealed, however, was that though the alternative input methods for executing the command were, indeed, more used than the button, the Paste button was the most frequently clicked button on the entire toolbar. This led the team to make the Paste button visually more prominent for an upcoming release (Harris 2006).

Most analytics research, however, falls somewhere between the ends of the continuum from unstructured to structured. If the approach is fully open-ended and unstructured, it may become difficult to put any limits on the time that one spends analysing the data; questions that one wishes to answer with analytics data help to put limits on the time that is spent and structure the research effort. Answering these questions, however, may not be as simple as first thought and in answering them, the researcher may well stumble upon new and interesting data on how users are interacting with the application under study. New observations often give birth to new questions, which, again, may require additional instrumentation and data collection to be answered. This balancing between structured and unstructured approaches to analytics research could be termed as the *semi-structured approach*: the researcher starts the analysis with some questions in mind, but also keeps her eyes open for other insights that emerge from the data. To gain as much value as possible from the semi-structured approach, several iterations of instrumentation, data gathering, and analysis are often needed. The middle ground between the structured and unstructured approaches is where, according to Beasley (2013, 14), complicated and interesting problems reside.

4.4. The semi-structured approach – data collection, transformation, and analysis

Though in reality user interactions in many software applications are continuously recorded and analysed for the purpose of making small improvements to the applications throughout their lifecycles, it might be useful to formalise a small chunk this ongoing activity into a single process. With the help of this formalised process, the actual reality of continuous data collection and experimentation can then be regarded as consisting of several of these processes, which start at different times, overlap, and

finish at different times. While keeping in mind that the reality can seldom be described in a simple chart, Beasley (2013, 14) offers the following model of analysis for a semi-structured analytics research effort:

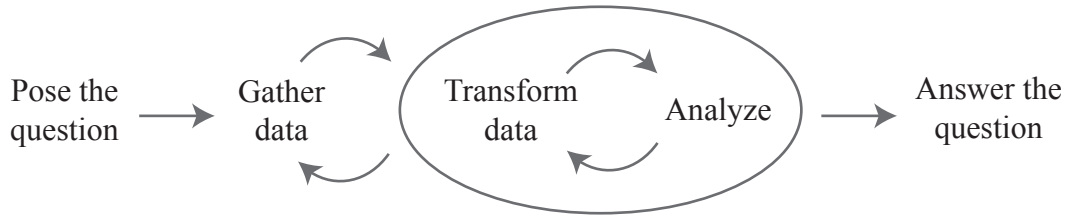

Figure 9. Model for a semi-structured analytics research project. Adapted from Beasley (2013, 14).

Posing the question that one wishes to answer with analytics data helps to give direction on what to measure and puts boundaries on the research. Analytics data can provide behavioural data, which can then be used to answer "what" questions. Attitudinal data, which could be used to answer "why" questions, however, is beyond the scope of what analytics can directly provide. Hence the questions that one wishes to answer with the data should fall into the "what" category; all other types of questions need to be reframed into the "what" format for them to be answered with analytics data. Once it has become clear which data is needed to answer the question, the research project can move on to the next step.

Gathering data requires, first, instrumenting the application that one wishes to study with appropriate code to record user behaviour that could potentially be used to answer the question that has been posed on the previous step. This logging code also takes care of sending the data that has been gathered to the analytics vendor's servers for processing and later access. Sometimes the question can be answered with the so-called foundational metrics, which require very little instrumentation to the source code. The foundational metrics provided by many analytics vendors can be used to answer questions related to the number of users that the application has, their geographic location, and application use times, for instance. On other occasions the instrumentation means attaching logging code to specific user-system interaction events that are related to the question that has been posed.

Instrumenting specific user-system interaction events requires, first, a decision on which events to track. If there is a clear sense of the questions that the in-page event tracking should answer, this process is more hypothesis-driven and the decision is guided by the research question. On the other hand, in-page tracking can also be used for more open-ended exploration to gain an overall picture of how users are interacting with different objects on pages. The former is a more focused approach, meaning that it might suffice to instrument only a few elements with analytics tracking, whereas the latter is an unfocused approach that requires instrumenting a larger number of elements and potentially results in a dataset that is also more unfocused and even confusing.

When appropriate instrumentation is in place, the instrumented version of the application needs to be deployed for real users to use in the wild. Sometimes this involves deploying different versions of the application to different user populations to be able to later study the version's effect on user behaviour. After the deployment has been done and some data been aggregated, the researcher can move on to the second step of gathering data, namely accessing them. At its simplest this could mean navigating to the right report on the web interface of the analytics vendor. For more complex inquiries the data may need to be gathered from various different reports or tools, or may involve downloading thousands of rows of formatted data into a single spreadsheet for further processing.

If the question that has been posed at the start of the process can be answered by simply looking up a number on the web interface of an analytics vendor, for example, there is no need to transform the data at all. Many of the questions that a researcher may want to answer, however, require some sort of a data transformation stage to take place. This can mean filtering out the data produced by the majority of an application's entire user base to be able to answer questions related to only a specific subset of the users; this process is often referred to as *segmentation*. Sometimes the question calls for a metric that is not directly available in the analytics reports, but needs to be derived by combining two or more numbers into a single metric with the help of a spreadsheet. On yet other occasions, the question relates to longer episodes of user interactions: in these cases, separate interaction events need to be combined into and looked at as sequences or patterns of events.

Once the data has been converted into a format that can be interpreted and used to attempt to answer the question, the analysis stage of the process can start. The goal of the analysis stage is to interpret the data so that the researcher is able to tell what it means and what the answer is that it provides to the initial question. However, during the analysis the researcher may find out that the data does not answer the question with enough clarity or fidelity. On the other hand, as a result of the analysis new and more interesting questions may arise. At this point more clarity and answers to the new questions can be gained by going back to transforming the data. These iterations between the initial analyses, new data transformations, and even new rounds of gathering data are at the heart of a semi-structured analytics research project.

Beasley (2013, 17) suggests that the best way to answer the question is to tell a story about it. Depending on who the stakeholders interested in the results are, the format and formality of the story can vary. If the researcher is finding an answer to a question just for her own needs, a mental note might be enough. If, on the other hand, the group of stakeholders interested in the results consists of a larger number of people, such as co-workers, clients or a community of peers, a more official report may be necessary.

Having presented the technical details of how analytics data is collected, it is now appropriate to turn to the nature of this data in more detail and chart its main benefits and limitations.

5. On the nature of analytics data

This chapter explores the main benefits and limitations of using analytics data in HCI research, presents some sources of uncertainties in the data, and discusses some ethical considerations that using analytics gives rise to.

5.1. Benefits of using analytics data

Studying behaviours unobtrusively and retrospectively with the help of digital trace data has some benefits compared to other data collections methods. Some of these benefits will be discussed in this section.

5.1.1. Avoiding or limiting the observer effect

In scientific research in general, *observer effect* refers to the effect that the act of observation has on the behaviour or phenomenon being observed. An example of a research method from the HCI field in which the observer effect can play a surprisingly large role is usability testing in a laboratory setting: the fact that the test subjects are confined into the unnatural environment of a laboratory and observed indirectly through screen and video recording applications or directly by the researcher can have major effects on their behaviour. For this reason, usability testing in a laboratory environment could also be termed a *reactive method*: the test subjects and their behaviour are likely to react in some way to the researcher intervention (Jansen 2009, 6). Huber and Schulte-Mecklenbeck (2003, 231-232) provide an example of the magnitude of the effect that an observer overseeing an experiment in a laboratory environment can have on the results: in an information search task used in their study the participants used about twice as many clicks and twice as much time in the laboratory setting than they did in the wild.

As was noted above, analytics is a fairly unobtrusive method for collecting data from user-system interaction episodes: though the researcher has to alter the environment under study by instrumenting it with a data tracking tool, in the majority of cases these alterations are effectively invisible to the user interacting with the system. Unlike in the case of questionnaires and laboratory studies, the users do not produce the data intentionally for the purposes of the research, but it is rather produced as a side effect of their interactions with the system.

One of the central benefits of using an unobtrusive method such as analytics to collect data is the fact that it can limit or fully avoid the observer effect. When a user of a web or mobile application is not aware that their behaviour is observed, there cannot be any observer effect; when a well-informed user is aware of the increasingly popular practice of using some form of analytics in these applications, this knowledge can potentially have some effect on their behaviour. Indeed, even the knowledge of the fact that servers collect log files as a part of their normal operation can theoretically have the same effect. Even in these cases, however, the observer effect is likely to be negotiable: the fact that analytics is used and log files are collected is often effectively invisible to the user and even the knowledge of some form of data collection taking place in the background is not likely to have major effects on the user's behaviour.

5.1.2. Limiting observer bias

As a specific sub-category of a human tendency known as *confirmation bias*, *observer bias* refers to the bias that the researcher brings into a research situation as they, often unconsciously, try to confirm their own hypotheses and expectations about a given phenomenon. For the sake of example, consider a situation in which a researcher is devising a questionnaire to be filled out by the subjects: the hypotheses and expectations that the researcher might have can significantly influence the way in which the questionnaire is worded and composed. Furthermore, observer bias can influence the way in which the data derived from the subjects is interpreted by the researcher: data that the researcher expected to find can receive unreasonable emphasis, while contrasting data might receive too little emphasis or be completely overlooked (Jansen

2009, 16). Though possible observer bias effects are rarely negotiated in HCI research, in some fields, such as medicine for example, observer bias is addressed by using double-blind measures when interacting with or observing the subjects.

Analytics helps in limiting the effect that observer bias can have on data collection: as the subjects are not specifically responding to any questions or stimuli that the researcher is producing, the hypotheses that the researcher may have cannot bias these. However, observer bias can be present in the data collection in other ways: researchers can, for example, bias the results by choosing to collect data from those user-system interaction events that confirm their expectations and not from those events that may conflict the expectations. When it comes to interpreting the findings of the study, results obtained from analytics data are just as susceptible to observer bias as results from any other type of data (Jansen 2009, 17).

5.1.3. Limiting the effects of Heisenberg's Uncertainty Principle

The Uncertainty Principle was first suggested by the German physicist Werner Heisenberg and is a well-known concept in the field of quantum physics. According to the principle, the "very presence [of a researcher or research instruments] in the environment will affect measurements of the components of that system" (Jansen 2009, 16). In the field of quantum physics, this effectively means that the researcher cannot escape the fact of influencing the behaviour of the particles under study: to know the speed of a particle means measuring the speed; to measure the speed means affecting the speed. Even the light that is used to see the particles has an impact on those particles.

