Geographitti Using Mobile Devices

A dynamic, user-centered mapping service

by
Torbjørn Halvorsen & Harald K. Jansson

A report in the course 'Digital Maps'
At Østfold University College, Norway

May 2005

Abstract

Geographitti is the act of adding metadata with a geo-component to a public map. The Growser-framework is built to enable this, using hand-held, mobile devices. Using a client application, users can manipulate map-data themselves.

The basic dynamic units in the system are Point(s) Of Interest (POI). A Point Of Interest holds interesting data pertaining to a geographical location. Traditionally these point are associated with public information, or data delivered by commercial vendors. By offering the public an easy way to distribute such information, the Growser-framework opens for new types of mapping services.

The framework consists of a server, a mobile client and an administrator-interface. Using his or hers mobile phone, a user can manipulate and share dynamic map data. The server works as a main hub, sorting information, searching and processing data, and keeping track of the users. The application presented in this report, is a proof-of-concept implementation, which features the basic functionality of a geographitti system.

Contents

List of Figures

Chapter 1

Introduction

Our modern society is saturated with information. Starting with the Gutenberg press, and leading up to the digital information-revolution, the act of publishing has become increasingly easy. Thanks to the advent of computer technology, we are now at the stage where ordinary citizens may publish their writings directly onto the internet, for the entire world to read.

Many take this opportunity to provide help and share their expertise with other people. This is liberating, but may lead to misinformation, or information overflow.

Location-based information, however, is still mostly provided by governmental or commercial bodies, such as The Michelin Guide [3], map24 [4], or local tourist bureaus. These services are not controlled by the users, and are often biased. The word-of-mouth method is utilized by most travellers, and is a good supplement to these services. The next logical step would therefore be to offer the public a way to distribute such information to the masses.

As the technology becomes cheaper and more powerful, we have seen a convergence between desktop technology and mobile applications. The mobile phone, which is used on a day-to-day basis by most people, pro-

vides an excellent platform to provide for geo-aware applications.

For such an application to succeed, it needs to be easy to use, quick, and have a large user group. The system needs to cover large information-areas, and have the capability to filter out information. Individual users will most likely have different fields of interest, and commercial offerings often lack the diversity needed to satisfy all. A user-centred system would expand dynamically as the users themselves entered information pertaining to their fields of interest.

This report describes a framework, built for providing such an application to the public. The project relates to the Onemap-initiative [5], hosted by Østfold University College, Norway. This report consists of five chapters. The first chapter is this brief introduction. The second chapter covers background information, useful for understanding the problems faced in this project. The third covers the actual implemented system, and use-cases related to the client-application. The fourth features the results of a usability test, performed on the mobile client. Chapter five concludes the report.

Chapter 2

Background

2.1 Technologies

During our work with the Growser framework, we have encountered, and used various technologies, which we will present in this chapter. These technologies pertain to digital maps, mobile devices, or general computing.

2.1.1 WMS Web Mapping Services

A WMS [6] produces visual representations of geo data. The representation is most often provided in an image format such as GIF, JPG or PNG. The central functions supported by WMS are GetMap and GetCapabilities. The GetCapabilities gives an overview of the capabilities of the WMS server while the GetMap request returns a map representation if it is inside the parameters returned by GetCapabilities. The WMS specification is developed by the Open GIS Consortium.

The WMS specification supports use of layers. The layers contain different properties of a map such as roads, names, buildings and landmass. The layers are often visible only in a range of zoom levels. Requests for

layers not available in the scope of the request will be ignored.

The maps offered by WMS servers are diverse both in information and area covered. Different servers also use different coordinate systems. This means that choosing WMS server will affect both the look and feel of the application and limits the area the application is usable in. There are WMS servers that offer the whole world, but to obtain the best possible details regional servers are favourable.

There are several aspects that must be considered when choosing WMS for an application. Given the uses of this application the most important qualities were found to be detailed information inside towns and cities and the look and feel of the map. The Norwegian map bureau (Statens Kartverk)[7] offers several different maps of Norway including good city maps. And since the source is reliable we decided to use this WMS for the application.

