Improved presentation of search results on a map

Magnus Johansson
Abstract

Improved presentation of search results on a map

Magnus Johansson

The purpose of this thesis is to build knowledge about geographically based content services and to investigate how improvements of presentation of search results on maps can be done. The main problem is how to present a lot of data on a map in a, for the user, readable and tangible way. A concept of a mobile application dealing with the problem is developed, tested on potential end-users by using paper prototypes and then implemented for Android mobile phones. The resulting application is tested on a couple of new users and results from tests are used to improve the concept and application. Throughout this thesis a user centered approach is adopted.
### Table of Contents

1. **INTRODUCTION** ............................................................................................................. 6

2. **BACKGROUND** ........................................................................................................... 7

3. **PROBLEM** .................................................................................................................... 8

4. **PURPOSE** ..................................................................................................................... 9

5. **THEORY** ....................................................................................................................... 10

   5.1. User centered system design .................................................................................... 10

   5.2. Iterative User-Interface Design ................................................................................ 11

   5.3. Testing the design ...................................................................................................... 11

   5.4. How many users to test on? ....................................................................................... 12

   5.5. Paper prototyping ..................................................................................................... 13

   5.6. Tools and mobile operating system ......................................................................... 13
        5.6.1. Android Mobile Operating System .................................................................. 13
        5.6.2. Ericsson Mobile Maps API .............................................................................. 14
        5.6.3. OpenStreetMap ............................................................................................... 14
        5.6.4. Eclipse IDE .................................................................................................. 15

6. **METHOD** .................................................................................................................... 16

   6.1. Existing solutions ...................................................................................................... 16

   6.2. Concept development ................................................................................................. 16

   6.3. User Centered System Design .................................................................................. 16

   6.4. Iterative design .......................................................................................................... 17

   6.5. Testing ...................................................................................................................... 17
        6.5.1. Paper prototype .............................................................................................. 17
        6.5.2. Prototype application ..................................................................................... 18

7. **EXISTING SOLUTIONS** ............................................................................................. 19

   7.1. Locago ....................................................................................................................... 19

   7.2. Guidu ......................................................................................................................... 19

   7.3. Layar ......................................................................................................................... 20

   7.4. Google Maps Navigation .......................................................................................... 20

   7.5. Bing Maps ............................................................................................................... 20
8. CONCEPT DESCRIPTION ................................................................. 22

8.1. Concept ................................................................................. 22

8.2. Detailed solution .................................................................... 25
  8.2.1. Startup ............................................................................. 25
  8.2.2. Searching .......................................................................... 26
  8.2.3. What data will be used? ..................................................... 26
  8.2.4. Rank objects ...................................................................... 27
  8.2.5. Slice map .......................................................................... 28
  8.2.6. Present on map ................................................................... 33
  8.2.7. Icons ................................................................................. 34
  8.2.8. Many objects in one slice ................................................... 34
  8.2.9. When objects are stacked on the same point ..................... 34
  8.2.10. Displacement of objects ................................................... 36
  8.2.11. Search area shown on screen ........................................... 36
  8.2.12. Pointers to off-screen items ............................................. 37
  8.2.13. What happens to the map view when zooming? ............... 38

8.3. Differences from the original Ericsson Invention .......................... 39

9. RESULTS .................................................................................... 40

9.1. User Centered System Design ................................................ 40

9.2. End-user testing ...................................................................... 41
  9.2.1. Preparation of the paper prototype usability test .................. 41
  9.2.2. Testing the concept – first test ......................................... 41
  9.2.3. Testing the prototype – second test .................................... 41

9.3. Results from the usability test on the paper prototype ................. 41

9.4. The prototype application ....................................................... 43

9.5. Results from the user test of the actual prototype ....................... 44

9.6. Implications of the results ...................................................... 45

10. IMPLEMENTATION ................................................................. 46

11. ANALYSIS .............................................................................. 47

11.1. User centered system design ................................................ 47

11.2. Usability testing using the paper prototype ............................ 48

11.3. Differences between concept and the prototype application ....... 48
  11.3.1. Implementation problems ................................................ 49

11.4. Usability test on the android prototype application .................. 50

12. DISCUSSION AND CONCLUSIONS ......................................... 52

12.1. Future work ........................................................................... 53

13. LITERATURE .......................................................................... 54
13.1.  Printed sources.................................................................................................................................54
13.2.  Internet sources..................................................................................................................................55
1. Introduction

This thesis investigates how to show search results on a map in a usable, readable way. By map I will in this thesis be referring to a map in a computer or mobile phone. Search results on a map, consequently, mean the results obtained from searching various databases containing geographically tagged information and placing these results on a map to see their creation or current location. As more services, social networks, stores, restaurants etcetera start adding locations to their offerings and items, on a map containing these search results it can in some situations be hard not just to see the map, but also to see all the individual objects placed on the map. Examples of geographically tagged items that can be shown on a map are not just stores, restaurants and similar common points of interest (POIs), but also data like pictures, user posts (from for example Twitter or Facebook), videos, locations of a Wikipedia article, events and so on. This thesis examines how to present these types of data in a, for the user, readable and structured way. This will be done by looking at and evaluating attempts to solve these issues by services and applications currently available, after which a concept of a mobile application dealing with the issue will be developed, tested and implemented. The implementation will be, at the request of Ericsson Research, where the thesis work was carried out, done for Android mobile phones.
2. Background

A map can be of help when navigating in unfamiliar territory. But, the usage of maps is not limited to finding or getting to a location. Along with the Internet and as maps could be used in computers and mobile phones a lot of new areas of usage appeared. Today, there are many programs and applications that make use of, and display, geographically tagged content in various ways. For example showing where photos are taken, where someone posting a message is geographically located, information about landmarks, restaurants and such are examples of things that can be bound to a location and shown on a map. One feature of a map is that it is, usually, a scaled down representation of a big area, so that this area can more easily be overviewed. When geographically tagged information is shown on the map it can easily result in a superfluity of information. This frequently occurs in cities and areas in cities where there is a lot of activity, and therefore a lot of things that can be shown on the map. This is the main problem focused on in this thesis.
3. Problem

There are many map applications and services out there today, where some are for use in mobile phones and others are not. What most of them have in common is that they encounter difficulty when attempting to display large amounts of data on the map using a limited screen area. When searching for, and showing, information on a map like in Google Maps, Bing Maps or in any other map service, one problem is that items and their icons start overlapping and covering big parts of the actual map. This can make it difficult to see parts of the map, especially on more zoomed out levels. Not only regular map points are shown, but also geographically tagged data of various sorts are presented on the map to show where they are situated or created. Data that are subject for tagging geographically are not just stores, restaurants and other common points-of-interest, but also different types of data like videos, pictures, tweets and other data from social media etcetera. Many of the map services out there today provides data views that are messy. Too many objects are shown too close to each other so that individual objects are hard to make out and sometimes the map even gets hard to see behind all these objects. In high density areas in big cities a lot of geographically tagged data tends to be created and positioned. It is in these areas problems, such as overlapping and objects covering parts of the map become most apparent. Some services or applications “solve” this problem by showing a limited number of search results or list the data in text views on the side of the map instead of showing the objects on the actual map.

The main problem investigated in this thesis is how to present a lot of data on a map in a, for the user, readable and usable way.
4. Purpose
This thesis is based around a patent solution that Ericsson Research holds. This patent deals with how to show search results on a map in an improved way. The purpose of this thesis is to build knowledge about geographically based content services and investigate how improvement of presentation of search results on maps can be done. The Ericsson patent solution will be evaluated and improved where possible. A concept of a mobile service displaying a selection of Points-of-Interest for mobile phones will be designed and implemented using a selection of suitable geographical application programming interfaces (APIs). The implementation will be done for Android mobile phones and will use the Ericsson Mobile Maps API as map source.
5. Theory
This part of the thesis will cover theory on user centered system design, how to work in an iterative way, how to test a design, how many users to test the design on and some about paper prototyping. Some information about the Android mobile operating system, which will be used for implementing the concept that will be developed in this thesis, and also the tools that will be used when constructing the prototype application are here briefly covered.

5.1. User centered system design
User centered system design (UCSD) is a design process where usability is focused on throughout the whole development process and throughout the system life cycle. According to Gulliksen et al. (2003) UCSD is based upon 12 key principles:

- **User focus** – goals of the activity, the context of use, the users’ goals, task and needs should early guide the development. Who are the users and what does their tasks look like are questions that all members of the project should know the answers to. This helps creating and keeping the focus on the users’ needs instead of a technical focus.

- **Active user involvement** – involve users actively, early and continuously throughout the development process. The users involved must be representative of the real intended user groups and plans for user involvement should be specified from the very start of the project.

- **Evolutionary systems development** – the systems development should be iterative and incremental. UCSD requires continuous iterations with users and incremental deliveries that can be evaluated before made permanent.

- **Simple design representations** – the design must be represented in such a way that it can easily be understood by users and all other stakeholders. Usage of prototypes and simulations that are more easily understood then UML diagrams or requirement specifications. The representation must be usable and effective so that all parties share an understanding on what is being built.

- **Prototyping** – early and continuously, prototypes should be used to visualize and evaluate ideas and design solutions in cooperation with the end users. Multiple sketches, mock-ups and prototypes should support the creative process. Prototypes should be designed and evaluated with real users in context.

- **Evaluate use in context** – baseline usability goals and design criteria should control the development. Developers observe the users’ reactions to paper sketches and mock-ups, later in the project users should perform real tasks with simulations or prototypes and their behavior, reactions, opinions and ideas should be taken into account and analyzed.

- **Explicit and conscious design activities** – the development process should contain dedicated design activities. Remember that for the users what they can see, the user interface, is the system.

- **A professional attitude** – the development process should be performed by effective multidisciplinary teams.
• **Usability champion** – usability experts should be involved early and continuously throughout the development lifecycle. There should be an experienced usability expert or group on the development team.

• **Holistic design** – all aspects that influence the future use situation should be developed in parallel.

• **Process customization** – the UCSD process must be specified, adapted and/or implemented locally in each organization. Usability cannot be achieved without a user-centered process.

• **A user-centered attitude should always be established.** UCSD requires a user-centered attitude throughout the project team, the development organization and the client organization. *(Gulliksen et al., 2003, p.401-403)*

5.2. **Iterative User-Interface Design**

Even the best usability experts can’t design a perfect user interface (UI) in a single attempt. Because of this, interface designers should build a usability engineering lifecycle around iteration. With the help of user testing and other evaluation methods, the interface should be redesigned and refined. When a design is complete, it is tested and the problems that several test users have with the interface are noted. These problems are fixed in a new iteration, which then is tested to make sure that the problems were resolved and to see if any new problems have emerged. If the designers are to choose between two or more interface design alternatives, they can test the alternatives and see which one is the most usable one. The aim of iterative design is to refine the interface based on lessons learned from earlier iterations. *(Nielsen, 1993, p.32).* Nielsen, in the same article, writes that a system is usable if it is easy to learn, efficient, easy to remember, relatively error-free and pleasant to use.