An analogy to some methods used in HCI research is obvious: in ethnographic research, for example, the researcher becomes a part of the system or the group of users using the system (Lazar et al. 2010, 222). The presence of the researcher is likely to have an effect on the behaviour of the "components" of that system. Observations of users interacting with a web or mobile application that are collected using analytics, however, minimise the effects that the Uncertainty Principle has: the researcher does not become

a part of the system that is being studied, but the data is rather collected on the background as users interact with the system. The software under study, however, needs to be instrumented with some source code that handles the recording of these interactions, which can have some effect on the behaviour of the software. Luckily, when this instrumentation is well implemented, the effect that it may have is more theoretical than practical.

5.1.4. Power

HCI research has one benefit which many other fields lack: the devices and systems that we study can also be configured to be used as powerful data collection tools (Lazar et al. 2010, 308). In this context, power refers to the fidelity and sheer number of data points that can be collected with relatively little work. The power with which we can collect digital traces in HCI research as compared to physical traces in sociological research, for example, is where computers as data collection tools shine. In Section 2.2 an example of controlled trace measures in a physical setting suggested that the floor material of a public building could, ideally, be changed into something that records the traces that are formed with greater fidelity: perhaps the researcher would like speed up the accumulation of the traces, know more about the time of the day the traces were produced, or know which trace was produced by which individual. For practical reasons, however, this is often not possible: a floor material that would record the traces at the required level of fidelity might simply not exist.

When the traces are of digital nature and recorded with the help of computers, however, we are not as restricted to a specific level of fidelity. There is a method for tracking almost any type of interaction that the user might have with the system: clicks, swipes, tilts, requested URLs, cursor movements and keystrokes can all be tracked with suitable instrumentation. Furthermore, these interactions can be pointed to a specific user (or at least a specific device; see Section 5.3.1 for a discussion on the unique user problem) and combined with contextual information about the browser, hardware, and location, for instance. A single click does not need to be just a click: it can be a click on a specific element, at specific viewport coordinates, at a certain time of the day, on a specific

operating system running on a specific device in a specific city. We can know what happened before and after that click and post-process the data along any of the trace data dimensions that were recorded and the contextual information related to them. The level of detail of the data that we record can often be configured to be as high as needed. In other words, digital trace data is like the magical floor material described above, with the exception that it actually exists.

5.1.5. Scale

Many of the research methods used in HCI research are limited in terms of the scale that they provide. Studies in a laboratory setting, for instance, are always restricted in terms of sample size, as only a limited set of subjects can be invited to take part in the study, mostly due to budget and time restrictions. The physical location of the laboratory is set, and typically only subjects living close enough to the laboratory will take part in the research. The duration of the data-collection is also often limited, which means that data cannot be collected over an extended time period to study long-term effects and changes in behaviours as users become accustomed to a particular application, for example. The relatively small sample of participants, recruited from a population living close to the location of the laboratory and measured at a specific point in time obviously leads to some restrictions on the generalizability of the findings of the research. HCI research is often carried out with a relatively small subset of all the possible data that there is; the smaller the sample, the more challenging it becomes to make extrapolations to larger populations. Furthermore, the generalizability of results that are obtained in a sterile laboratory environment to real-world settings that can be noisy, busy, and dirty is also questionable. This generalizability of research findings to the world outside of the research is referred to as *external validity*, which has been defined as the "generalizability of an experimental result to a particular real-world population, situation, or setting different from that represented in the experiment" (Martin 2008, 341).

Far too often the participant recruitment for a laboratory experiment in HCI is done, for example, through university mailing lists, which can lead to a biased sample of subjects:

the sample can consist of 20–25-year-old right-handed male Finnish Computer Science majors. This might not be a problem inside the realm of the study itself, but is a serious issue for the generalizability of the findings unless the population that we wish to describe is, indeed, 20–25-year-old right-handed male Finnish Computer Science majors: we can extrapolate the findings only to the population which the current sample is a representative subset of. If the goal is to be able to say something about the general public of, say the entire population of Finland, the sample should be a representative subset of that population. If the subject sample is something else than a representative subset of the entire population that we wish to describe, our sample is biased and the level of external validity low.

Furthermore, in a highly controlled laboratory experiment many of the circumstances that can have an impact on the results are made into control variables: variables that the researcher sets at some desired level and makes sure that they stay at the set level for all subjects. The control variables are set at a specific level in order for the researchers to be able account any variation in the dependent variable by the changes that they brought about in the independent variable. In a laboratory setting the environmental circumstances are typically set as control variables: the lighting, temperature, and number of other persons in the room are controlled. Though in experimental research some control variables are necessary in order to establish a causal relationship between the independent and dependent variables, having too many control variables is a threat to external validity: if all the variables in an experiment are controlled, the researchers create a unique set of circumstances and the results cannot be generalized to any other situation (Martin 2008, 27).

Analytics data escapes many of the disadvantages that are related to the limited scale of some other research methods in HCI research. Firstly, the sample size that can be studied is large. In fact, if the goal is to describe the entire population of the users of a given application, there might not be any reason to draw a sample of this population, as data can be collected from the entire user population with practically no extra cost. When the data is collected from the entire user population, the findings can also be extrapolated to any future users with more certainty. The fact that there is no need to recruit participants to take part in a study can also help in overcoming population

restrictions: it can be practically impossible to get certain subsets of the entire population to take part in a laboratory study, whereas with the help of analytics, data can be collected from all the users of an application. As the sample size of the analytics data can be large, it becomes more likely that the researchers are able to establish statistically significant relationships between different variables. According to Jansen (2009, 18), the amount of data collected from Web transaction logs can be so large that almost all relations are significantly correlated.

Analytics data also escapes the location restrictions related to data collected in a typical laboratory study. When the users of a web or mobile application are spread over a geographically large area and potentially even all over the world, the analytics data that are collected from their interactions with the application are, obviously, geographically distributed. Furthermore, the data are produced not in an unnatural laboratory setting, but rather in naturalistic settings in which the users choose to use a given application: a laboratory environment can never adequately represent the real-world circumstances in which people interact with technology. This sort of data collection has been lately described as research *in the wild*. Research findings that are derived from data collected in the wild can achieve a high level of external validity. As the data are collected from geographically distributed locations, naturalistic physical settings, varying social contexts and are produced using different devices, browsers, and operating systems, the generalizability of the findings to different populations, environments, and contexts can be far better than that of laboratory research where many of the factors mentioned above are made into control variables.

The duration restrictions on the data collection are also not as pressing with analytics as they often are with laboratory-based methods. As there is no need to recruit research subjects, book a research session for them, and reserve a laboratory space and equipment for the duration of all the tests, analytics data can be collected over a much longer time period and possibly even continuously during the entire lifespan of an application (Jansen 2009, 18). This can enable the researchers to answer research questions that require extended and continued interaction between the users and the application. Such research questions might include topics related to the learnability of a user interface or the level continued engagement with an application, for instance.

5.2. Limitations in the use of analytics data in research

No research method is perfect and all types of data have certain restrictions in what they can tell us. This section charts some of the inherent limitations that analytics data have.

5.2.1. Analytics data is bad at telling us the "why"

At the core of analytics data are user behaviours as they happen in a user-system interaction episode. These behaviours are captured as user-system interaction events and sent to a database for further processing. These behavioural data are great at telling us the "what": when users click on a link, select an element, or move the cursor, the data that is recorded on these events can tell us with great fidelity what behaviours have occurred. What the analytics data cannot tell us, however, is why the users have chosen to perform these actions. As Lazar et al. (2010, 312) note, "[a]s a stand-alone tool, web log analysis is limited by a lack of contextual knowledge about user goals and actions." Though analytics is an extension of traditional server log analysis in the sense that it can capture more fine-grained data, the same limitations hold true also in the analysis of analytics data. We might be able to make educated guesses on the reasons behind certain behaviours or infer these reasons by combining the behavioural data with knowledge on the layout and structure of an application, but in most cases the reasons cannot be deciphered based solely on the analytics data collected in the wild. For example, a long session time that the user stays on a given page might signal that they were engaged in the content of the page, but might also be a result of distractions in the environment, such as answering a phone call while browsing the site. The action of clicking on a menu item might signal that the user perceives the menu item as taking them closer to achieving some goal in the application, but might also be an opportunistic experiment or simply a slip in the use of the mouse. However, as the data is in most cases analysed and reported not on the individual user lever, but on a certain level of aggregation, the interaction events that were produced by slips in the use of input devices, for example, are not likely to skew the data significantly.

The "why" behind user-system interactions events cannot be answered with only behavioural data. For that purpose, we need attitudinal data, which can tell us about the

mental models that the users have of the system, the cognitive processes leading to a certain behaviour, and the precise social context in which they are using an application, for instance. If these insights are needed, the behavioural data needs to be combined with attitudinal data collected with other methods, such as interviews, observation, or laboratory studies (Jansen 2009, 2). This sort of data triangulation has been suggested for usability studies, in which the big picture of application usage could be uncovered with analytics data, while the finer nuances of user-system interaction would best be researched using human observation (Hilbert & Redmiles 2000, 418). Behavioural data can also be used as a starting point and guidance for collecting attitudinal data: the most interesting patterns of behaviour that emerge from the analytics data can be studied further using attitudinal data collection methods to flesh out the meanings and finer details behind these behaviours.

5.2.2. Abstraction problem

The behaviours that can be captured using analytics indicate fairly low-level user-system interaction events such as clicks, taps, and scrolls. The challenge, then, is how to abstract meaningful higher-level concepts and operations out of these low-level events. For instance, if we are interested in how users produce a publishable text document in a given application (a high-level operation), a list of clicks and keystrokes (low-level interaction events) is not a very helpful starting point for the analysis; abstracted data on when the users typed text, pasted it from somewhere else, formatted it, or created a table are likely to provide a more fruitful starting point (Lazar et al. 2010, 334). Inferring these meaningful high-level operations out of a list of low-level user-system interaction events, however, can be challenging especially as important contextual information on how these events were produced may not be available (Hilbert & Redmiles 2000, 395).