Iterative design can yield a larger improvement between versions of the same design alternative than the results from picking the better out of two average design alternatives. We can expect that the largest improvements of an interface from the first iterations, as the major usability catastrophes are discovered and removed. A recommendation is to go beyond the first iteration, because designers sometimes introduce new usability problems when fixing old ones *(Nielsen, 1993, p.41)*

5.3. **Testing the design**

Testing of a design or a concept can be done in many different ways. Nielsen & Monlich (1990) state four ways of evaluating a user interface: Formally by some analysis technique, automatically by a computerized procedure, empirically by experiments with test users, and heuristically by looking at the interface and passing judgment according to one’s opinion *(Nielsen & Molich, 1990, p.249).* In their paper Nielsen & Molich (1990) recommend that heuristic evaluation is done with between three and five evaluators and that it has the advantages of being cheap, intuitive and easy to motivate people to do. It can also be used early in the development process. Heuristic evaluation involves having usability specialists decide whether each dialogue element follows the heuristics, the established usability principles *(Nielsen, 1994, p.413).* Heuristic evaluation has also been proposed as a substitute for real empirical user testing because it can be performed based solely on the specifications
of an interface, thus no working prototypes would be needed (Nielsen and Phillips, 1993, p.220-221). Disadvantages of using heuristic evaluation include that several evaluation experts are needed and that it discover “false alarms”, issues that users may never be bothered with in real use (Hollingsed and Novick, 2007, p.250). Other authors claims that while heuristic evaluation is a cheap, fast and easy-to-use method, it often finds too specific and low-priority usability problems, including not even real problems (Hwang and Salvendy, 2010, p.130).

Empirical methods, like user testing, are the most commonly used methods when evaluating interfaces. Real users can sometimes be hard or expensive to acquire enough of in order to test all aspects of all the versions of a design continuously evolving. Studies have shown that usability inspection methods find problems overlooked by user testing and vice versa, so in order to get the best results a combination of the two should be employed. (Nielsen, 1994, p.413)

5.4. How many users to test on?

The question of how many people to test a design on has been the subject of significant debate. Hwang and Salvendy (2010) bring up the question of sample size, how many users/evaluators are needed to reach the targeted overall discovery rate. They claim sample size of usability evaluation depends on an estimate of problem discovery rate across participants and to achieve an overall discovery rate of 80% eight evaluators are needed when doing heuristic evaluation. Jakob Nielsen (Bevan, Barnum, Cockton, Nielsen, Spool & Wixon, 2003) recommends planning for 5 users but claim that 3 - 4 users is the optimal number for most usability studies. He states that this is not a perfect study that will discover everything possible to know about the design. But it is an acceptable trade-off in return for having more iterations in the design process and conserving resources for evaluations of these iterations. According to Nielsen (2000) the reason of using five users to test the design on is that after the first user there will always some overlap in what you learn from the test. As more users are added, you learn less new information because you will keep seeing some of the same things over and over again. (Jakob Nielsen's Alertbox, 2000)

Carol Barnum (Bevan, Barnum, Cockton, Nielsen, Spool & Wixon, 2003) suggests that five users is only enough for certain test teams with close cooperation and good communication, when the original discount model for testing is followed and when the expected result is insight and not validation. Jared Spool (Bevan, Barnum, Cockton, Nielsen, Spool & Wixon, 2003) writes that N users will always be better then zero, for whatever N chosen. And that design teams are limited by time and resources and will test as many users as they can within their constraints. They’ll be done when they are done. In the same paper, Gilbert Cockton writes that five users can be enough some times but that those times are rare. His opinion is that one can’t know in advance how many users will be needed for testing and he says that usability is about risk management. The risk diminish as we test more users and that for one product one test user might be enough but for others 100 test users will not even suffice. Dennis Wixon (Bevan, Barnum, Cockton, Nielsen, Spool & Wixon, 2003) is of the opinion that selecting how many test users to have is the wrong problem to study. The goal is to produce the best product/design in the shortest time at the lowest cost with the least risk and therefore verifying that the fixes made actually address the users’ concerns is what should be focused on. Most authors seem to agree on one thing - that some usability testing is better than none. In this thesis the developed concept will be tested, first in the form of a paper
5.5. Paper prototyping

Prototyping is a fast way to get direct feedback from users on a design or a concept. Paper-based prototypes save time from making a coded user interface. One reason of using paper prototypes is that it is easier to involve the whole team working on the project and changes to the design can be made on the fly. Users also feel more comfortable criticizing a paper prototype that does not have a polished look, and you will save time by making conceptual changes at an earlier stage in the design process. When users are testing tasks that include data, the data should be as real as possible although it is often unpractical to have a lot of data available. The test will give better results if realistic data are used. (Klee, Five Paper Prototyping Tips, 2000)

By definition if something has a user interface it can be prototyped, but paper prototypes are less suitable for visually complex or highly interactive interfaces such as video games. Prototyping mobile devices or a device operating in a rich contextual environment is more challenging, but for many business and consumer applications, paper prototypes are considered to be quite useful. Paper prototypes are great for finding problems including concept issues, terminology, navigation, workflow and screen layout. (McKee, Streamlining the Design Process with Paper Prototyping: An Interview with Carolyn Snyder, 2007)

5.6. Tools and mobile operating system

In this section the tools and the mobile operating system used for programming the prototype application are covered. That the application should be done for Android mobile phones and that it should use Ericsson Mobile Maps API were requirements that Ericsson had in the thesis specification. Eclipse IDE is a programming tool that the author has previously used in some courses at university level, and this tool was also recommended by supervisors at Ericsson Research.

5.6.1. Android Mobile Operating System

Because the implementation of the mobile service targets Android phones, some basic information about the mobile phone operating system is presented.

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. It allows developers to write code in the Java language and control the device via Google-developed Java libraries. Necessary tools and APIs (Application Protocol Interfaces) to develop applications on the Android platform using the Java programming language are supplied by the Android SDK (Software Development Kit). Android is based on a Linux kernel and relies on Linux version 2.6 for core system services like security, memory management and process management. All applications in the Android environment runs in its own process, with its own instance of the Dalvik virtual machine. For underlying functionality like threading and low-level memory management the Dalvik VM relies on the Linux kernel.
In Android one application can use elements of other applications (if the application permits it). Therefore the system must be able to start an application process when a part of it is needed, instantiating the Java objects for that particular part of the application. Because of this Android applications don’t have a single entry point for everything (a main( ) function). Instead they have four types of components that the system can make instances out of and run them as they are needed. These four are Activities, Services, Broadcast receivers and Content providers. Android makes sure that the application process of the called component is running, if not Android starts it. Components might be shut down by the system when no longer used, or when Android needs more memory for more active components. Activities, services and content providers must be registered in the AndroidManifest.xml file else they won’t be visible to the system and thus never run. All Android applications have a manifest xml file included in their .apk file, which also holds the application code, resources and files.

In the Android environment one activity can start another activity and that is what, from a user perspective, is viewed as an application. In reality Android just keeps many activities in one same task, and it is a task that is perceived as a real “application”. In a task activities are arranged in a stack and the activity at the top of the stack is what is currently running (is focused). When an activity starts another activity the new one is pushed on the stack and it focused. If the started activity is closed it gets popped from the stack and the calling activity again receives focus. (http://developer.android.com/guide/topics/fundamentals.html) (Android Developers homepage)

5.6.2. Ericsson Mobile Maps API

Ericsson Mobile Maps is an application programming interface developed at Ericsson Labs. It enables developers to easily add map functionality to mobile phone applications. Mobile Maps API is vector based and renders the maps in the mobile phone application, therefore using very low bandwidth compared to other map services that downloads the image tiles. When rendering the maps in the application, the developers get more control and flexibility when creating applications. This also results in faster running applications and lower data costs for the users of the application. Mobile Maps is based on the Java programming language and is available for both Java ME and for Android. Map data is provided by Open Street Maps as it is an exponentially growing database that users continuously add data to. (Ericsson Labs Homepage)

5.6.3. OpenStreetMap

OpenStreetMap is a free editable map of the world. It lets you, in cooperation with others, show, change and use geographical data from around the globe. OpenStreetMap is a project with the intention of providing free geographical data such as street maps to anyone who wants them. The project was started because most maps that claim to be free, have legal or technical restrictions that discourages people from using them in creative or unexpected ways. By “Free maps” we are not primarily referring to the map cost being zero, but that the maps can be used freely without hard restrictions and copyrights. This is the same notion of freedom as for open source code like the operating system Linux or the free content of projects such as Wikipedia. OpenStreetMaps was founded in July 2004 by Steve Coast, a PhD in computer science at the University College London. OpenStreetMaps serves as a source for map data for Ericsson Mobile Maps API. (OpenStreetMap homepage)
5.6.4. Eclipse IDE

Eclipse is a multi-language software development environment comprising an integrated development environment (IDE) and extensible plug-in system. It is primarily written in Java and can be used to develop applications in Java and, with help of the various plug-ins in other programming languages as well. With the Android Development Tools (ADT) plug-in for Eclipse one can create and debug Android applications easier and faster. During the development of the Android mobile application prototype Eclipse with the ADT plug-in will be used.

(Eclipse homepage)
6. Method

The methods used to solve the problem of this thesis are evaluation of existing map services and applications, developing a concept for a mobile phone application, testing the design using a paper prototype, implementing the concept for Android mobile phones and performing some user-testing on the resulting application. All this while working in a user centered way with iterative steps in the design and implementation.

6.1. Existing solutions

Prior to everything else an evaluation of existing map services and applications was performed in order to get an overview on what is today available. When conducting this evaluation, no real guidelines were used other then aiming on collecting information about the service in question and finding apparent pros and cons with them. This was done to obtain some knowledge about how the various services handled displaying data on the map and how browsing through data on the map was handled. The downsides found in the various services and applications were to be avoided when developing the concept and the positive things were to be taken inspiration from.

6.2. Concept development

The development of a concept description for the mobile application was based on a patent solution owned by Ericsson Research. A part of the thesis was evaluating the patent solution, coming up with improvements if possible and while the concept was based on the patent, some improvements were made during the evaluation. The changes that were made were based on the test of map services and the user testing of the paper prototype of the concept.

6.3. User Centered System Design

Throughout this thesis a user centered approach was adopted. This includes involving the users at an early stage (concept stage), using early prototypes and testing them iteratively. The application was to be developed in an incremental way where parts could be tested before the application itself was complete. Through using paper prototypes and scenarios simulating possible real usage of the application as well as noting all test-user reactions and comments, users were involved in several parts of the design and development process. No real usability expert was involved in any part of the concept or application development.