An issue that that adds to the problem of abstraction is that sequences of events that are structurally different can be semantically identical: two users may produce a different set of interaction events as they complete the same higher-level operation that is of interest to the researcher, say use the search function of an application. In a traditional usability evaluation using researcher observation, for instance, the capture of these

higher-level operations is fairly straightforward, as the researcher can note and code the operations into the data regardless of which specific interaction events were produced: the abstraction takes place inside the researcher's head. When the same higher-level operation is studied using analytics, the analysis must abstract these different sequences of events as denoting the same higher-level operation. Without this abstraction, even simple questions such as how many times the search function has been used during a time-frame can be difficult to answer. Furthermore, as the set of data collected using analytics can be very large, this abstraction should be automated to as large extent as possible. As Hilbert and Redmiles (2000, 384) note, "because user interface events are typically voluminos [sic] and rich in detail, automated support is generally required to extract information at a level of abstraction. . . ." Luckily, analytics allows more fine-grained data to be collected than a traditional web server log file analysis, and in some cases the low-level interaction events can be abstracted as belonging to the same higher-level operation with the tools provided in modern analytics suites. Completing a purchase on a web store serves as an example of this: the detailed interaction events that a user needs to complete during the checkout process can all be traced and, with the help of analytics software, abstracted as belonging to the same purchase process. This, then, would allow us to see what portion of the users who started the purchase process (by leaving a trace of the first interaction event specified as belong to it) actually completed it (by leaving a trace of the last event).

5.3. Sources of errors and uncertainties in the data

Similar to any other method of gathering data, behavioural data gathered with the help of analytics has some uncertainties and possible sources of errors. The following two sub-sections will introduce two such nuisances: the unique user problem and the unnatural data that stem from application development and testing.

5.3.1. Unique user problem

The word *unique* as it is used in analytics reporting often refers to unique users: the number of unique users of an application is meant to designate the actual number of

individual persons that have used it. The issues related to counting unique users, however, are many and in practice render it impossible to have reliable metrics on anything that has unique users as part of the equation. The fundamental issue is that most analytics solutions actually refer to a *unique device* when using the term *unique user*.

The first dimension of these problems is related to how people access web and mobile applications: the same person can access the same application on several devices or several persons can access the same application on the same device. Nowadays many users own several devices ranging from stationary workstations to mobile devices and use all of these to access a specific application. On the other hand, the same device, such as a household PC in shared use, can be accessed by several persons to interact with the same application. Most analytics solutions rely on cookies or device ID's to identify a unique device, which is then somewhat misleadingly reported as a unique user.

The second dimension is connected to the nature of cookies and relates especially to web applications: as most web analytics vendors rely on cookies to identify unique users, any time a cookie gets lost or deleted between use episodes from the same device, the new use episode gets marked as originating from a new unique user. What is more, the users can disable cookies from being saved into their device entirely, which means that each visit from that device would get marked as coming from a new unique user; the use of the incognito features available in many modern web browsers leads to the same result. The unreliability of the unique user metrics collected from web applications can lead the researcher to reporting 1 million potential opportunities to convert a visitor to a given website into a customer, while the real number could be much less (Belkin 2006). This has led to many companies beginning to track cookie-independent unique session or unique visit metrics because their better reliability (Jansen 2009, 29).

The unique user problem is not as severe in mobile applications as it is in web applications; in mobile applications, a unique user is in most cases identified by the device ID, which is a more persistent method of identification than a cookie. Though the same device can still be accessed by several persons and hence *unique device* would

be a more fitting term for the metric, at least the device ID remains unchanged across use episodes.

A potential solution for the unique user problem in web applications in which users sign in with their credentials is to use the user ID generated by the authentication system of the application to distinguish unique users in the analytics reporting, too. In fact, some new solutions such as Mixpanel's People feature or Google Analytics' User ID feature allow for this. Still, however, this does not permit us to conclude with absolute certainty that all the data points received from the same device ID or user ID were actually produced by the same individual; the same level of certainty on the identity of the user as in a laboratory session where we can at all times verify that the data comes from the same individual cannot be reached in analytics data.

5.3.2. Usage originating from application development and testing

Whenever the development and testing of an application are ongoing at the same time with an analytics research project, it is essential to be able to filter out the interaction data that results from these activities; application testing, in particular, can result in a large number of recorded data as the team of developers are triggering all interface events a number of times to verify that different features work as expected. If all these event data are captured and sent to the application vendor's servers for later access, they can significantly skew the data towards unnatural test use and hence be a serious threat to the internal validity of the data set: the data do not measure what they are supposed to measure, i.e. actual user-system interactions as they take place in the wild. The larger the proportion of this unnatural test use data is of all the recorded interaction data, the larger the threat to internal validity.

There are broadly speaking two different ways of filtering out this data from the reports. First, the code that captures and sends the interaction data to the analytics vendor's servers can be left out or disabled from the development version of the application and included or enabled only in the production version. This approach has the benefit of making absolutely sure that no data is recorded from those application versions that are

distributed in the development team, but requires somewhat more developer resources, as the code for both the development and production versions needs to be specified, written, and tested individually.

The second option is to filter out the interaction data that is caused by application development and testing through different filtering options provided by the analytics vendor's reporting interface. The filtering options provided by different analytics vendors are numerous: IP address of users' network or device, the geographical location based on the IP address, operating system, device make and model, and version number of the application, along with many other user properties, can be used to filter out data that results from application testing.

Which of these approaches to choose depends largely on the situation and is a trade-off between having more certainty in the data originating from natural usage and using developer resources: Disabling the tracking code from the development versions distributed among the team takes some developer resources, but using this strategy the researcher analysing the data can be absolutely sure that no data is captured from these versions; some testing, however, is often also carried out using the production version of the application and data originating from this usage would then be captured. Using the filtering options provided by the analytics vendor's reporting interface takes less resources, but finding suitable filter dimensions can be tricky: if, for instance, the application development and testing takes place in a single office location, filtering out the IP address of its network may work in some cases, but if the application is mobile native, any usage that takes place while the device is connected to a mobile network would not be filtered out using this approach.

5.4. Analytics and ethics

Analytics as a way of collecting controlled accretion trace data in digital settings is, from the users' perspective, an unobtrusive method. The benefits of using an unobtrusive method were discussed above: the researcher can limit or fully escape the effects of some evaluation artefacts, such as the observer effect, that may be present

when some other methods popular in HCI research are used. In the sociological research tradition, the notion of unobtrusiveness has been valued to an especially high extent: it has been stated that the researcher's "intervention should not impair the nonreactivity of the erosion and trace measures by permitting the subjects to become aware of his testing" (Webb et al., 2000, 43). However, tracking users of a software application without them being permitted to become aware of this might give, and indeed have given, rise to some ethical considerations.

Aggregate web analytics, which enable services such as Google Analytics' demographics feature (Google 2014d), have been a target of criticism from this ethical point of view: aggregate web analytics track user behaviour across the internet and based on the recorded behaviour infer the demographics and interests of a given user. This demographic data can then be offered to web site publishers as the demographics of their user base. The criticism against aggregate web analytics and other forms of behaviour tracking have led to restrictive legislative actions and to institutional policies such as the Do-Not-Track by the W3C (Akkus et al. 2012, 687). In an attempt to establish some integrity into the field, a community of analytics practitioners have published a Code of Ethics document crafted to address these issues (Digital Analytics Association). Several browser vendors have also been tackling these issues by building features that allow their users to opt out of analytics tracking (W3C 2011), while some researchers and practitioners have been building their own analytics solutions on more solid ethical grounds: Akkus et al. (2012) and Chen et al. (2013) describe architectures for aggregate web analytics solutions which do not track individual users at all.

A common stance in the literature and practitioner side towards the ethics of using analytics seems to rely on the notion of non-identifiability of the collected data: When building their own analytics tool, Leiva and Vivó (2013, 26) decided to collect data on keyboard events without their associated character codes, as opposed to tracking raw keystroke data, which could potentially be used to identify individual users and to uncover sensitive information or business secrets. Harris (2005) describes a data collection effort by stating that data was collected on anything "we think might be interesting and useful as long as it doesn't compromise a user's privacy." Many analytics solutions also take away some of the ethical decisions from the researcher by

reporting the collected data in abstract and aggregate form; the analyst cannot see, for instance, the users' exact IP numbers, which could be used to identify specific devices or to specify which actions were taken by which users. Furthermore, in countries such as Finland the law requires the holder of personally identifiable data to publish a personal information registry notice (Finnish personal information law 1999).

Another ethical consideration is the principle of transparency: Lazar et al. (2010, 312) recommend that applications which record interaction data should have publicly available privacy policy statements stating which data are collected and how they are used, while the Web Analyst's Code of Ethics encourages the full disclosure of data collection practises in a clear and understandable manner (Digital Analytics Association).

The research carried out for this thesis adheres to the principle of non-identifiability: none of the data that was collected could be identified as originating from a specific user or device. The privacy policy statements that were in place in the studied applications were not, however, changed because of the data collected for this thesis.

6. Case studies: research procedures, data analyses, and implications on improvement

This chapter presents the three case studies that were used to study the possibilities that analytics brings to improving web and mobile applications. The first two case studies concern mobile-native applications and the third one a web application. Furthermore, the first case study involves an experiment with a between-subjects experimental design. For confidentiality reasons, the last two case studies were purged of all information that could be used to identify the applications in question, including their real names and any screenshots of their interfaces, for instance. All three case studies are presented in the same format: First, a rough-level presentation of the application is given. Second, the research question and the procedure that was used to study it are presented. Third, results of the study along with the data that were used to arrive at the results are discussed. Finally, implications of the results on improving the applications are discussed.

6.1. Improving conversions with user interface A/B testing

Different applications and even specific sections inside of those applications can have a variety of different goals: some aim for engagement and hence long use session times, others for enabling their users to quickly find a specific piece of information, while at the heart of more transactional applications are conversions, such as sign-ups and purchases.

As a specific case of conversions are in-app purchases inside a mobile application. The term *in-app* refers to *inside application*: once a user has installed an application into their mobile device, they can be offered additional features that do not come as a part of the core version of the application. The business model of several mobile games is also based on in-app purchases: the games are free to download and play, but in order to

acquire certain virtual goods or to advance faster in the game, the players are offered these features as in-app purchases.

Maastokartat is a Windows Phone mobile application that provides its users with topographic maps of Finland into their smartphones. During the studied time period it was one of the most downloaded applications for the Windows Phone operating system in Finland, though exact statistics are not published. Though Maastokartat mobile application is more functional than transactional in its nature, it nevertheless offers two additional features as in-app purchases for the price of 1.99€ each. Firstly, the users can purchase a feature that allows them to record the routes that they travel and later view these routes imposed on the map. Secondly, the users can purchase a feature that allows them to save map areas into their device for offline use, which can be convenient in locations where data connections are poor.