As it is hard to get a hold of a usability expert (with limited time and resources) let alone obtain access to several usability experts, heuristic evaluation doesn’t seem like an ideal choice when evaluating the design and concept in this thesis. With more at stake than just usability testing, empirical testing using potential real users seems like a more appropriate, quicker and easier way of testing the concept and design. That way some diversity in background, age and skills is also introduced in the testing.
6.4. Iterative design

Two tests was conducted, one before the implementation of the concept and one on the resulting prototype application. Testing two times, once before the implementation of the concept and testing again after the programming part of the thesis was done, provided some iteration steps in the development process and a better end result. The first test was done on a paper prototype of the application and the second test was done on the prototype application on an actual Android mobile phone. Both tests were followed with an analysis of the results obtained, after which adjustments in the concept design or application could be made.

6.5. Testing

Finding eventual problems such as the concept issues, terminology, navigation, workflow and screen layout would be very beneficial for the development of the concept that the application in this thesis will be based on. In agreement with my supervisors I therefore found paper prototyping to be a suitable testing ground for the concept. The concept and application will be tailored towards Android mobile phones and because of that the pieces of paper resembling the interface in the paper prototype was about the same size as that of the screen of an Android mobile phone with a touch screen to make the test as real as possible. The number of test-users aimed at was five to start with, so that most of the testing could be conducted in one day each. As no real consensus on how many to test on exist in theory, some saying that one cannot test enough and others that 4-5 is enough. Because of the concept not being too complex and not having too many functions to test, five test-users seemed enough to start out with. The two tests ended up having six test-users each from which a lot of useful observations and information could be extracted.

6.5.1. Paper prototype

When the concept description was somewhat complete, a paper prototype of the application was constructed in order to test the concept as well as the application. In the paper prototype testing that was conducted in this thesis the focus was not on real data per se, but when testing screenshots of real maps were be used as map in the paper prototype. The map in the paper prototype will be real but the places (represented by icons) on the map will be made up, in order to better fit the made up but seemingly real scenarios, that the testing will be based on. The paper prototype was tested on six test users with different types of backgrounds. Some had more experience of mobile phones and map applications than others and the testers were between 15-56 years old. The test-users got to navigate through the paper prototype where most part’s of the application content was covered. After testing, the users answered questions regarding what they thought of the application, what they thought could be improved and if the user interface was easy enough to understand. Empirical testing on a paper prototype as a method of testing a concept or design is a cheap and easy way to test things before actual programming is done. The limited timeframe of this thesis made this an attractive way of testing as it did not require too much planning and could be used early on in the development process. A downside of using empirical user-testing is that it can identify usability problems without any direct solutions. An attempt to avoid this was designing the questions presented after the test in a way that got the test users thinking of what they thought needed to be improved about the program and if they had any ideas on how to accomplish this. This made it possible to use input from the test-users in order to create solutions to the problems that were discovered. Benefits of the first user test being on a paper prototype, other than that it is easy to change the prototype on the fly, was the fact that, in theory, users are supposed to be
more comfortable criticizing a paper prototype and one can after testing on a paper prototype save time by making conceptual changes at an early stage in the design process. Paper prototypes are not suitable for all types of testing, sometimes testing the real prototype will give more to the creators then testing a paper prototype will. The combination of testing the concept on a paper prototype, designing and programming the prototype application and then performing a second usability test provided some iteration steps in the testing and designing.

6.5.2. Prototype application
The second test, performed on six new test users, was a kind of validation of the first test and helped to determine whether the real prototype was usable and understandable in terms of navigation, terminology and layout. This test provided more feedback on the concept as a whole, as the users themselves easier could navigate and try different features in the real mobile phone prototype then they could in the paper prototype test. The paper prototype was limited as it was a bunch of printed pieces of paper with drawings representing the application while this test was freer including most parts of the concept.
7. Existing solutions

Examining some of the existing map services and applications

Before the concept description was written, some research was conducted on what map applications and services existed on the market and how they handled different problems, for example how they show geographically tagged information on their maps. Here follows a summary of some of the services and applications that was tested. The first two are applications developed at Ericsson using the same map API (Ericsson Mobile Maps API) that will be use when constructing the prototype.

7.1. Locago

Locago is a java based mobile map browser with directions and GPS support that was developed using the Ericsson Mobile Maps API. Here different types of information and data are divided into layers and the user can view and add new layers to their map. Some of the layers available are business listings, traffic info, Wikipedia articles, weather forecasts, and other information that can be displayed on a map. Some layers use a live connection to the source to get the latest data (photos from Flickr, wiki, weather), while others are compiled into a static (speed cameras and the WCities layers). Within each layer the user can step between the objects that are currently showing on the screen, and map, in that layer using up/down on the phone joystick. Locago is built on an open API and therefore users can create and publish their own geographic data if they so please. Some of the pros with Locago are that the program is quite easy to use, layers can be flipped between easily and locations and layers can be bookmarked for easier access later on. Cons on the other hand are that since geographically tagged data often gets bunched up around certain areas (cities and specific areas in cities) it often get displayed as a pile of objects on the map and because of this it gets hard to make out individual objects and the map behind the objects cannot be seen at all. In this application they have tried to overcome this problem by showing ten results at the time for one layer, but this also reduces the usability because if the user is looking for, say that 11th object he has to specify his search further in order to narrow down the results.

7.2. Guidu

Guidu is a mobile application that enables the user to tell the world what they are doing by updating their Twitter from their mobile phone. It also lets the user keep track of what their friends are doing by showing their updates. This application also provides the user a fully featured map in which the user can search for and find information. Guidu uses Fire Eagle to keep track of where the user is. Fire Eagle is a site that stores information about the user location, with their permission. Other services can update this information or access it. Pros with Guidu is that it is quite easy to use and location tagged tweeting using Fire Eagle works well when it has been set up. Cons are similar to those of the previous service, it is troublesome to flip through the objects at a specific location. To do this one has to center the map on that position and this is impossible without zooming in a lot when objects lie close to or on top of each other.
7.3. Layar

Layar is a reality browser that displays real time digital meta data on top of what you can see through the camera of your mobile phone. The camera can be pointed in a direction and the user can see layers of information on top of real world objects. The layers can be stores, bars, tourist information, tweets and other things that can be shown on a map. Besides the reality browser view, the user can select to use map view which looks a lot like a regular GPS application. Layar is available for both Android and iPhone mobile phones and works with the GPS and compass of the phone to determine location and what direction the user is facing. The location is used to retrieve the layer information to be displayed in the layers of the application. Pros with this application are that it is an exciting and intuitive way of displaying information, point the phone to get information about places in that direction. In addition to being “normal” map application the reality browser provides instant direction for objects in walking distance. When objects are in the same area or direction it is difficult make out specific objects, especially in the reality browser. Objects overlap and a list view has to be brought up to view the overlapping objects one by one.

7.4. Google Maps Navigation

It is a free turn-by-turn navigation with street view, satellite imagery and voice-powered search for Android OS 2.0 only, for now. Addresses are input by text or voice, but also business names, restaurants and soft queries like “that museum that has the King Tut exhibit” can be searched on and return a list of suggested locations. It takes data from local road authority, highway cameras, and services like INRIX\(^1\) and also from other user’s cell phones. Maps cache along the intended route so even if the connection dies the user can see what they need to. One can search for locations (petrol stations, restaurants) along the route and the results will show up on the map as long as they are within a radius along one’s path. Google also say that it is easy for them to also add more layers to it. There is a landscape and portrait mode, plus a big-icon UI for using the software when driving. Positive things about Google Maps Navigation include GPS navigation, auto updated maps, smart search and possibility to add more layers. Downsides are that the voice search can only be done in English at the moment and I had some trouble finding the street view feature and apparently I wasn’t the only one. Since last update one can only tell if street view is available by zooming in or long pressing a location an look at the address bubble and see if it has a picture on it or not. Google Maps Navigation shows all data from the search on the map, so at popular areas or locations objects get stacked, and it is hard to see individual objects and the map behind them.

7.5. Bing Maps

Bing maps is an online map similar to Google Maps with different layers or map apps as they call their layers. One data type they include in their map apps is the Photosynth layer, which uses photosynth.net which makes 3D views of pictures taken in a circle at one location, kind of like the street view of Google Maps, but of other places then streets and intersections. They also have layers like what is nearby, local events, street side photos (from Flickr), current traffic, today’s front pages, and restaurant finder and so on. Bing maps has their own solution

---

\(^1\) INRIX is a traffic services company providing historical and real-time traffic information to businesses and individuals in the United States and Europe.
when many objects for example tweets, came from the same location, then they just had them under one icon that could be clicked and then expanded to a circle of icons around where the original icon was located on the map. This made the map easier to see when the icon was not expanded.

Figure 1: Left picture - an icon not expanded, right picture - the same icon when clicked on.

Even with this gathering of objects under one icon the map was sometimes hard to make out behind the many object icons on the map.
8. Concept description

Improved presentation of search results on a map in an Android mobile phone

This concept includes:
- What data and type of objects that will be searchable and shown on the map
- Searching for data in a specified area
- Extracting and gathering of data from different sources
- Ranking data, to display the most relevant data for there and then
- Slicing the map into smaller areas for easier data distribution
- Slicing of the map by a given slice method for the specified zoom level and map size
  - How to slice map
  - Re-slice map when zooming on the map
- Position the objects in each slice according to the ranking criteria
- Present results on the map according to some presentation rules
  - How the data will be presented and viewed on the map

8.1. Concept

The basis of the concept is a mobile phone application that displays points-of-interests and other geographically positioned data on a map. What this application does is that it presents (search) results on the screen of a mobile phone on a map in a way that, even if there is a lot of data, the map is still visible and the individual items are still clickable. When the application is started the user is taken to the map that is showing the area the user is currently in. The user’s position in a preset zoom level will be showing on the map along with location based data existing in the area.

When performing a search somewhere on the map, the search area will be set to a circle around the centre of the map view with a diameter of what is vertically visible on the screen (see figure 1). The user can always zoom in, out and navigate on the map until they are happy with the area that is on the screen before searching. Parts of the circular search area will end up to the left and to the right of the current screen view and objects there will be indicated by pointers at the left and right edges of the map view (figure 1). A specified search, for example on the name of a restaurant, can be done to narrow down the number of results given. When choosing My Location then Search, objects close to the user’s current location will be shown.
There are two cases to consider when searching for objects within an area. One is when we get too many search results to present them all at the same time and the other is when we get zero results. The ones in-between are not a problem if the two extremes can be handled.

If we are presenting all search results at the same time and there are 500 objects to be displayed, showing all of them at once on one mobile screen would not be readable or useful from a user perspective (figure 3). The idea then is to rank all the location data we get from the search to be able to sort it by its rank. Ranking is done in different ways depending on what data it is. Examples of how to rank data are by creation date, distance to user, search word matching, and number of views and so on.
Figure 4: 48 icons containing 78 actual items shown on a mobile phone screen-like area.

The results will be displayed and objects that are in the same slice will be stacked on top of each other. To view objects in a stack, the user clicks the object icon and is then able to flip through the more detailed object view for that stack, one object at a time. The order in which the objects will appear will be according to the relevancy ranking. The relevancy ranking within a stack can also be changed by the user. When the user wants to see any one particular object on the map they can click “Show on map” and just the one object along with the search point will be shown on the map (Figure 6).

Figure 5: Flipping through a stack of 4 different objects.

The map will be sliced using square slices because they are easy to use, makes placement of the icons easy at the same time that they combined with circular formed icons gives us some space between objects in different slices so that the map behind them can be seen. Some of the other slicing methods tend to waste screen real estate or give us an unbalanced distribution of objects.
If the user gets zero results when searching in an area, they will be prompted to zoom out or choose a different search point. Another way of handling cases where there are no objects in the search area is to expand the search area and search until an object is found.

8.2. Detailed solution

8.2.1. Startup

When the application starts the user is taken to map view centered on the user position at a predetermined zoom level. An appropriate zoom level might be a view where the users can see the map of a 1 km radius surrounding their current position. When starting the application, objects in the area where the user is situated are shown on the map. To get the map view centered on the user the function “My Location” will be called upon startup. My location will use the GPS in the phone to get latitude and longitude of the user and show the user location in the middle of the screen of the mobile phone. This function will also be accessible from the menu of the Android mobile phone.
8.2.2. Searching
The search function will be accessible from the menu in the map view. When searching, the objects will be retrieved from their respective source and ranked. The objects will be placed in the slices on the map according to their ranking and the map view will be updated to include the search objects.

Area searching
When searching, the program will always search within a certain radius around a given point. As default this point is where the user is standing but it can also be specified by the user by selecting a point on the map to base the search around. This point is selected by simply scrolling the map to wherever the search should be based around. The search radius can be specified by the user by zooming in or out at the particular location. The search radius based on the zoom level and set to be a little more than the height of the map view on the screen of the mobile phone. The majority of the search area will be shown on the screen along with arrows pointing at objects, inside the search area but outside of the map view, which might be of interest to the user. Search objects will be retrieved from the area within a circle of the radius specified by the zoom level.

Specific search
If a user only wants to see apartment ads, then he should not have to search the map and every pile of objects on the map before he can find one. Therefore, in the search it can be specified what data the user want to search for. If they only want apartment ads then they can set the search to only search for and only show apartment ads. This can be applied to any data type. Say the user only wants to see Twitter posts and pictures from Flickr, this can then be set up in the search.

Searching can be done not just in an area but also on a specified search query. If the user is looking for a football field, they can specify the search to look for a Point-of-Interest along with the search term football. Should the user be interested in a particular place that may be represented on the map by many different data types at the same time, they can search for the name of that place to get all data corresponding to it.

When there are no objects in the search area there are two solutions. One is to tell the user to search in a different area or at a different zoom level. Hopefully a more zoomed out level will lead to finding some objects. Another way to handle empty search results is to automatically expand the search area and search until an object is found. The direction in which the object is located can then be pointed to, or the zoom level could change so that the object will be visible on the map. Before an expanded search is performed the user should be told that there were no hits in the area and asked if they want to expand the search until an object is found. If they want to expand the search, the search area will be based on the next zoomed out level to search within, zooming out and searching until an object is found.

8.2.3. What data will be used?
The concept itself includes using many different data sources to retrieve geo-positioned data from. For the implementation of the concept a selection of freely available sources has been made. The data sources that can be used are for example Open Street Maps, Twitter, Flickr, Booli, Wikipedia and weather information from Yr.no.
From Open Street Maps we can get a square area specified by two latitudes and two longitudes that contains information about the places inside the area. This information can be exported as an XML file from where the individual locations can be taken. (OpenStreetMaps homepage)

One single Twitter message is called a tweet. A tweet is a text based message with a limited length of 160 characters. Geographical spots can be searched for tweets in the area and Twitter has an API that developers can use to more easily make use of the different functions the service provides (Twitter homepage). A tweet will be represented on the map by a “t” icon similar to the real Twitter logo.

Flickr is an online photo management and sharing application that lets its users share photos and videos with others (Flickr homepage). A Flickr photo is another type of geographically tagged data that can be presented on a map. Flickr has, just like Twitter, an API that is available for non-commercial use by outside developers. Photos from Flickr will be another type of data that will be used to illustrate the benefits of the application. A Flickr photo is represented on the map by a Flickr icon, or by a small thumbnail of the picture in question.

Booli is a search engine for housing ads that is easy to use and a quick way of searching for available homes for sale in Sweden (Boolis hemsida). Booli has an API that can be used for getting apartments and houses for sale and rent along with their location. On the map apartment ads will be represented by an orange Booli icon with blue text.

GeoNames is a free geographical database that covers all countries and contains over eight million place names. Its Wikipedia web service give access to geographically referenced Wikipedia articles in 240 different languages and the GeoNames database contains about 1.2 million Wikipedia articles. Wikipedia articles will be provided from the GeoNames database and their API for retrieving articles in an area. (Geonames homepage)

Yr.no can provide the user with information about the current weather conditions and what is about to happen in a few days at the specified point on the map. Weather can be associated with every point on the map where some type of object exists, but because weather is area specific and not coordinate specific it is not a high priority data type when making the prototype application that this thesis includes. (YR homepage)

8.2.4. Rank objects
There are many ways to rank the objects extracted from the result of a search query.
- What are the most popular objects in the area?
- What objects are closest to the users’ position? (distance to)
- Descriptiveness score or number of views? (for media such as photos or videos)

The rank an object has will decide their importance when showing them to the user. The most important objects will be shown first and when objects are stacked under one icon the rank decides what object will be shown where, when browsing through them. We can’t, for instance, use only object creation date to rank all the different data. A restaurant or landmark could instead receive its ranking based on its distance to the user (or search point), while a tweet can be ranked by its creation date. Starting out from the search point the objects in a radius around the search point can be ranked in the following ways.
**Ranking regular points-of-interests**

When the different POIs in a search are ranked, the most popular objects in the search area will be ranked higher than the others. The most popular restaurants, pubs, stores and so on will receive highest rank. If no popularity rating is accessible, then the closest POI to the search point receives highest ranking. Most of the other search objects are ranked such that the most recent object gets the highest ranking and the oldest the lowest ranking. This ranking applies to tweets, and apartment ads, and could be applied to many others. When using Twitter as a data source results older then a week might not be that interesting to the user because tweets are usually about what is happening right there and then. Looking at the most recent Tweets in the area is of greater interest to the user. By ranking apartments by their upload date the users will get to see all recently uploaded apartments first because they might be of more interest than the ones that have been up for a longer period of time.

**Ranking photos**

Photos from Flickr will be ranked by a combination of how many views a photo has combined with how long it has been since the photo got uploaded. So, in order to rank a picture its number of views will be divided by how long the picture has been up there. If for example, a picture A has say 52 views and have been posted 4 days ago and another picture B has 45 views and been posted 1 day ago, picture B will be ranked higher because it has more views/time unit and is thus more popular from a user perspective. (Picture A will have 52 views / 4 days = 13 views/day and Picture B will have 45 views / 1 day = 45 views/day). This is so that a newly taken picture still can get as high ranking as a picture taken 4 years ago have.

**Ranking Wikipedia articles**

Wikipedia articles will not be ranked in any special way. The first received article will receive the highest ranking and the last one will have the lowest ranking. As the Geonames API for retrieving Wikipedia articles did not include any upload date, any views or other entry that could be used for ranking in a smart way, Wikipedia articles will be ranked as they are received.

**Ranking objects in a pile/stack**

How objects within a stack of different objects should be ranked depends on what objects that will be used from the start. When having POIs, tweets, photos and apartment ads it is difficult to compare the different objects with each other. Whether the highest ranked POI is more important than the highest ranked photo is hard to determine and can also depend on what the user prefer. For starters the ranking within a stack will be set to highest ranked POI, tweet, photo, apartment ad, then the second highest ranked POI, tweet, photo, and apartment ad and so on. Inside a stack or pile of objects what to show could be specified by the user. If they for example just want to see apartment ads they set the detailed object view to just show the apartment ads. This could also be handled in the search, setting it to only show apartment ads or whatever objects the user is after.

**8.2.5. Slice map**

The map view area and the current zoom level will be used to calculate the slices into which the map will be divided. The slices will be used to distribute the search results on the map in a more readable and usable way. When dividing the map into smaller areas before placing the objects into the map, we can choose how many objects we want to display in each slice and
thus make it easier both to read the map and see the individual objects. The question is how to divide the map and why one should divide it in that particular way. A couple of different slicing methods will be examined.

**Slicing methods**
When going through the different slicing methods the assumption will be that the centre of an icon will be placed on the latitude and longitude an object is supposed to be at, even though this might not be the case in the application. Also, for easier comparison between different slicing methods only one object will be placed into each individual slice.

**Pie slices, circular belt slicing and combinations of the two**
Dividing the map into pie slices will give us a cake looking map where each slice area has a direction and a certain angle. The individual slices when dividing the map like a pie will be broader the further away from the search point we get. When dividing the map like a pie all areas will be the same size. We choose to only display one item per slice and can still end up in a situation where all objects shown are close to the search starting point; there is a risk that parts of the view area (lower half of the blue rectangle in figure 8) will be unused. If using pie slices we limit the amount of objects that can be shown at the same time. Using more slices we can show more objects at the same time, but the risk of objects overlapping will also increase.

![Figure 8: Pie slices with 16 objects in, one in each slice. The blue area is the map view on the screen of the mobile phone.](image)

**Circular belt slicing**
Dividing the map using circular belt slices will give us belts at different radiuses away from the search point to work with. The area of the individual belts will be $\pi$(outer edge radius)$^2$ - $\pi$(inner edge radius)$^2$ and thus it will be a bit tricky to get the individual belt areas the same size. Having the individual slices the same area is desirable so that the belts can have a somewhat equal distribution of objects in them. The area of a circle is $\pi r^2$ and in order to get each belt the same size as the original inner belt we have to base the area of the belts on the first slice area and express the radius for the second belt it in terms of the radius of the inner belt.
Using circle area and belt area formulas a general formula can be derived:

\[ r_n = n^{\frac{1}{2}} r_1, \quad \text{for } n \geq 1 \]

For example if \( r_1 = 1 \) the 5\(^{th} \) belt will have the radius \( r_5 = 5^{\frac{1}{2}} \) and the 5\(^{th} \) belt will have a thickness of: \( r_5 - r_4 = 5^{\frac{1}{2}} - 4^{\frac{1}{2}} = 0.236068 \). The width of the 5\(^{th} \) belt will be 23.6% of the size of the width of the inner most circular belt.