As conversions, such as in-app purchases, are important for the business model of a variety of companies, it is in the interest of these companies to improve the conversion rates: the larger the proportion of their users who buy a given good, service, or, as in the present case, a feature, the better the companies will perform businesswise. This reason acted as the motivation for the present research case, too: optimizing the navigation flows leading to in-app purchases for higher conversion rates were of interest to the developer of the application. Hence the notion of improvement will in this case refer to establishing higher conversion rates. As the interest in the industry to improve conversions is, for financial reasons, widespread, the present research case exemplifies how analytics can be used for this purpose.

6.1.1. Research procedure

In Maastokartat mobile application the navigation flow leading to the in-app purchase of the two additional features consists of few steps that the user must complete in order to acquire the feature:

1. The user opens the description of the feature by navigating to it from the application bar or application menu. The description is opened when the user taps on the menu item that would access the feature if it was already purchased. In other words, if the user has not purchased the offline maps feature yet, tapping the menu items that are used to access the part of the application where maps can be saved for offline use opens the description of the feature.

2. On the description screen the user taps the *buy from store* button (*osta kaupasta* in Finnish).

3. After tapping the *buy from store* button, the user is taken to Windows Phone Store where they confirm the purchase. In order to do this, the user must have valid payment methods set up in their Windows Phone Store account. The developer of the application has no control over the appearance of the confirmation dialogue in the store.

4. The user is taken back to the application and directly onto the view where they can start using the feature that they just purchased.

The navigation flows leading to the in-app purchases of the record routes feature and the offline maps feature are visualized in Figure 10 and Figure 11, respectively. Both flow visualizations are from Maastokartat version 2.4.1.0, which served as the first version for which analytics tracking was set up.

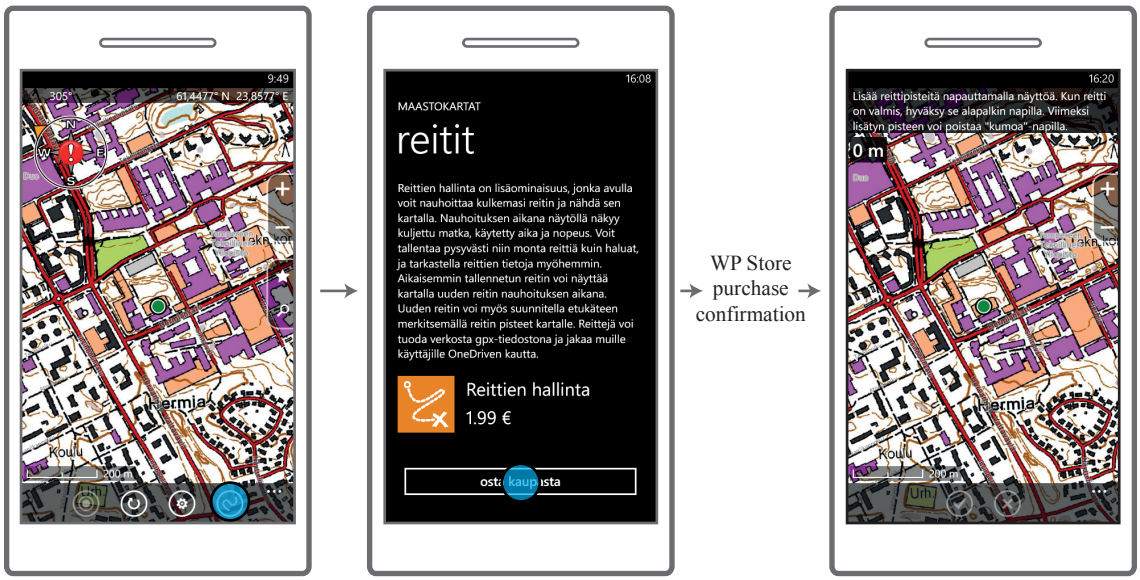

Figure 10. Flow leading to the in-app purchase of the record routes feature in Maastokartat version 2.4.1.0. The blue dots designate the taps that the user makes in order to purchase the feature.

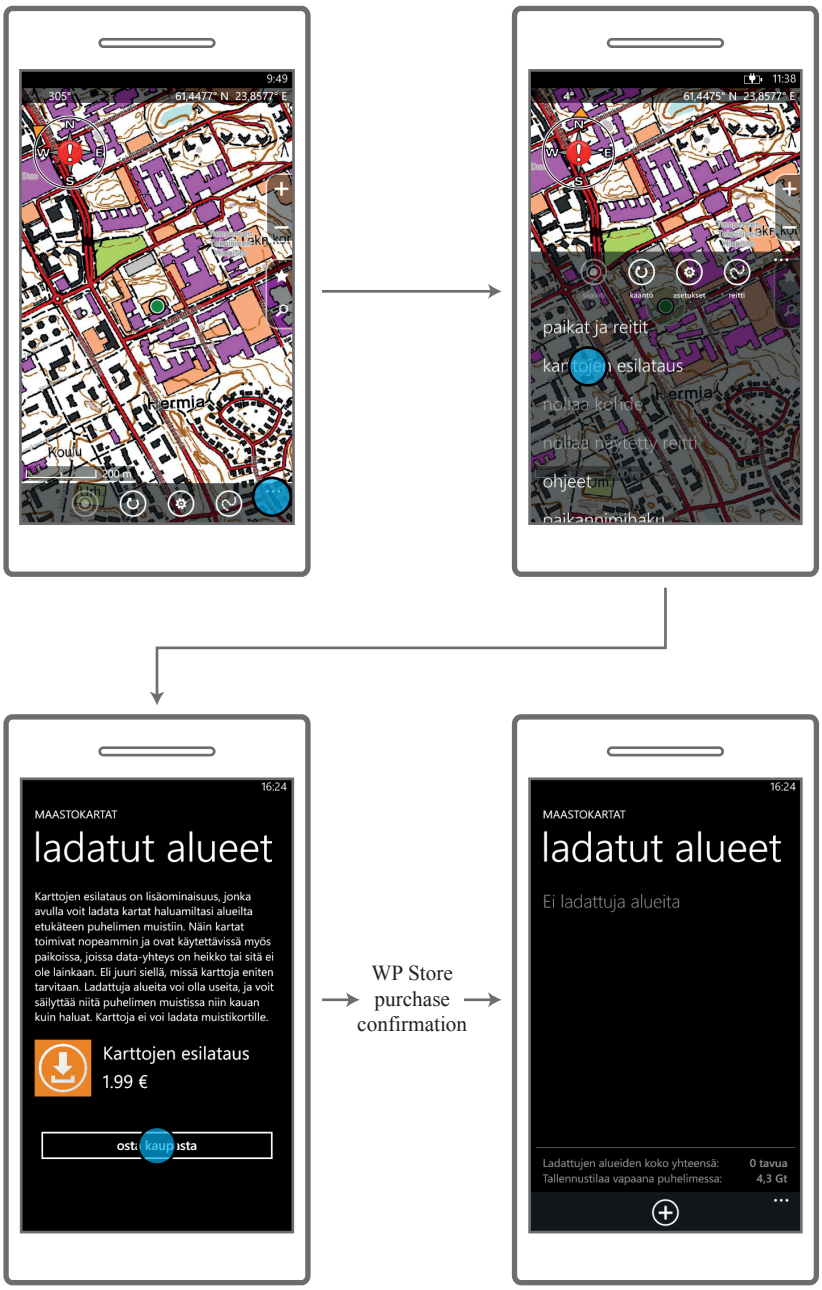

Figure 11. Flow leading to the in-app purchase of the offline maps feature in Maastokartat version 2.4.1.0. The blue dots designate the taps that the user makes in order to purchase the feature.

A subtle yet important difference between these two flows is the way in which the description view is navigated to: in the record routes flow in Figure 10, the user navigates to the description view by tapping an icon in the application bar; the description is accessed with just one tap. In the offline maps flow in Figure 11, the description is navigated to by first opening the application menu by tapping an icon in

the application bar and then tapping the correct item in the application menu; the description is accessed with two taps.

To study the navigation flows leading to these in-app purchases, the in-view tracking approach described in Section 4.2 was employed: the source code of the application was instrumented with additional code to track the user-system interaction events related to these flows. For both of these features this meant tracking the following three events:

1. The user opens the feature description view (event fired when the view is opened).
2. The user taps the *buy from store* button (*osta kaupasta*, event fired when the button is tapped).
3. The user confirms the purchase in Windows Phone Store (event fired when the user returns from the store with a purchased feature).

The data from these events, along with all other application usage data, was sent to a third party analytics service called Flurry. The data post-processing tools provided by Flurry allow for some convenient ways to abstract higher-level patterns out of these low-level user-system interaction events. Whenever the higher-level pattern can be phrased as a goal that consist of several steps that the user may complete, such as a registration process or, as in the current case, a purchase process, a funnel report is especially useful (Beasley 2013, 60). A well-constructed funnel report shows the number of users who start the process and the number of those users who abandon the process on each of its steps.

The funnel reports in Flurry are user-based: though a single user can trigger a specific interface event belonging to the funnel several times, for example by looping between the main map view and a feature description view and hence triggering the *feature description opened* event each time, in the funnel report for a given time period each unique user is counted just once. Even if a user enters the first step of the funnel several times (even in different use sessions) before continuing to subsequent steps, the number of events that the user triggered on the first step is coalesced into one in the funnel report. It should be noted, however, that different analytics solution vendors offer

funnel reports in which these numbers are counted differently, for example by showing each entrance to the funnel separately.

Furthermore, the funnel report in Flurry was constructed so that a time frame of seven days was allowed to take place between each step. This means that a user does not need to complete the funnel in one interaction episode for it to appear in the report, but rather they can view the description of the feature in one session, ponder over purchasing it for a while, and then come back to the application to actually purchase the feature. When constructed this way, the funnel report was useful in showing what percentage of users who entered the funnel ultimately completed it and on which steps they were dropping out of the funnel.

As was noted above, the first version of Maastokartat Windows Phone application that was instrumented with analytics tracking was version 2.4.1.0. The initial funnel report data from this version showed that not only was the record routes feature purchased more often than the offline maps feature, but also that a significantly larger number of users were entering the funnel by triggering the event specified as its first step. In fact, in both funnels roughly the same percentage of users who entered the funnel were completing it; it was just the case that the record routes funnel attracted significantly more users to enter its first step. Intuitively, this difference could be related to the way in which the first step of the funnel, i.e. the feature description view was navigated to. As was shown in Figure 10 and Figure 11 above, the record routes feature description could be entered from the main map view with one button tap, whereas the offline maps feature description was buried two taps deep into the menu structure of the application. On the other hand, the difference could, among many other variables, quite simply result from the fact that one feature was more attractive to a larger number of users than the other one, resulting in more interest towards that feature.