The thickness of each belt quickly drops below 25% (already at 5 belts) of the original radius when using circular belts with same area sizes. Say we have a search radius of 0.5km. Then the center belt will have a radius around 200m or so and if we then have 5 different belts, the outer belt will be \( 200 \times 0.236068 = 47.2 \) m wide. A very different belt width, but the belts will have the same area. When having more belts this can lead to a scenario where objects in a particular direction from the search point are overlapping because of the circular belt areas they are placed in getting thinner (see corners of the map area in figure 10).

If we instead set the belts to have the same width no matter how far away from the center we are, the outer belts will then potentially contain much more data compared to the inner belts. This makes flipping through the data when only showing one item/slice a bit strange because the inner belt will most often run out of data earlier then the outer ones. Where to place an icon to represent the objects within the slice is a question that is hard to answer when using belt and/or pie slices.

Figure 9: Four belts of the same width with two items overlapping. The outer belt area is much bigger then the inner ones.

Figure 9 looks good as the worst case when only showing one icon per slice area on the map is that two objects are overlapping on one or more places of the map. A downside of using circular belt slices of different areas is that big parts of the map view go unused. In figure 9 four objects are shown on a map where users could easily handle ten times that. But, if we triple the number of belts, each area would become much thinner and the risk of overlapping would therefore increase.
Figure 10: Tripling the belts from figure 2 results in this figure with twelve different slices and objects.

As shown in figure 10, even though we only tripled the amount of belts from that of figure 9 we still have a lot of unused space at the bottom left and top right parts of the map view. And, if we would add more belts the risk of overlap would increase, as the belts would become thinner still. The maximum number of belts one can use is somewhat limited by how big an icon is and if more then a few belts can “fit within” the size of an icon, overlapping can depending on object positions become a problem.

Combining pie slices with circular belt slicing we end up with map areas enclosed by an angle and two radiuses. The slices will have the same properties as circular belt slicing combined with the properties of pie slices in the center area. Belt wise, the slices will get thinner the further away from the search point they are and pie wise, the slices will get narrower the more pie slices the area is divided into. A drawback of using pie combined with circular belt slices is that it will be just as hard as with circular belt slicing to get slices, at different distances from the center, being the same size. Slices the same size are preferable, especially if stacking objects within a slice, since larger slices potentially can contain more objects then the smaller ones and might then give us a skewed distribution of objects when the slices are of different area sizes. Another downside is that slices at different distances from the center will also be of different shapes from one another. This will make it more difficult to place icons within a slice, icons always being of the same shape while the slicing areas will be of different form at different radius.
In figure 11 there is no more than one object in every separate slice and still it is hard to make out single objects because of overlap close to the search point, in the corners of the map and also where objects are close to each other, even though they are in different slices. When combining pie and circular belt slicing many objects can be shown at once with overlapping as one of the risks. In the middle pie slice area the problem of not using the available screen real estate also comes up. Space is somewhat wasted around the middle while other parts of the map view are too crowded. When combining pie and circular belt slicing we get the downsides from both pie and circular belt slicing with the upside of being able to show some more objects at the same time.

Square slices
The easiest and most intuitive way of slicing a map is using slices in the shape of squares. Using this type of map slicing makes it easy to make slices to be the same size. By using the height and width of the map view the map can be sliced into areas so that one icon fits into a slice by some margin. The problem we had with a lot of items close to the search point or at a particular direction from it will not be a big problem if we only place one object in each slice, since the worst case is that one object in the corner of a slice and three objects from the three neighboring slices are overlapping. When the area of each individual slice is bigger then an icon no more then 4 icons can possibly overlap. When using square slices it is easier to fit icons into slices in a more straightforward manner. Compared to using many belts or pie slices one can have a lot of slices visible on the screen of the mobile without having too much overlap or problems like parts of the view area on the screen of the mobile being wasted.
Figure 12: Worst case when using square slices when showing one object per slice, four middle objects are overlapping.

In the figure above 48 objects are within the view area each clickable, except for the four middle ones that could pose somewhat of a problem. When using square slices and just showing one object icon per slice a lot of data can be shown and still be quite readable for the user.

**What slicing method to use?**

When choosing in what way to divide the map, strengths and weaknesses of the individual slicing methods are compared. Primary goals of a mobile map application, particularly in this thesis, is to avoid too much overlapping of objects and to make sure that the map still can be seen while showing objects on the map. Another goal is keeping the slice area size from being a lot bigger then that of a single icon, so we don’t waste a lot of space within a slice when only showing one icon in each of them. Therefore, the slicing method that will be used in this prototype will be square slices. When using square slices a lot of data can be presented without the risk of too much overlapping. Placing the object icons into square slices is also easier then placing them into a circular belt pie slice. If using pie, a combination of pie and circular belts or any other slicing method, placing icons within slices can get complicated because of the slices being different shapes depending on where they are on the map.

**Rank objects in each slice**

Objects will be placed in the slice where their latitude, longitude are and they will be presented based on their ranking. Higher ranked objects will have priority over lower ranked ones following the ranking described earlier.

**8.2.6. Present on map**

When a search result is returned with ranked objects the question of how to present them on the map remains. When there are few objects that are far apart, presenting them on a map is not difficult at all. Problem arises when many objects have the same place of origin or are concentrated to a small area of the map view.
8.2.7. Icons
Points-of-Interest and data types of different sorts will come from many sources and will be represented by different icons on the map. Different types of POIs can have different icons, a landmark and a restaurant could be represented by two different icons. One way to distinguish high from low ranked objects is to show the icons of the lower ranked objects in a lower opacity than icons of the higher ranked objects. But, because only one icon will be shown in each map slice area, there is no need for lower/higher opacity icons.

Icons on a square sliced map are best being constructed in the form of a circle or some shape that doesn’t quite fill the whole square slice area. This is so the map behind the icon can be seen and not all covered behind the icons when there are many objects to show in one map view. When pointing to exact locations a pinpoint type of icon could be used to visually point at the position of the object.

8.2.8. Many objects in one slice
When a map slice has two or more objects to be placed in it there are a couple of ways to handle this. One way of presenting two or more objects in the same slice is to place the objects at their specific latitude, longitude within the slice and hope that they do not overlap too much. As we do not want to rely on luck when presenting our results, we use another option then just hoping that items do not overlap. Another way is to show the one object with the highest ranking in that slice, putting the lower ranked objects on “next search page”. The next search page would then show the second ranked objects in each slice, where there is one and so on.

Yet another way of showing many objects in one slice at the same time is to stack objects that are in the same slice on top of each other. The highest ranked one will be on the top and as one browses through the stack of objects the ranking decreases. The lowest ranked object will be the one shown last. As explained later this application will stack objects on top of each other when there are many to be shown in the same map slice.

Figure 13: The way Google Maps show that there are many pictures on this specific location - an icon looking like a pile of photos.

8.2.9. When objects are stacked on the same point
When objects are stacked on one particular point on the map they will be represented by a single icon that has object pages within itself. The objects will be shown on the map as one clickable icon with a “to the power of” number at the top right corner of it to highlight that it can be flipped through. The number reflects the number of objects included in the stack. If 5 objects are on the same spot it might look something like “an icon”^5. “An icon” should be replaced with some appropriate icon that represents the stacked objects in an intuitive way.
The objects in the stack will be sorted with the one with highest rank showing as the first object when flipping through them. Clicking the stack will open a text box with information about the first object and the user can then flip through the objects in the stack by swiping to the next object. When opening one stack of objects the map view will change so that the stack of objects (that the user wants to explore) is placed at the top of the map to allow more information about the objects to be shown under it. If the user wants to go back to see the map again the back button is always available, and when clicked the user will be taken back to the previous view.

Figure 14: Example of icons, first one with 3 different types of objects stacked. The following three each with 2 of the same sort and the last icon represents one object.

This stacking of objects can also be used in other situations than that when objects are situated at the same latitude, longitude. For example if two or more objects are close to one another, so that it is difficult to click on and open either of them, they could be stacked. The drawback with this is that at least one of the objects will appear to be situated somewhere else than where it actually is. This could be communicated to the user by a link to a map view with the exact location of the object where only the one particular object is shown on the map along with the position of the user. No matter what slicing method is used, when presenting the objects on the map, problems like object icons overlapping with other object icons can always come up. It might then be hard for the user to select the lower ranked one of those objects. This is especially the case when no more zooming can be done. Therefore, in this solution, objects that are situated within a slice area will be gathered under one icon to make the map more readable and each object more clickable (the left drawing in the figure below). Instead of 10 icons, the map (right part of the figure) will show 5 icons where 3 of them contain more than one original object and can be flipped through. The user will still be able to see that there are 10 objects on the map to the left because each icon will have a “to the power” of number that indicates how many objects the individual icon really contains. We eliminate overlapping and obtain a more visible map at the price of moving objects a relatively small distance in the map view.

Figure 15: Example of stacked and not stacked objects in a combination of pie and circular belt division of the search area.
8.2.10. Displacement of objects

When stacking objects in a slice they will be moved from their real latitude, longitude position by some distance. When we zoom out the distance will be amplified and when zooming in this distance will be decreased. To avoid objects like a store being placed on the map in the middle of a lake or something similar, objects could be set to not be visible until we are at a certain zoom level. The furthest an object will be moved from its original position will be if it is situated in one of the four corners of the square map slice. If a square slice has the side of 20 m, the furthest an object will be moved will then be the hypotenuse (or diagonal) of the square area divided by two. This is when an object is in the corner of one square slice and moved to the center of the slice. In the case of a 20 m square side the half of the hypotenuse is $\frac{1}{2} \times (20^2 + 20^2)^{\frac{1}{2}} = 14.14$ m. When more zoomed out the slice sides are bigger and the distance an object will be moved increases, but is still the same relative to the side of the square.

A circle shaped icon in a square slice will have a smaller diameter than the square slice side. We need some space between different icons to show the “power to” numbers to show the number of objects an icon represents. Some space is also needed if the map in the square slice is to be somewhat visible. The circle formed icons are here assumed to be 75% of the size of one square. This makes the radius of an icon to 7.5 m (= half of 20 m * 0.75), reducing the visual displacement of an object in a square slice with the side 20 m to (½ hypotenuse – 7.5 = 6.64) 6.64 m or 33.2 % of the original square slice side. A worst case displacement of 33% of the side of a square (when assuming an icon diameter of ¾ of the side of a square slice) slice is not that much regardless of zoom level and if an object appears to be somewhere it should not be, zooming in will reduce this displacement significantly.

![Figure 16: Estimated worst case object displacement in a square slice.](image)

8.2.11. Search area shown on screen

Fitting the search area to the mobile map view area would not be the best choice because we want to use as much as we can of the screen to show results on. Instead having the search area set to being one or two square slices bigger than the mobile screen view will make use of the whole screen. Some objects from the search area will end up outside the screen, but this can be solved by having pointers on the screen that shows where objects outside the screen are.
In the right part of the figure above just a small part of the screen, the top and bottom corners of the map view (blue), are outside the search area and still shown on the map view. But it is not that bad, considering that the corners are a bit troublesome to click on.

Figure 18: An example of square slices with some icons projected upon a screenshot of a Google Map search of the nation pubs in Uppsala.

In figure 18 the only real drawback of stacking objects in the centre of a single slice is apparent. One pub, which is located in the north east corner of the square slice, seem in the application to be located on the “wrong” side of the river (second row, fifth column slice). This displacement will be reduced if the user zooms in or uses the “Show on map” feature to see only the one object in the map view.

## 8.2.12. Pointers to off-screen items

Pointers to objects that are outside the view area, as in the figure above, will always be situated in between two square slices. This is so the two slices can keep their own space for showing objects at the same time as objects outside the map can be pointed to. We use a single pointer on each side of the view area, with a number on saying how many objects are situated in that off-screen direction. When using a search area, such as the one in figure 15,
we only have to point to items to the west and the east. If the search area were bigger, then the view area would also include arrows identifying items to the north and south of the current display area.

Figure 19: Left part of the figure explains how the pointers to items outside the view area work. Right part of the figure is how the view will appear on the screen of the mobile phone.

8.2.13. **What happens to the map view when zooming?**

Searching is done at a particular zoom level and the map slicing is based on that zoom level. If the user, after a search - when the objects have been placed on the map - zooms in or out, the map will be re-sliced in order to fit in the new zoom level and the objects in the search area will be redistributed in the new map slicing areas. The icons will always be kept the same size so they always are visible and clickable. Icons will not shrink when zooming out or get bigger when zooming in.

If we keep the slicing when zooming out, icons located next to each other will tend to overlap because the slices become smaller compared to the mobile screen map view. When zooming in we get very big slices and a lot of screen real estate is wasted. The screen real estate is instead sliced irrespective of the zoom level in order to maintain a constant icon density. This will make objects stack more when zooming out and stack less when zooming in. When zooming in, objects from one square slice will be divided on the new slicing square areas and when zooming out, objects will stack up more in the new slice areas that cover a bigger area.

Figure 20: Redistribution of objects when zooming in, icons will be kept the same size and stacks of items will often be split.
At big zoom levels or where there is high icon density, object stacks can get very big, containing many items. Therefore, items won’t be shown until a certain zoom level is reached. If a user searches at too high a zoom level they will be prompted to zoom in, if they want to see any items on the map.

8.3. Differences from the original Ericsson Invention

This thesis was originally based on an Ericsson Invention on how to present search results on maps in an improved way. This invention also identified the problem of items stacking, overlapping and the difficulties for the user to make out parts of the map and or items placed on the map close to each other. Originally the idea in the invention was to extract objects, slice the map, rank the objects and place the ranked objects on the map with some presentation rules. The presentation rules included approaches to limit the number of objects shown on the map in some way, like having search result pages with lower ranked objects on the later search result pages. To have icons change depending on their rank, high rank would mean a bigger icon and lower ranked icons would be smaller and a bit transparent to visually show their lower ranking. Different distribution functions, for instance one distribution function is to evenly show a maximum number of results in each slice. Another distribution function which showed a minimum number of objects in each slice, and then the objects left to distribute with highest rank, was also suggested in the original solution.

The concept developed in this thesis slices the map, and the objects are ranked and placed on the map, but the presentation has changed a lot from the original invention. Instead of limiting the amount of data shown to the user at once, the data is now sorted in a way that allows the user to be able to handle high icon density. The ranking is used in the way that the highest ranked object in every slice will be shown first in the “object-view”, where objects represented by one clicked icon can be flipped through. This simplifies map navigation as well as supporting an intuitive approach to flipping through a “stack of objects”.
9. Results

9.1. User Centered System Design

In order to be as user centered as possible the key principles for a user centered system design process were followed as much as possible.

- **User focus** – user tasks, needs and goals of the users’ guided the development. As the potential user group is quite wide, people with different background, age and interests were used in the testing.

- **Active user involvement** – users were involved early on in the development of the concept and paper prototype and the users involved was representative of a real user group.

- **Evolutionary systems development** – two user-tests were conducted to make the development and testing more iterative. The different parts of the application were developed in increments that could be evaluated and tested before being made permanent.

- **Simple design representations** – the concept of the mobile phone application was represented by paper prototypes that were easily understood by test-users.

- **Prototyping** – the paper prototype was developed in an early stage of the concept and was tested with potential real users in a mobile phone like context.

- **Evaluate use in context** – in the testing seemingly real scenarios were used to make the testing as real as possible. Later on when the concept was implemented, the implementation was tested on new potential real users and their behavior and ideas were captured and analyzed.

- **Explicit and conscious design activities** – design activities regarding parts like icons and buttons were conducted in the development process. A re-work of the icons were done after both of the two tests.

- **A professional attitude** – the development process should be performed by effective multidisciplinary teams. The author of this thesis was mostly working alone but with two supervisors supporting him when needed.

- **Usability champion** – One of the supervisors of the author of this thesis was somewhat of a usability expert, but as he had other work assignments then to aid a thesis worker he could not constantly be involved. However, ideas about the bigger issues were discussed and evaluated together with the two supervisors.

- **Holistic design** – Through working alone not too much was developed in parallel, although the development of the two user-tests was overlapping with the concept and implementation work.

- **Process customization** – the user centered system design process was followed as much as possible using the resources at hand.

- **A user-centered attitude** – was established and maintained through-out the project.
9.2. End-user testing

9.2.1. Preparation of the paper prototype usability test
Before conducting the real usability test, a “test of the test” was done. This pre-test testing was done on a person without much experience of touch screen mobile phones and mobile phone technology in general. This test was conducted to see whether the made up scenario made sense and was good enough to test the application concept. After some hesitation and a couple of questions the test subject got busy navigating around in the paper prototype. It became clear that some of the icons used needed more explanation and had to be reworked. Some terms, especially the term POI (point-of-interest), confused the test person and were rephrased in the paper prototype.

9.2.2. Testing the concept – first test
The first test, conducted on a paper prototype based on the concept, was based on scenarios where the users were supposed to solve two different problems by using the application. The first one (scenario A) was searching for apartments in an area and to find a particular one. The “right” apartment was situated off-screen so the users had to realize that there were objects outside the view area and search wider in order to solve the problem.

In scenario B, the second part of the paper prototype testing, the users were supposed to find a particular store located a bit to the east from a named place on the map. This scenario was complicated by the fact that there were a lot of objects showing on the map, partially covering the place on the map they were supposed to find. The idea of scenario B was to test how users would react to zooming while searching, and also to try out the clear map function that could be used to better see the map when parts of it were obscured by search objects.

9.2.3. Testing the prototype – second test
The second test was done after the programming of the prototype application and was carried out using a real Android mobile phone. This test was of a more the observing kind then the first test. The test users were first to familiarize themselves with the application for 5-10 minutes by testing it on their own. During this part the test users were monitored and notes were taken on how they reacted to different parts of the program and whether they had any trouble while testing. Afterwards the users were asked to complete some tasks by using the application and searching for information. The tasks were designed in such a way that they would get the users to test most parts of the application.

Both the test on the paper prototype and the test on the prototype application were concluded by letting the test users answer a couple of questions regarding, for example, what they conceived to be troublesome, what worked well and what they thought needed improvement regarding the application and design. They were also encouraged to share any suggestions they might have on how the application could be improved. This way a lot of different suggestions, hopefully some good suggestions would be obtained at the same time as testing the current solutions and design.

9.3. Results from the usability test on the paper prototype
The first usability test was based on two scenarios where the users were supposed to solve two different problems by using the application. In scenario A the assignment was to search
for apartments ads to find a particular apartment. The apartment was off-screen so the users had to realize that there are objects outside the view area and search there to solve the problem. In scenario A, one out of five test users used the zoom function to zoom to be able to see all the objects at once. The rest of the test users flipped through every stack of objects they could find on the map until they found the apartment ad that they were supposed to find. One out of five test-users used the search function to search for apartment ads while the rest did not touch the search function at all during scenario A. In general, users seemed unaware that there actually was a menu that was clickable below the mobile screen. Once the users dared to “click” on anything on the paper prototype, it seemed that they quickly got the hang of how the objects in the different stacks could be reached.

In scenario B the users were to find a particular store to the east of a place on the map, while there were a lot of objects showing on the map, partially covering the place on the map they were supposed to find and base their search from. The idea was to force the users to either zoom in or to clear the map and look for the place they were supposed to start from. What made this a little tougher was that on the printout the text of streets and other features on the map was really small, too small to be readable for one of the testers. This small, hard to read text was unintentional and may have made this scenario more difficult that it should have been. To be able to see the map in scenario B many used the clear map function. When they afterwards searched for objects they focused their search on what they were looking for, so when the users got to the search function they knew what to do in order to find what they were looking for.

User feedback
Two of the test users said that the search function in the program was good, because through it you can reduce the total number of objects shown. One user forgot that a search function even existed when looking for the apartment in Scenario A (in scenario B the test users were reminded of the search and zoom in the scenario description because that’s what the B scenario was supposed to test). One user said that it was good that one could see the map even though showing really many objects. One user wanted to be able to search for roads and such as well, but specified text search and such is not really the point of the concept or prototype application. Another one commented on the small text in the B scenario, that scenario might have affected the test users a bit negatively because of the small text. The test otherwise showed that the interaction with the program was to satisfaction for all the test users. Most of the test users thought that the feedback received was quite enough but one of the users said that it could be good with some more information during usage.

When it comes to the icons the users understood the meaning of the icons once they knew what Twitter, Flickr, Booli was, everyone knew what Wikipedia was. The icon that some users had trouble of understanding was the icon for restaurants, shops, landmarks and other points-of-interest, “P” and it should be replaced with something else that better reflects what the icon represents. POI or point-of-interest is not a commonly known term, so that is something to keep in mind. One user said that she sometimes forgot what the P icon meant and suggested that maybe some more symbolic picture could be used. One other user commented on the green arrow pointing on off-screen items, and said that it should be improved so the user knows what exists in that direction. This user also said that the mix-icon should reflect what items it contains, so if it only contains Flickr photos and tweets it should only have those two small icons within itself.
9.4. The prototype application

One of the results of this thesis is the Android mobile phone prototype application based on the concept description presented earlier. In the program the user can search for tweets, photos (from Flickr), housing ads and Wikipedia articles in the area currently viewed on the map. When the search is done the map is sliced into square slices with size based on the height and width of the view area on the mobile phone. Zooming will change this slicing and the distribution of search items. Search items in the different slices of the map are represented by one icon per square slice and these icons can be clicked to view all objects within that slice.