An experiment was set up to test the intuition that the interface of the application had an effect on the number of conversions. More specifically, the research question that was derived from the initial set of data was the following:

- Does the layout of the application and the navigation flow leading to an in-app purchase cause a change in the number of purchases?

In terms of experimental design, the layout and the navigation flow of the application form the independent variable to be manipulated, whereas the number of purchases is the dependent variable to be measured. As the number of purchases of the offline maps feature was trailing behind the record routes feature, the offline maps feature was identified as a more suitable target for the experiment.

To suggest an answer to this research question, the users of the application were subjected to two different navigation flows leading to the purchase of the offline maps feature. The experiment was carried out using a between-subjects design: the users who upgraded to version 2.5.0.0 of the application were programmatically and randomly assigned into one of the conditions during their first launch of this version of the application. In fact, the users were taking part in the experiment without them even knowing about it. In the industry, this type of experimentation is often referred to as *A/B testing* (in case of two alternative designs) or *multivariate testing* (higher number of variables and their interactions). As defined by Beasley (2013, 201), A/B testing involves taking two designs, determining a single metric for measuring success, and subjecting users to both designs until there is a statistically significant difference in the measured metric between the designs. In this experiment, the original navigation flow that was presented in Figure 11 was treated as the A, or control condition, whereas a new navigation flow was designed and built for the B condition:

Figure 12. Redesigned flow leading to the in-app purchase of the offline maps feature in Maastokartat version 2.5.0.0B. The blue dots designate the taps that the user makes in order to purchase the feature.

Compared to the A condition, in the B condition the users could navigate straight into the feature description screen by tapping an icon in the main map view; in the A condition the first step of the funnel is opened with two taps, whereas in the B condition this takes just one tap. In this sense, the navigation flow for the offline maps feature in the B condition resembles the navigation flow for the record routes feature, which, in the earlier version of the application, was being purchased by a larger number of users.

6.1.2. Results

The numbers of users who entered each of the funnel steps leading to the purchase of the offline maps feature in Maastokartat version 2.5.0.0 during March 2014 are cited in Table 2. Figure 13 shows the same information coupled with interface screenshots. Numbers are given separately for both A and B variations, together with conversion percentages for each step and for the entire funnel. Furthermore, to be able to calculate the conversion percentages for the entire funnel, the numbers of those users who were exposed to each of the conditions should be quoted in the table. In practice this would mean having, for both variations, a number of those users who had not purchased the

feature before the experiment began and had at least one use session during the sampled time period; users who have already purchased the feature cannot purchase it again. However, this exact number could not be attained with the analytics instrumentation that was in place: no trace was left of whether a given user had already bought the feature before this period, and hence should not be included in the experiment because they would not be able to enter the funnel. As this number was not available for the analysis, the number of active users of both A and B variations in March 2014 are given in the table instead (*active user* defined as a user who had at least one session during the time frame); during the sampled time period the application had 4402 active unique users in total. This number is not perfect as it also includes the users who had purchased the feature before the experiment, but was the best statistic that was available. However, the inaccuracy caused by this flaw will not introduce any bias between the A and B variations as all users were assigned to these groups in random, but will just simply cause the conversion rates for the first step and for the entire funnel to be lower than they are in reality. In the ensuing statistical analysis the flaw can only bias the results towards the direction of statistical insignificance and hence any statistically significant differences between the variants are in reality even more significant than the analysis suggests.

Step	Version A		Version B	
	Users	Conversion % from previous step	Users	Conversion % from previous step
(Active users)	2171		2231	
1. Open the *offline maps* feature description	197	9.1%	345	15.5%
2. Tap the *buy from store* button (*osta kaupasta*)	55	27.9%	87	25.2%
3. Confirm the purchase in Windows Phone Store	32	58.2%	49	56.3%
Total conversion %		1.5%		2.2%

Table 2. Numbers of users who entered each of the steps of the purchase funnel for the offline maps feature in Maastokartat version 2.5.0.0 for both A and B variations in March 2014.

Figure 13. Interface screenshots of the purchase funnel steps.

In the A condition, 32 users, or 1.5%, out of the total of 2171 active users went through all of the steps, purchasing the offline maps feature on the last one; in the B condition the corresponding numbers were 49 users out of the total of 2231 active users, equalling the conversion rate of 2.2%. As the conversion data is nominally scaled, the significance of the difference between the total conversion rates in conditions A and B was calculated using Pearson's chi-square test. For the total conversion rate, the test yields a value of $\chi^2 = 3.18$, which corresponds to the p-value of 0.07 (two-tailed, df = 1). The p-value of 0.07 means that there is a 7% possibility of the difference being simply a result of chance, i.e. caused by sampling variation, while our confidence in the difference being a result of the difference between conditions A and B is 93%. For the purposes of optimizing the navigation flow leading to a purchase, this level of confidence is more than enough to make a decision on which flow to select.

If we look at the conversion rate only on the first step, however, in the A condition 197 users, or 9.1%, of the total of 2171 active users opened the feature description screen; the corresponding numbers for the B condition were 345, or 15.5%, out of the total of 2231 active users. For these numbers, Pearson's chi-square value is $\chi^2 = 41.61$, corresponding to the p-value of 0.00 (two-tailed, df = 1). This difference is statistically close to certain, meaning that we can be quite certain that in the B condition the users open the feature description screen for the offline maps feature more often.

6.1.3. Discussion

Deciding which version of an interface performs better in terms of user engagement, task completion times, or purchase conversions, for instance, can be tricky with qualitative research methods. What this A/B testing case study showed was that by actually building two versions of the interface and pitting them against each other, while measuring their performance in a given aspect with the help of analytics, these decisions can be made into more informed ones. As the whole idea to perform an A/B test was based on the initial set of data, analytics data was also discovered to be a fruitful source of hypotheses to perform experiments on. In the present research case the difference in purchase conversion rates between two interface versions was discovered with a high

degree of statistical significance. Though this case study was about in-app purchase conversions, the same procedure could be used to study which type of a design would lead to longer use times or to a larger number of returning users, just to mention a few of an array of possible dependent variables that can be studied with A/B testing.

As this research case was, in Beasley's (2013, 14) terminology, a fairly structured one, the results were also highly actionable and can easily be used to improve the application. As the difference in conversion rates between the A and B versions was discovered with high certainty, the developer of the application can feel confident in the decision to choose the B variant for all users of the next application version. For further improvement, however, A/B testing should be adapted as a continuous process: even better performing variations would surely be discovered with new rounds of testing.

6.2. Feature use counts

Modern web and mobile applications are laden with features. Whereas some years ago applications related to complex tasks such as accounting, word processing, or spreadsheet calculations used to exist only as native desktop applications, nowadays these tasks and many others can be accomplished using a web application or even a native mobile application running on a portable device. When used in the study of these applications, traditional usability and user research methods can provide qualitative data on how users are interacting with them: different task flows, interface concepts, and terminology can be evaluated using these methods. Gathering quantitative data on the use of these applications, especially on some level of statistical confidence, however, requires a different approach.

Which features are the most used ones in this application? How many times was this button pressed? How many users performed searches in this application? Equipped with answers to these questions, the team designing and developing the application would potentially be in a better position to make design decisions on which views to make more easily accessible, which features visually more prominent, which features to drop

from future releases, and how to structure the entire architecture of the application; the data could be used to improve the application from the users' point of view.

As an example of a mobile application that is, indeed, laden with features and hence difficult to gather quantitative data on using traditional user research methods, this section focuses on a mobile application of an enterprise document management system, which will be referred to with the name *Alpha*. This document management suite differs from many others in the sense that it is based on document metadata as opposed to traditional folder structures which users could navigate to in order to find specific documents. In its entirety the system comprises of several components and versions for different operating systems, but as the native application for Apple's mobile operating system iOS was under development at the time of writing this thesis, it was of special interest to the company to gather data on how users were interacting with some of the earliest released versions.

6.2.1. Research procedure

If the research project on Maastokartat Windows Phone application described above was more from the structured end of the analytics research spectrum described by Beasley (2013, 14), the current project could be seen as falling more into the unstructured end of the spectrum: the company was more interested in open-ended exploration of how users were interacting with the application, which views they were engaging with, and which features were most used. Because of this open-endedness, no clear hypothesis for the research project could be pinpointed and the research question also remained more open-ended:

- Which views and features the users of the application use the most?

To provide answers to this broad research question, the in-view tracking approach described in Section 4.2 was used extensively: several interface events that corresponded to using different features of the application were instrumented with appropriate analytics tracking. Furthermore, as mobile applications do not have pages

that could easily be tracked in the same sense as websites do, different sections of the application, or, in analytics jargon, screens, were instrumented to gather data on whenever users viewed these screens. In this sense, the current research effort was similar to what was described by Harris (2005): data was collected on anything that could potentially provide interesting and useful insights.

As Alpha document management system is based entirely on document metadata as opposed to traditional folder structures, the search function provided by its applications is of vital importance for accessibility, usability, and user experience: as users do not have folder structures to navigate to documents profiles, the search becomes one of the main tools to locate documents relevant to the users' tasks at hand. Hence the user-system interactions related to the search function were studied in more detail: the search event was instrumented so that it recorded not only the occurrence of the search event, but as a parameter also the number of search results that the query returned. The search interaction consisted of typing in a search string, optionally restricting the search to a specific document profile type (such as persons, documents, or projects), and optionally restricting the search to metadata fields only or file contents only.

Conceptually the research on the interactions related to search behaviour within information systems builds upon a specific subset of the analytics research tradition: in the literature this subset is often termed *search log analysis*, or SLA, and it can be used to study the processes related to users' information searching behaviour (Jansen 2006, 407). An understanding of how users search for information can then be used to design for a better search experience (Russel-Rose & Tate 2013, 2). As the name suggests, search log analysis has been carried out mainly by studying the log files generated on web servers; for the research purposes of this section, and more generally of this entire thesis, however, the data was collected using the page tagging analytics approach that was described in more detail in Chapter 4.

All these usage data were captured and sent to further processing using a third party analytics service Google Analytics. The basic version of Google Analytics is free to use and it is one the most popular analytics services today.

In the current research case, the development and testing of the application were also ongoing during the time of carrying out the research. As was discussed in Section 5.3.2, in cases such as these it is important for the internal validity of the data set to filter out data that results from the unnatural test use. For Alpha iOS mobile application, the option to use the filtering capabilities of Google Analytics' reporting interface was deemed appropriate: as the development and testing of the application took place only in two office locations, it was relatively easy to filter out data that originated from the networks of those two locations based on the networks' IP addresses. If the device on which testing was performed, however, was connected to some other network, the test usage would have been recorded, too. With this proviso in mind, the present author is confident that the vast majority of the collected data originated from actual users using the application in the wild.