![Figure 21: The search window (left), when pressing search (middle) and the returned results (right).](image)

Objects are viewed in a form of gallery where they can be scrolled trough and clicked on to see their respective data below the gallery. Objects that are currently outside the view area of the map are pointed at by green pointers located at the top, bottom, left and right parts of the screen.
9.5. Results from the user test of the actual prototype

The second usability test was conducted on the prototype application using the phone from the pictures above. The test users got to try out the application prototype on their own for a couple of minutes without any help or guidance unless they had any questions, for example not knowing what Twitter and Flickr is, or other questions not directly concerning how to use the application.

Observations from the test
One user started with searching for points-of-interest in the Stockholm area and when he wanted to go back on the map to where he currently was located he started scrolling upwards and then asked if there was any way to see your current location on the map. There is a button in the menu called Locate Me. Many persons testing the application had something to say about the map and might have had more focus on the map then on the searching and the objects on the map. Not all of the test users took time to read all the information on the start up screen when starting the program. The ones who did read everything barley needed the “assignments” of the test as they were trying out almost everything on their own without any real help.

Generally
Apart from the map the test users thought that the usage of the program was easy and natural. In answering the questions one user said that the concept could have interesting areas of usage but that he was missing some relevant objects like hotels, restaurants and such, for when traveling. Good things about the program said one of the people trying the application is that it is easy to understand, the information is clear and it is easy to search and clear. The same person also complained on the map that it for example should show a difference between nature and densely populated areas.
**The icons on the map**
Generally the people who tested the program thought that the icons were good, and most could see and understand how many objects every icon represented. The meaning of the green arrows was also understandable for most of the testers. One test user had some trouble clicking the icons on the map.

**The object view**
The object view where the objects within a icon stack can be viewed one a time was considered rather logical, and easy to navigate. One user said that this view was a good way of flipping through the objects from an icon.

**Regarding feedback during usage**
On the question addressing if the user understands how the program can be used, one user said that the feedback and information received is enough, but that some additional information to get the user started, could be good. When clicking an icon on the map all test users said that they got enough feedback and one user commented that it was good that the icons changed color when clicked so you could see which one you hit.

**Other**
Most of the test users were unaware that the search was for an area on the map and thought it was for the whole map at once. One user suggested that the search area should be indicated in some way, either by graying out the area outside the search area, marking the search area with a square, circle or something. How to go back from the different views, like for example when viewing a picture or a Wikipedia page in full screen and wanting to go back to the previous view, some of the test users had to ask how they can go back and one of them said that, here I would press the “back button” on the phone, if I knew where it was.

**9.6. Implications of the results**
The results from the testing of the paper prototype were used to improve the concept design before implementation. The changes made were of a design character, like changing icons and clarifying things here and there. The object view was also changed between the original concept and the final application design. Instead of showing a bit of map at the top of the view, the object view was made full screen, but transparent so that the map behind it could be somewhat seen. This way the object view had more space for showing information about the objects at the same time as one could still see what icon had been clicked. After the results of testing of the prototype some changes were made. As in the case of the paper prototype, this was mostly design changes, like moving buttons, changing icons and so forth.
10. Implementation

Before analyzing the results obtained from the testing, some details about the implementation will here briefly be covered. The prototype application uses the Ericsson Mobile Maps API for the map, Twitter, Flickr, Booli and Wikipedia for geographically tagged data. Searching is done in a square area around the center of the current map view. The respective data source APIs are called with the center position of the map and a radius based on the height of the map view. The returned answers are of XML format, which get parsed and the respective data is extracted and associated with an icon and placed on the map. All objects within the search area are saved in a special designed list (ObjectsInArea, picture below). This list contains objects called ObjectsInSquare, which contains a middle longitude and middle latitude position of the objects that are located in that square. The ObjectsInSquare also contains a list of objects that has the same middle longitude and latitude position.

ObjectsInArea List<ObjectsInSquare>  ObjectsInSquare  ...

Object ObjectsInSquare (midLongitude, midLatitude, List <MapObject>)

Object MapObject(longitude, latitude, type, data, imgLink, rank)

When a new object is added to the Objects in the search area its middle square position (the middle point of the object’s square of origin) is calculated and is searched for within all the ObjectsInSquare objects. If the middle position exists, the new MapObject is added to the appropriate ObjectsInSquare list containing one or more MapObjects. If the middle position of the new object does not exist a new ObjectsInSquare with the new middle position and a list containing the new MapObject is created.

When an object is added to its ObjectsInSquare (list containing MapObjects) it is sorted by rank so that the highest ranked object is placed first in the list. Icons are drawn on the map at their middle positions taken from the list of ObjectsInSquares and the icon appearance is determined by the contents of that specific node.

When an icon on the map is clicked all object data that icon represents is sent to the Object View method and the object with highest ranking among them will be shown first to the user while the rest of the objects are in a gallery scroll list above the data view. How an icon on the map looks depends on what it represents, if its only one type of data, for example Flickr photos a Flickr icon will be used. But if the icon represents, say two Flickr photos, one Twitter post, and three apartment ads a mix icon will be used to indicate that that square contains different types of data. A number indicating how many objects one icon represents is below every single icon. The application also has some additional functionality like a Locate Me function that gets the user position provided that the GPS of the phone of the user is enabled. Another function is in every object’s own data view there is a link to view that specific object alone on the map at its exact position instead of in the center of its map slice.
11. Analysis

Here follows an analysis of the results obtained in the earlier sections of this thesis. Topics that will be covered are user centered system design, usability testing on the paper prototype and on the prototype application, differences between concept and application and a brief discussion of the problems that emerged during implementation.

11.1. User centered system design

In this thesis some of the key principles of user centered system design are more relevant than others of the simple reason that the author is not working in a team, has little access to a usability expert and has no development team other than his supervisors.

- **User focus** – By putting the needs of the user foremost, the work was steered in a more user centered direction than it otherwise would have been.

- **Active user involvement** – Maybe not involved from the first day, but early on in the development of the concept, the user involvement provided a lot of feedback that could not been found elsewhere. Because of the multitude of the users involved many different views could be captured and be used in order to create a better design.

- **Evolutionary systems development** – Through multiple user-testing and re-design activities the development process has been made more iterative. Results that came up during the two tests had to be taken into account in the concept and implementation work. Without the testing performed some things would never have been learned and this definitely improved the resulting concept and prototype application.

- **Simple design representations** – Using a paper prototype has provided an easy way of testing the concept towards end users while developing it. Instead of having the users reading and commenting on the concept description, the paper prototype has helped to showcase and test different ideas from the concept in a way that the users could easily understand.

- **Prototyping** – Using a paper prototype of the size of a mobile phone screen definitely facilitated testing the concept without writing any code at all. The pros of using paper prototypes where apparent, it was easy to change the prototype on the fly and the users testing the paper prototype felt more comfortable criticizing it then the ones testing the real application on the Android mobile phone did. Due to paper prototyping improvements could be made early and not as much time was later needed when implementing and testing the real application.

- **Evaluate use in context** – The scenarios used when testing helped the test-users to see what they could use the application for. If no scenario existed when testing the paper prototype performing a task would seem much more random then it now did. When testing a paper prototype it is easy for the user to feel lost and not knowing what to do, but when having a scenario in the back of their mind, solving a task seem more real then the task alone would be. The set of new test-users trying the real application had an easier time than the ones testing the paper prototype, since searching for points-of-interest where they live, for example apartment ads in their neighborhood, is much more real then the data in a paper prototype ever can be.
11.2. Usability testing using the paper prototype

In scenario A of the paper prototype, where the object the users were supposed to find was outside the current map view, only one test-user used the zoom to see all objects on the map. This being the first part of the test might have had an impact on how the task was solved as the test users in the beginning of a test tend to be a bit more careful. The one that zoomed out instead of scrolling on the paper prototype map read the introduction text more carefully then the other test users and therefore had a better idea of how to locate objects outside the current map view, what the icons meant and such. When using a paper prototype things like scrolling and zooming become a bit awkward as the two functions do not happen smoothly and quickly as they would in a real application. When scrolling on the paper prototype one cannot quite be sure what is happening because it is not as interactive, as in the case when using a real mobile phone. Only one out of five used the search function in scenario A to limit the number of objects, this was influenced by the next result from the testing, that the test-users generally seemed unaware that there was a menu containing functions below the paper prototype mobile like screen. Being unaware that there was a menu where the search function could be found made it difficult to use the search function located in this menu. Again, users who had read the introduction text thoroughly did not have this problem. Navigating in the paper prototype seemed to be easy enough, and the test users quickly got the hang of how to view the different objects on the map.

Scenario B was to test the search and zoom a bit more then the other functions. The task was to find a certain place on the map, search around that spot and find a store located close to it. The unintentionally too small text on the map made this scenario harder then it should have been but most test-users cleared the map (that from the start of the scenario had quite a lot of objects on it) in order to easier find the spot to search from. Some had trouble finding the search function, but when it was found they knew what to do and how to find what they were supposed to find. As the test-users only got information at the beginning of the test by reading the introduction in the first part of the paper prototype, the ones that did read the intro text had no real trouble finding the search functions, but the ones that did not read the intro text had to click about until they eventually clicked on the menu and found the search.

While testing, one of the users said that it is good that the map is visible, even though there are many objects shown on the map at the same time. As this is one of the original points of the thesis, and the design of the application itself, it is not too strange that someone commented on it. In scenario B some users commented on the small text on the map, this might have affected the scenario negatively when it is not the text size that should be the focus, it’s the searching, scrolling and zooming.

11.3. Differences between concept and the prototype application

The resulting prototype application largely followed the original concept. Most differences came in design of the various parts of the application. Icons, the object view and the size of the search area are parts that were changed a bit. Icons got changed after user-testing, mostly after the paper prototype testing. From the start one data type that should have been searchable was regular points-of-interests such as restaurants, stores and such but no appropriate source of data was found to use for this. Open Street Maps were used at first, but because of excessive file size and too much irrelevant data was excluded from the prototype application. This will be covered more in the section 10.2.1 Implementation problems.
Weather was in the concept as something that could be added to every object position on the map, but as stated in the concept description weather is area and not location specific and therefore was finally not included.

**Object view differences**
The object view, where the user ends up when clicking an object on the map, differs from the original concept design. It ended up being a semi transparent window that popped up when an icon got clicked, as opposed to the concept design where part of the map (where the clicked icon was positioned) was visible at the top of the object view. Through the object view being transparent the users can still see the icon they clicked as it changes color and the map is visible through the object view window.