The data analysed in this section was further filtered down to just one application version to ensure internal consistency of the data set. The sampled time period was set to cover a time frame of 24 days. During this time period the studied application version had 464 active unique users who used the application in altogether 1383 sessions.

6.2.2. Results

The studied application version was divided into five main screens which the users could access through a tab bar. Data on user-interactions with these views are listed in Table 3 below:

Screen name	Total number of views	Number of sessions in which screen was viewed	Number of users who viewed screen	Average time on screen per view
Home	3651	1281	462	47 s
Assigned to me	2603	558	296	35 s
Search	2370	701	320	53 s
Recently accessed	1007	430	255	38 s
Favourites	493	311	215	18 s
Total	10124			

Table 3. List of recorded screen views and numbers of their total occurrence, sessions in which they occurred, users who viewed the screen at least once, and average time spent on screen per one view in Alpha iOS application. The list is ordered by the total number of views.

The application's *home* screen was the most viewed screen with over one third of total screen views. This is partly explained by the fact that the application opened to the *home* screen whenever it was started after having been inactive for a certain period of time; these screen views were recorded, too. The *assigned to me* and *search* views also collected large portions of total screen views, while the *recently accessed* and *favourites* views were accessed less often. Whereas all but two users viewed the home screen at least once during the studied time period, the four other screens were viewed by smaller portions of all active users. On average, most time per screen view, 53 seconds, was spent on the *search* screen.

Table 4 below provides a list of all the different user-system interaction events on which data were recorded along with the counts of the times that these events occurred:

Interaction event	Total number of events	Number of sessions in which event occurred	Number of users who triggered the event
Open document	4925	988	384
Open file	1574	590	268
Search	1180	1159	428
Save property edits	811	105	66
Open properties edit	573	192	128
Mark assignment complete	509	48	32
Switch vault	334	298	235
Cancel property edits	179	114	86
Change workflow	166	59	40
Comment document	92	38	21
Open add properties	91	49	40
Add property	22	15	13
Add photo to new object	20	15	12
Mark assignment incomplete	16	14	11
Add photo to existing object	15	15	13
Delete property	11	7	7
Total	10518		

Table 4. List of recorded interface events and numbers of their total occurrence, sessions in which they occurred, and users who triggered the events in Alpha iOS application. The list is ordered by the total number of events.

The list reveals that the core features of a document management system, opening document profiles and the files attached to those profiles, top the list of the most used features. The *search* function, along with the different functions related to editing the

properties of a document profile (i.e. its metadata fields) are also popular features, while users rarely add photos to document profiles or delete any of their properties.

The list also reveals an error in the instrumentation of the application: opening the properties edit of a document profile should logically always be followed by either saving the properties edit or cancelling the properties edit. Hence the total number of events for the *open properties edit* feature should equal the number of events for the *save property edits* plus *cancel property edits*. As is revealed by the numbers in Table 4, this is not the case. The developers responsible for the instrumentation could not easily locate the exact origin of this error and hence the issue was not pursued any further. There is not, however, any reason to believe that there are any other errors in the numbers quoted above and the present author is confident that they give a good overall picture of the general usage of the application.

With this list of how frequently different features were used, the team of designers and developers behind the application are obviously in a better position to make design decisions for new versions. For instance, the second most frequently used feature, opening a file attached to a document profile, was not visually that prominent on the document profile screen and required the users to make two button taps to perform. Making the files more easily accessible from the document profile screen is, therefore, one target of a redesign for new versions of the application. On the other hand, as adding photos directly from the device to a document profile, either to an existing one or a new one, was performed only 35 times in altogether 2.2 % of the sessions during the studied time period, it perhaps need not be one of the most prominent features in the interface.

During the sampled time period the users of the application performed 1180 searches. The frequency distribution of the numbers of results that these searches returned is depicted in the histogram in Figure 14:

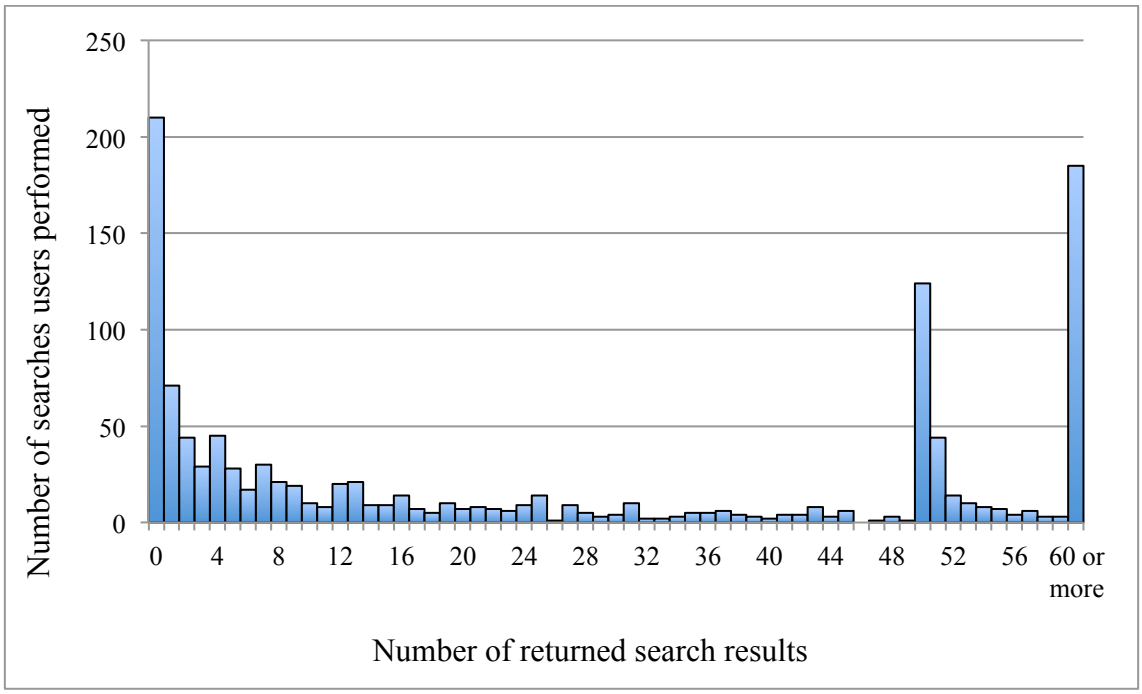

Figure 14. Frequency distribution of the numbers of search results that user-performed searches returned in Alpha iOS mobile application.

By far the most common number of results that the searches returned was zero: of all 1180 searches that were recorded, 210 resulted in no matches. As the frequency distribution starts levelling out towards higher numbers of search results, there were altogether 424 searches that returned somewhere in the range of one to 20 results, which can be seen as a helpful range of search results. After that, the frequency distribution peaks again at 50 search results. At first the peak at 50 results appeared to be an anomaly in the data, but on closer inspection it turned out to be simply a result of an application feature: on the server side the search functionality was configured so that it returned a maximum of 50 results per document profile type. Hence a search interaction that was restricted to a specific document profile type could return a maximum of 50 search results, resulting in the peak in Figure 14. After 50 search results, the distribution levels out again, with relatively few searches returning more than 50 results and 655 being the highest number of search results that were returned. The long tail of searches that returned 60 or more results was coalesced into the rightmost column in the histogram for presentation reasons.

For a user who is attempting to find a specific document or a set of documents through the search function of an application, zero number of search results is obviously not a good outcome, especially as in the studied application version this meant that the user simply came across a screen with a fairly unnoticeable *No results* text on it. Though some search interactions are bound to result in no matches, reducing the number of those searches, or, in case of no results, offering the user with some additional means of finding the information they are after, are potential targets of improving the application. Many web search services, for instance, achieve the former by offering the user with some related search strings to try or, in the case misspellings, suggestions worded in a format such as "Did you mean [search string spelled right] instead?"

Above I deemed the range of one to 20 search results as being a helpful number of results. Deeming this range as helpful is, of course, only speculative: if a user is attempting to find all documents with pictures attached to them, for instance, a higher number of results than 20 is likely to better meet the user's intentions. The qualitative statement *helpful* was hence used more from the interface design perspective: the application version that was studied displayed the search results simply in list format. Long lists with tens or even hundreds of search results stacked over each other are potentially confusing to users, especially as they are viewed using mobile devices with relatively small screen sizes. When put into a single list, the highest number of search results that was recorded, 655 results, will form a list so long that it compromises the usability of the entire results listing and search function.

In the course analysing the search data, it was discovered that to be able to learn more about users' search behaviour, additional instrumentation would have been helpful: had the search interaction been instrumented in a way that would have captured not only the number of search results that the query returned, but also the search string and other search parameters that returned a given number of search results, more could have been deduced on why certain searches returned a useful number of results while others returned none or an overwhelming number of results. There are, of course, ethical considerations related to this: while tracking the search interaction along with the number of results is entirely unidentifiable and cannot to be linked to a specific person or organization, information on what search strings the users typed could in many cases

be used to deduce who or in which organization the search was performed, or be used to uncover business secrets.

6.2.3. Discussion

By using the in-page or, in the case of a mobile application, in-view interaction event tracking to perform feature use counts, it was discovered that plenty could be learned on how users interact with an application. By listing which features are the most and less used ones, decisions, such as how to structure the application's information architecture and which features to prioritize in its interface design, could be backed up by actual quantitative interaction data.

Furthermore, by gathering additional data on how many results each search interaction resulted in, guidelines for how to redesign the search experience could be gathered. As the most common number of results that the search returned was zero, it is evident that this scenario should be designed for: in the studied application version the user simply saw a results screen with textual information that no results were found. No additional tips on related search strings or search techniques were offered, which, based on the frequency of this event, could improve the search experience for a large number of users. As a large number of searches also returned tens and some even hundreds of search results, this case should also be designed for: the practice of simply listing all these results on top of each other on the screen of a relatively small portable device may result in difficulties when the user is viewing the long results listing. Practises such as paginating the search results are often seen in web search engines, for example.

As the current research project was more from the open-ended and unstructured end of the spectrum, similar issues to what were reported by Harris (2005) were encountered: as the data were gathered with no strict research question in mind and on the majority of actions that users could perform, much of these data proved to be not very actionable in improving the application. However, the numerical information on feature usage can help in gaining an overall picture of how large masses of users are using the application in the wild.