**Search differences**
In the concept the search area was set to just be a little outside the current map view. In the prototype application the search area was set to be a bit bigger (ten times the screen view width/6 and ten times the screen view height/8) in each direction thus including more search results and showing more items on every search conducted. This bigger search area also resulted in pointing at objects outside the current map view had to be done upwards and downwards as well as to the right and left sides. The concept description has a short section about query searching in it, but query searching is something that falls outside the scope of this thesis. Even if searching for something explicitly is something that users of this type of application would like to be able to do, query searching will not be included because it complicates the programming part of the thesis and would take longer time to implement.

**11.3.1. Implementation problems**
There were some minor troubles during the implementation phase of the thesis and these will briefly be covered here.

**Problems with data types**
Starting on the data type choices, specifically with Open Street Maps and retrieving points-of-interests using their export to XML function. The xml files, even for very small areas got quite big because of the extensive coverage of items on the map (examples of those items are fire hydrants, bike parking’s, traffic lights, cycle ways, and highways and so on). With the intention of getting data about restaurants, stores and such while getting this volume of irrelevant data from the search, made using Open Street Maps a not so good idea. When searching in a quite small area in New York City using Open Street Maps, over 2000 items were returned even with some filtering when parsing the xml. Every icon object stack on the map contained 30 or more items with 75%, sometimes more, of them being “trash”. The positive aspect of this problem was that the map was still visible, even though many objects were shown on the map. One more data type that resulted in a bit of trouble was Twitter. When getting longitude and latitude from one Twitter a special solution had to be implemented because the Geo point xml tag, which was supposed to contain the location data, did not always do so. A workaround was created using the longitude and latitude stored when users posted from an Iphone or Uber-twitter (a program used to tweet) which both most often sent location information in their posts.

**Problems with ranking**
Another problem regarding the data types was how to rank them within themselves. Most got ranked by their age, the newest first and oldest one last. Ranking Flickr photos by views/time
unit proved to be difficult because of difficulties on how to represent time in this division. Time ended up being represented by milliseconds and therefore got very large. When dividing the number of views by a very large number the quotient tend to go towards zero no matter the number of views. Most pictures from Flickr had a number of views raging from 0 to 10 (most had 0) and as one day is equal to 86 400 000 milliseconds, the ranking will often be very close to 0. The fact that most of the Flickr photos number of views was zero, this ranking got kind of obsolete. Because of this the Flickr photos instead got ranked, just like the other data types, with the latest or newest object first in a descending order.

11.4. Usability test on the android prototype application

In this second usability test using the prototype application many of the test-users had something to say about the map and the focus seemed generally to be more on the map then on the searching, the objects and different views in the application. This can be an indication that searching, objects and the different views work well, but it is hard to tell when there is something else (the map) that the test-users are focusing most on.

Reading provided information
One of the test-users did not realize that there was a function to center the map on your current location, which is a bit strange because when starting the application a lot of information about how to search and so on is displayed. In order to search for anything, the menu has to be opened and once there most people would read all choices as there only are 6 of them. While testing one of the users commented that some more information to get the user started would have been good, that being said, this test-user did not read the introduction screens, so the comment does not have that much weight into it. The people that read the introduction texts carefully had no problems using the program, which indicates that the information given at start up is enough to get a user going.

Missing relevant objects?
The comment received about missing relevant object like hotels, restaurants etc. is kind of expected as the idea from the start was to have those kinds of objects on the map along with the Tweets, Photos and articles. To be able to see a restaurant on the map, then maybe if someone has commented on the restaurant on Twitter or if anyone got anything else to say about it might be interesting to when choosing where to have dinner. But since the data source that was supposed to be used to show hotels and-, restaurants was discarded, no further effort was put into this, as the main aim of the thesis was to investigate how to handle a lot of information and not to show one particular type of data.

Icons and screen sizes
One of the test-users had particular problems with clicking the icons on the map. This can depend on many things, maybe that the person is not used to touch screen mobile phones and when the items to click are small the trouble of using a touch screen is amplified. The screen of the mobile phone used for the testing was 3,2 inches (T-Mobile G1). The biggest screens are up to 4 inches (Sony Ericsson Xperia X10) and 3,7 inches (HTC Desire) while the smallest currently available is the Sony Ericsson Xperia X10 Mini at 2,6 inches. While developing the application the author was fortunate enough to get to test run it on all mobile phones mentioned above. It was harder clicking icons the smaller the screen got. On the Xperia X10 Mini some configuration of the application for a smaller screen size might have been in order, since when using that phone the icons were smaller and thus harder to click,
more about this in the future work chapter of the thesis.

**Search, where?**
One thing that could have been pointed out more clearly in the application was that the search was not for the whole map at once, something most of the test-users seemed unaware of. To indicate or highlight the search area in some way was one of the suggestions one test-user had.

**Views**
The object view (the view where objects that an icon represents are shown) has to be considered to be working quite well, as none of the ones testing the applications had anything negative to say about it. Something that some of the test-users had problems with was going back from one view to the previous. The comment “here I would press the back button on the phone if I knew where it was” says a lot on this matter. Every Android mobile phone (that I have come across) has a back button and an assumption that one can make is that the users know where to find the back button on their own phone. Therefore this problem of finding the back button cannot be attributed to the design of the application.
12. Discussion and Conclusions

The purpose of this thesis was to increase our knowledge about geographically based content services and investigate how improvement of presentation of search results on maps could be accomplished. Maps mean maps in computers or mobile phones and search results mean various geographically tagged data that can be displayed on the map. The problem was how to present a lot of data on a map in a for the user, readable and usable way. The thesis work resulted in a concept description of a mobile application dealing with the issue and also an implementation of this concept for Android mobile phones. The concept and application were tested on a number of potential end users to discover weaknesses and receive feedback. The two tests introduced some iteration in the design and development process which was intended and a part of keeping up a user centered way of working. While performing the testing that I did in this thesis, I found that it is, as the one administering the test, hard to stay silent in some cases. Many things can be learned by observing a user trying to solve a problem. The tests proved to give useful results that could be used in order to improve both the concept and the application. Something that is difficult is to determine when one has had enough users involved in testing a design. In this thesis multiple testing was applied, with redesign and an implementation phase in between all to be as user centered as possible. By having a total of 12 users testing the concept and application, along with evaluations and discussions with supervisors, some usability has been achieved in this thesis. By comparing the tests on the paper prototype and the tests on the programmed application it was clear that users testing the paper prototype were more prone to point out things they thought needed improvement then the users testing the real application was. The test users did not all read the information provided in the application. If read this information could help the user in some situations, like how to get back to their current location on the map. Even if not reading information text in an application a user should be able to use it in a meaningful way. Some things, like where physical menu and back buttons are located on a phone, users have to learn by themselves, but it can also be aided by smart software design and programming.

In the concept description one solution on how to show a lot of objects at once on a limited screen area was developed. The solution involved slicing the map, stacking items from each slice, representing them with one icon describing what that particular stack contains. Icons were placed on the map in the middle of their slice and could be clicked in order to browse through the items that they represented. When clicked an icon changed color and the object view popped up, displaying all items represented by that icon. In the object view the items was presented in a gallery like view in a straightforward manner where the highest ranked item came first. Whether this is the best solution can be discussed, but one important thing that this concept handles is the case when items close to each other are overlapping and being hard to distinguish from one another.

The concept of sorted presentation of objects on a map has a broader application area than has been showcased in this thesis. This way of presenting data combined with for example augmented reality can be a powerful way of showing information. Usability is important, especially in applications used in mobile phones, where screens are small and a lot of information has to be compressed in order to be shown. This concept makes it easier to display a lot data on a map. The concept could be incorporated into an API (application programming interface) or an SDK (software development kit) that developers could use for making their own map applications and through using this concept simplify the presentation of a lot of data. The concept can be used to display any type of geographically tagged content that can be displayed on a map. Something that is difficult when displaying a variety of
different data on a map is to achieve a balance between application usability and map visibility, to get the icons big enough to be easy to click while still not covering too much map area behind them. This turned out quite well using phones with medium or large sized screens, but a bit worse when using phones with small screens, the icons simply got a bit too small. Icons and map slicing in relation to the physical screen size is one of the things that can be a topic of future study.

12.1. Future work

Something that should be added to the prototype application is some source of data for searching for standard points-of-interest like restaurants, hotels and shops. This would make the application more usable then as just a showcase for a concept. In addition to information about a restaurant the user could receive, for example, comments or pictures from other user’s experiences eating at that restaurant. Regarding improvements to the concept, the case where it only exist one object in one single map slice should be dealt with. In the way that it is handled now the object is put in the middle of its square, just like when there are many objects to stack under one icon. This causes the map view with objects on, while being structured, to become a bit too rigid, especially when more zoomed out. One way this could be avoided is to put the single object that is alone in its slice, on its real actual location on the map. This however could lead to overlapping if there are three other single objects in any of the adjacent slices on the same area of the map. Overlapping is in this thesis considered something that should be avoided and therefore what to do with an object that ends up alone in its slice becomes a bit of a new problem to solve.

Ranking is something that could have been used more to dictate what data should be shown first from a collection of objects. While having ideas (that proved not to be working that well in practice) on how to rank for example Flickr photos, ranking is a tool that can be used to sort different search results in a meaningful way. Another thing to do, regarding the application, is to make the icons on the map easier to click as some that tested the application had trouble with that when using mobile phones with smaller screens. Preferably without making the icons bigger so that they would cover more of the map view. This could be done by making an icon’s whole map slice clickable or to have a bigger area clickable then the icon itself. Something else that can be done about this is to have the map slicing take into account the size of the screen of the mobile phone. By for example, having fewer maps slices visible on the map view of the mobile phone at one time, each icon could be made bigger and easier to click. In the prototype application 8*6 map slices are visible on the map view of the mobile phone after a search, this could for phones with smaller screens be reduced to something like 7*5 or 6*4 map slices, depending on screen size.

Something else that can be done is to include a function in the object view (where the objects that one icon represents is presented) to see next icon on the map, maybe a button or some slide to go to next icon. This would help taking the user to the next icon on the map without having to go back to the map view.
13. Literature

13.1. Printed sources


### 13.2. Internet sources


Ericsson Labs Homepage, 2010 April, [https://labs.ericsson.com/apis/mobile-mobile-maps](https://labs.ericsson.com/apis/mobile-maps)

Fire Eagle homepage, 2010 July, [http://fireeagle.yahoo.net/help](http://fireeagle.yahoo.net/help)


OpenStreetMaps homepage, 2010 April, [http://wiki.openstreetmap.org/wiki/Introduktion_p%C3%A5_svenska](http://wiki.openstreetmap.org/wiki/Introduktion_p%C3%A5_svenska)


Twitter homepage, 2010 March, [http://apiwiki.twitter.com/Twitter-API-Documentation](http://apiwiki.twitter.com/Twitter-API-Documentation)