6.3. Form filling

Receiving input from the user is one of the design challenges that practically any interactive software application must solve in one way or another. Though with technological development many new methods for receiving input, such as speech and touch-free gestures, are nowadays available, the majority of web and mobile applications still rely at least partly on textual input either through a hardware keyboard or an on-screen keyboard.

A typical example of textual input is filling in some type of a form. Though forms can differ greatly in length and complexity, even a single field asking, for instance, users' name or password can be seen as constituting a form. Good form design is considered to be important because, first, users dislike interacting with forms and, second, they are ubiquitous in all kinds of applications ranging from online commerce to social media (Wroblewski 2008, 4). Moreover, filling in forms often play an important role in interactions which are critical for the success of a given business: sign-up and checkout processes usually require the user to fill in some form fields.

Analytics provides an efficient and unobtrusive way of studying how users are filling in forms and what problems they may experience when doing so. By uncovering potential problems, these can be fixed and the form improved. Analytics is, again, a method that can quantify form interactions, whereas traditional HCI methods are restricted in scale and scope. To exemplify how analytics could be used for this purpose, the current section describes a research case that studied a web application with a sign-up form. The web application, which will be referred to as application *Beta*, acted as a reporting and management interface for a commercial service, which the users needed to sign up and pay a monthly fee for.

6.3.1. Research procedure

The current research case had relatively clear goals and was hence a somewhat structured effort. The parties involved in developing the service had obvious interests in learning how successful the gate into the service, the sign-up form, was when users

were interacting with it. If users were experiencing problems with the form, the nuisance caused by these problems could potentially make them leave the form and interrupt the sign-up process, costing the company in lost revenues. With information on how many of the users who started interacting with the form actually completed it and on which form fields, if any, they were having difficulty with, the sign-up form could potentially be redesigned or streamlined to improve the users' experience and the sign-up rate to the service. From this starting point, the following research question was arrived at:

- Which form fields do the users of the application have trouble with?

The studied sign-up form consisted of five sections, which were divided into separate tabs in the interface: *availability*, *type of contract*, *personal details*, *payment details*, and *summary*. The exact content of the *personal details* tab varied based on whether the user had selected a personal plan or a business plan as the contract type on the *availability* tab. As moving between the tabs did not result in new page loads, the foundational web analytics metrics that measure page-to-page movement could not be used study users' progress through the tabs. Hence the five tabs, all sign-up form fields, and the form submit action on the web page were instrumented with additional in-page tracking to record the following user-system interactions:

1. The user opens the sign-up form.
2. The user navigates from one tab to another. The name of the tab that was navigated to was also recorded.
3. The user submits the form, but one or several of the form fields on any of the five tabs are either empty or contain content that was validated to inadmissible (invalid email address or social security number, for example). The IDs of the invalid form fields were also recorded.
4. The user successfully submits the form and as a result is automatically logged into the application.

There were a few reasons as to why a given field might be invalidated on form submission: a required field was left empty, an email address or a social security number already registered in the service was being used again, or a field contained a

string that did not match a specific format. Examples of the latter are the email address and the social security number, which both have a standard format. If a user's input did not match this format, the field was invalidated on form submission.

All these interaction data were recorded using Google Analytics and sent to its servers for storing, processing, and later access through its web interface. A two-month period was chosen to be studied. During the sampled time period the application had 2989 unique users.

As in the previous research case, some threats were caused to the internal validity of the collected data by application development. Luckily, the testing related to the development work took place mostly on a development server with a different URL from that of the production version; all data originating from users who had had interactions recorded on the development server (i.e. developers) could be excluded from the studied data set using the URL filtering options provided by Google Analytics' reporting interface.

Another threat to the validity of the data resulted from the fact that the administrators of the service were using the same application in an admin user role to view, for instance, lists of users who had signed up to the service and to access some reports. All administrator level actions, however, took place in pages that had the word *admin* as a part of the URL string; all data originating from users who had had interactions recorded on pages with *admin* in the URL string could, again, be excluded from the studied data set. Without these filters, all interactions on the sign-up form for testing purposes from either the developers or admins of the application would have been recorded in the analysed data set and skewed the data towards unnatural use. However, these filters are not, again, perfect; cookies may be lost and browsers used in incognito mode, resulting in users who should be filtered out not being filtered out. Despite these uncertainties, the present author is confident that the filters used for this research case provide a good level of internal validity.

6.3.2. Results

During the studied time period the sign-up form was submitted 172 times. Out of these submissions, 85 were successful and 87 unsuccessful. Table 5 below details the reasons for the unsuccessful submissions as clusters of form fields that were invalid for each such submission, while Table 6 specifies the numbers that each form field was one of the inadmissible fields that caused the submission to be invalid. The former is useful in seeing if certain groups of form fields, such as all fields on a given tab, often cause an invalid submission, while the latter makes it easier to see which individual form fields caused the most problems.

Clusters of form fields that caused an unsuccessful submission	Total
social security number	22
first name, last name, phone number, email	10
plate number, first name, last name, phone number, email, social security number, address, postal code, city, billing email, acceptance of the terms of service	7
plate number	6
acceptance of the terms of service	6
email	4
business ID	3
first name, last name, phone number, email, social security number	3
plate number, email, social security number	3
plate number, first name, last name, phone number, email, social security number, address, postal code, city	3
discount code	2
email, social security number	2
plate number, first name, last name, phone number, email, social security number, address, postal code, city, billing email	2
email, name, business ID, address, postal code, city	1
email, social security number, postal code	1
email, social security number, acceptance of the terms of service	1
e-invoice address, e-invoice intermediator	1
first name, last name, phone number, email, address	1
first name, last name, phone number, email, billing email, acceptance of the terms of service	1
first name, last name, phone number, email, e-invoice intermediator	1
first name, last name, phone number, email, social security number, address, postal code, city	1
first name, last name, phone number, email, social security number, address, postal code, city, discount code, acceptance of the terms of service	1
first name, last name, phone number, email, acceptance of the terms of service	1
plate number, email	1
plate number, first name, last name, phone number, email, e-invoice address, e-invoice intermediator	1
plate number, first name, last name, phone number, email, name, business ID, address, postal code, city, billing email, acceptance of the terms of service	1
plate number, acceptance of the terms of service	1
Total	**87**

Table 5. Form fields that caused unsuccesful submissions in application Beta, clustered.

Even when viewed as clusters, *social security number* by itself causes the largest number of invalid form submissions. The second most frequent reason for the invalid submission was the cluster of *first name*, *last name*, *phone number*, and *email*, which, curiously, were the four upmost fields on the *personal details* tab. It appears that for some reason, this group of form fields was sometimes overlooked. The third most frequent reason for the invalid submission was the cluster of all required fields on the sign-up form; it seems that some users were simply attempting to submit the form without filling in any of the fields on it; this implies that they were not genuinely attempting to sign up to the service, but rather just clicking around to explore what would happen. The *plate number* and *acceptance of the terms of service* fields also caused some invalid submissions by themselves. The *plate number* field was in a rather unnoticeable place in the *type of contract* tab and the *acceptance of the terms of service* was actually a small checkbox on the *summary* tab; these layout-related problems were probably the reasons as to why these fields were sometimes invalid, probably as a result of being overlooked. After the top clusters of form fields that caused an invalid submission, the numbers level out to only a few or just one instance of a given cluster.

Individual form fields that were part of an unsuccessful submission	Total
email	46
social security number	46
first name	33
last name	33
phone number	33
plate number	25
acceptance of the terms of service	19
address	17
postal code	17
city	16
billing email	11
business ID	5
e-invoice intermediator	3
discount code	3
e-invoice address	2

Table 6. Form fields that caused unsuccesful submissions in application Beta, individual.

When the form field clusters are broken down to individual fields, the *email address* and *social security* number fields were largest separate causes for invalid submissions. As can be seen in Table 5 above, of these *social security number* was the sole invalid field in 22 submissions, while the *email* field was so only in four cases. This means that whenever the *email* field was one of the invalid form fields, it was a part of a cluster in the vast majority of cases. The next three individual form fields on the list, *first name*, *last name*, and *phone number*, were never the sole reason for an invalid submission, but always a part of a cluster: this implies that they belong to a cluster that was often overlooked.

Equipped with the knowledge of which form fields caused the user input on the sign-up form to be invalid, the form could be redesigned with data-driven insights to reduce the number of these invalid submissions. Fields such as the *plate number* and *acceptance of the terms of service*, which were relatively unnoticeable in the interface and which the data confirmed to cause some invalid submissions, could be made more salient in new versions. Clusters of fields that often caused an invalid submission, such as the *first name-last name-phone number-email* cluster on the *personal details* tab could also be made visually more prominent or appear at a different phase of the sign-up process.

In the course of going through the form submission error data, it became clear, again, that further instrumentation would have been useful. Had not only the names of the form fields that caused an invalid submission been recorded, but also what the user had typed into the field, if anything, more could have been deduced on why certain fields were problematic. If a required field was left empty, this could have been seen as indicating that the user had overlooked the field, whereas user input that did not match a specific format could have indicated a misspelling or a misunderstanding of what the form field was asking. For ethical reasons this additional instrumentation, however, would have been dubious: knowing what a user had typed into the *first name* or *social security number* fields when they were invalidated is a threat to the user's privacy. A sound compromise could have been achieved by recording just the information of whether the form field was empty or whether it contained a string that was programmatically invalidated.

6.3.3. Discussion

Though qualitative usability testing, for example, can teach us plenty on how users experience a given form and whether they have difficulties filling it in, insights on which exact form fields cause the most difficulties and how often, quantitatively speaking, they cause these problems in the wild can be gained only with a quantitative data collection method such as analytics. The in-page tracking approach was discovered to be helpful in collecting data on which form fields are invalid when the users submit

the form. This knowledge is useful in deciding if certain sections of the sign-up form should be designed for more fluid interactions.

Though the present research case was a fairly structured one and hence the results were quite actionable, they were not as actionable as in the first case study, which involved the A/B experiment. The results that one gains from an A/B test are directly actionable: the version of the interface that performs better in some desired aspect can be confidently set as the interface for the entire user population. The results of the current case are also actionable, but only indirectly: The data show that some fields on the sign-up form clearly cause trouble to the users, which indicates that the design of the form is not entirely successful. But what the data do not offer is an answer to the question of how the form should be redesigned; the redesign phase that could follow the research phase would then rely entirely on a designer's know-how. However, a study such as the one outlined in this section could also be seen as a helpful starting point for an A/B experiment: the redesigned form could be pitted against the old form and changes caused to the numbers of errors measured as the dependent variable.

7. Conclusion

In this thesis I have analysed behavioural trace data collected from two mobile applications and one web application. The reach and scope of the study were fairly broad; the studied applications had altogether close to 8000 active unique users during the sampled time periods. The data was collected using a method known as *software analytics* and with the goal of studying how analytics research could be used to improve these applications. What was meant with the notion of *improvement* was defined separately for each of the case studies. Furthermore, this thesis placed analytics into the sociological and psychological research traditions and into the toolbox of user research methods available to an adept HCI researcher. The strengths and weaknesses of this research methodology were compared and contrasted with those of other methods popular in the history and current practice of the field. With this theoretical background and research setup, plenty of insight into the applicability of analytics as a user research method that can be used to improve applications was gained.

The practice of using analytics as a way of setting up experiments and measuring the changes in the dependent variable was discovered to be especially potent way of improving software applications. In the industry, these types of experiments are referred to as A/B tests. Intuitions on how different interface designs might change the users' behaviour could be tested and the results presented with a degree of statistical significance attached to them. In the first case study, the metric that was defined for measuring success was in-app purchase conversion rate, but the measured metric could very well be anything else that can be measured with suitable analytics instrumentation. User engagement, as measured in longer use times or degree of user retainment, serves as an example of one such metric. Furthermore, the results of an A/B test were found to be highly actionable: the interface condition that was discovered to result in higher conversion rate could confidently be set for all users of the application.

Using analytics to perform feature use counts and screen view counts was studied with the help of the second case study. When the tracking involves gathering data from a

wide array of features, the instrumenting part of the research process was found to be laborious and subject to errors: the data set revealed that one such error had crept its way into the instrumentation of the studied application. Despite these nuisances, the feature use and screen view data were found to be valuable in showing how users interacted with the application in the wild. Equipped with these data, the designers and developers working with the application could make data-informed decisions on how to structure the interface of the application, for example, and hence improve it from the users' perspective. Furthermore, by studying the search feature of the application in more detail, certain events, such as the user getting zero search results, could be recognised as potential targets of improvement that would have considerable impact on the users' experience because of their high frequency of occurrence.

The third case shed light on how analytics can be used to study form filling. Through suitable instrumentation, trace of which form fields caused the form submission to be invalid could be recorded and studied. With knowledge of the pain-points in the form, the form could, again, be redesigned with the goal of reducing the number of invalid form submissions. As the success of the form was vital for the operation of the company behind the web application, any improvements in the sign-up rate would have a high impact on the financial performance of the company. However, the results of this research case were less actionable than those of the first two case studies and any improvement based on the results was left on the designer's expertise.

Through the results obtained from these three case studies, I have fulfilled the aims that I set for this study at the outset. Analytics was found to be an extremely useful user research method and the ubiquity of different analytics tools for measuring different aspects of user behaviour with very little cost make it a method that practically any researcher and designer with an access to a publicly deployed application can add to their toolbox. Analytics offers plenty of opportunities for improving web and mobile applications from a variety of points of view. Combining quantitative analytics data with more qualitative data collected with other research methods also appears a potent strategy: for instance, if the behavioural data collected with analytics reveals that a given sections of an application is rarely used, attitudinal data collected with traditional user research methods could be used to shed light on why this is the case. In some cases,

the inherent limitations of analytics data require the researcher to turn to this type of data triangulation.

Based on the research carried out for this thesis, the present author recommends a researcher or designer who is aspiring to start using analytics in their own projects to start from the structured end of the analytics research spectrum: the more structured the research project was, the more actionable the results in improving web and mobile applications were found to be. Unstructured research was discovered to be useful in shedding light on how a given application is used on a more general level, but these types of research projects require more work in instrumenting the application and in uncovering actionable insights in the data. In reality, a researcher who starts using analytics is sooner or later likely to find themselves in several rounds of gathering, transforming, and analysing the data, characteristics of semi-structured analytics research.

Though new and interesting analytics tools are at the moment appearing by the week, the principles of recording behavioural trace data that were discussed in this thesis are more enduring and offer a solid starting point for using the tools that will appear as the field matures. With these new tools, innovative ways of gathering and using behavioural data to study the interaction between humans and technology will undoubtedly emerge and continue to enrich the HCI field.

References

Akkus, I. E., Chen, R., Hardt, M., Francis, P., and Gehrke, J. 2012. Non-Tracking Web Analytics. *Proceedings of the 19th ACM Conference on Computer and Communications Security*, 687-698.

Apache Software Foundation. Log Files. *Apache HTTP Server Version 2.4 Documentation*. [Internet] Available at http://httpd.apache.org/docs/2.4/logs.html. [Accessed 20 September 2014]

Atterer, D., Wnuk, M., and Schmidt, A. 2006. Knowing the User's Every Move – User Activity Tracking for Website Usability Evaluation and Implicit Interaction. *Proceedings of the 15th international conference on World Wide Web*, 203-212.

Beasley, M. 2013. *Practical Web Analytics for User Experience: How Analytics Can Help You Understand Your Users*. Morgan Kaufmann.

Belkin, M. 2006. 15 Reasons why all Unique Visitors are not created equal. *Adobe Digital Marketing Blog*. [Internet] Available at http://blogs.adobe.com/digitalmarketing/analytics/15-reasons-why-all-unique-visitors-are-not-created-equal. [Accessed 24 October 2014]

Chen, R., Akkus, I. E., and Francis, P. 2013. SplitX: High-Performance Private Analytics. *Proceedings of the ACM Special Interest Group on Data Communication*.

Digital Analytics Association. The Web Analyst's Code of Ethics. [Internet] Available at http://www.digitalanalyticsassociation.org/codeofethics. [Accessed 30 September 2014]

Dykes, B. 2013. Web Analytics vs. Mobile Analytics: What's the Difference? *Web Analytics Action Hero Blog*. [Internet] Available at http://www.analyticshero.com/2013/07/24/web-analytics-vs-mobile-analytics-whats-the-difference. [Accessed 23 June 2014]

Finnish personal information law. 1999. [Internet] Available in Finnish at http://www.finlex.fi/fi/laki/ajantasa/1999/19990523. [Accessed 20 October 2014]

Fisher, C. and Sanderson, P. M. 1994. Exploratory Sequential Data Analysis: Foundations. *Human-Computer Interaction*, 9, 251-317.

Fitts, P. M. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology* 47, 6, 381–391.

Flurry. 2012. Flurry Windows Phone 7 API Class Reference. *Flurry Support Center.* [Internet] Available at

http://support.flurry.com/index.php?title=Analytics/Code/Doc/Windows. [Accessed 16 June 2014]

Google. 2014a. Event Tracking - Web Tracking. *Google Developers*. [Internet] Available at https://developers.google.com/analytics/devguides/collection/gajs/eventTrackerGuide. [Accessed 4 October 2014]

Google. 2014b. Introduction to Analytics.js. *Google Developers*. [Internet] https://developers.google.com/analytics/devguides/collection/analyticsjs/. [Accessed 4 October 2014]

Google. 2014c. New vs Returning Users. *Analytics Help*. [Internet] Available at https://support.google.com/analytics/answer/1144424. [Accessed 24 October 2014]

Google. 2014d. Overview of Demographics & Interests reports. *Analytics Help*. [Internet] Available at https://support.google.com/analytics/answer/2799357. [Accessed 2 October 2014]

Harris, J. 2005. Inside Deep Thought (Why the UI, Part 6). *An Office User Interface Blog*. [Internet] Available at http://blogs.msdn.com/b/jensenh/archive/2005/10/31/487247.aspx. [Accessed 3 May 2014]

Harris, J. 2006. No Distaste for Paste (Why the UI, Part 7). *An Office User Interface Blog*. [Internet] Available at http://blogs.msdn.com/b/jensenh/archive/2006/04/07/570798.aspx. [Accessed 3 May 2014]

Hilbert, D. M. and Redmiles, D. F. 2000. Extracting Usability Information from User Interface Events. *ACM Computing Surveys* 32, 4, 384-421.

Huber, O. and Schulte-Mecklenbeck, M. 2003. Information search in the laboratory and on the Web: With or without an experimenter. *Behavior Research Methods, Instruments, & Computers*, 35, 2, 227-235.

Jansen, B. J. 2006. Search log analysis: What it is, what's been done, how to do it. *Library and Information Science Research*, 28, 3, 407-432.

Jansen, B. J. 2009. *Understanding User-Web Interactions via Web Analytics. Synthesis Lectures on Information Concepts, Retrieval, and Services*. Morgan & Claypool.

Jansen, B. J. and McNeese M. D. 2005. Evaluating the Effectiveness of and Patterns of Interactions With Automated Searching Assistance. *Journal of the American Society for Information Science and Technology,* 56, 1480-1503.

Lazar, J., Feng, J.H., and Hochheiser, H. 2010. *Research Methods in Human-Computer Interaction*. Wiley.

Leiva, L. A. and Vivó, R. 2013. Web Browsing Behaviour Analysis and Interactive Hypervideo. *ACM Transactions on the Web*, 7, 4, article 20.

Martin, D.W. 2008. *Doing Psychology Experiments*. Thomson Wadsworth.

Münch, J., Männistö, T., and Pagels, M. 2014. Analysing User Behaviour to Provide Insights into Customer Behaviour. Poster presented at the Q2 2014 review of the Digile Need4Speed program, Tampere, 10 June 2014.

Russel-Rose, T. and Tate, T. 2013. *Designing the Search Experience: The Information Architecture of Discovery*. Morgan Kaufmann.

Sellars, W. 1963. Philosophy and the Scientific Image of Man. In *Science, Perception, and Reality*. Ridgeview Publishing, 1–40.

Shneiderman, B. 2008. Science 2.0. Science, 319, 1349-1350.

Skinner, B. F. 1953. *Science and Human Behavior*. Free Press.

Terry, M., Kay, M., Van Vugt, B., Slack, B., and Park, T. 2008. Ingimp: introducing instrumentation to an end-user open source application. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 607-616.

Web Analytics Association. 2008. Web Analytics Definitions. [Internet] Available at http://www.digitalanalyticsassociation.org/Files/PDF_standards/WebAnalyticsDefinitions.pdf. [Accessed 11 February 2014]

Webb, E.J., Campbell, D. T., Schwarz, R. D. and Sechrest, L. 2000. *Unobtrusive Measures*. Sage.

Wroblewski, L. 2008. *Web Form Design: Filling in the Blanks*. Rosenfeld Media.

W3C. 2011. W3C Workshop on Web Tracking and User Privacy – Call for Participation. [Internet] Available at http://www.w3.org/2011/track-privacy. [Accessed 2 October 2014]