All handling of WMS are done by the server and kept as a separate part of the application. This is mostly done to enable an effortless extension or alteration of the application. Extending the area covered by the application can be done by adding more WMS servers supported. The main challenge with this is to coordinate the different WMS server and the coordinate systems used. Although WMS providers are advised to support WGS84 not all servers supports this.

2.1.2 OpenLS

Open Geospatial Consortium have developed an open standard for location based services called The Open Location Services Initiative (OpenLS Initiative)[8]. The OpenLS Initiative includes among other things standardisation for POI sharing on the web.

Implementing support for OpenLS both server side and client side was considered at the start of the project. The reasons for not implementing support of OpenLS are simple. Implementing OpenLS support in a sat-

isfactory way is time consuming and the time available limited. It was clear on an early stage that the finished application would need major reconstruction and extensions to make a fully usable product. Although implementing a descent OpenLS support is time consuming, it can be done without any major reconstructing of the system.

2.1.3 Positioning

For a geo-enabled application to be feasible, it need to be able to pinpoint it's current position. There are numerous ways of doing this, and some of them will be presented here.

GPS

The Global Positioning System is known by most people. The GPS-receiver works by obtaining a time signal from multiple satellites, and calculate the position using trilateration. One downside with GPS is that it does not work indoors. The satellite signals is only received if there are no obstacles hindering it. Currently, GPS-support in mobile phones is enabled using the bluetooth interface and external GPS-devices. However, some manufacturers plan on integrating GPS-receivers with their phones. This would facilitate for easy development and open for a broader user-group.

Gazetteer-service

A gazetteer-service is a bit like a phonebook, but instead of addresses and phonenumbers, it holds coordinate information pertaining to a particular address or location. For example, if your application uses geo-coordinates to initialize a map, it would be useful to let the user enter his location (i.e. Oslo, Norway) and fetch those coordinates from a gazetteer-service. It is not likely that the user would remember the numeric coordinates, but he/she would usually know the name of their current location.

There are numerous services available to the public. The Growser-project features communication with the LAMA gazetteer-service, which is being developed by Andreas Knudsen, at Østfold College university. It is used when the user wants to initialize the Growser client at a specific location.

Location-ID

Perhaps the most interesting method of obtaining the current position of a mobile phone, is by using the cell-id. All mobile phones must be connected to a radio-tower, in order to communicate. Each of these towers has a unique identifier, and these, in conjunction with a tower-database, can be used to locate the phone. The only real obstacle for this method to be feasible, is that there is no good tower-location database at the moment (apart from those the providers have themselves, which are closed). There are, however, community projects underway to create such databases. The JSR179-API [9] defines methods for getting the current location from the cell-id, but it is not implemented by any manufacturer yet.

2.1.4 Blogging and MobBlogging

Blogging has become a widely adopted way of expressing opinions and feelings to a wide audience. Non professional writers publish their articles to the Internet, and you can say that it is a form of grass-root journalism. However, there are some notable differences between blogging and traditional content-publishing.

First of all there's no need for a publishing body. This eliminates any censorship or manipulation of the article. This is not always a good thing, as publishers often add credibility to a given piece. Second there's no delay from written article to publishing. As soon as the post, or article is written it can be published immediately, and this can sometimes be of big

importance to the writer.

There is, however, one area in which traditional media does a better job than the blogging community; live coverage. The amount of resources needed to produce live coverage of a given event is often a hurdle for grassroot journalists, who publish their articles from their home-pc. Blog-postings relies on a internet-connection and a pc, and either of these are freely available at the users whim.

But what is the difference between ordinary blogging, and content adding, and the mobile way of doing it? Put in few words, mobblogging is blogging using mobile devices, such as cell-phones, PDAs, and the likes. These are devices which most people carry on them during the day, and if an interesting event should occur, the blog could quickly be updated with content. As most newer cell-phones also features cameras, media can be added to the post.

Most bloggers would ramble about themselves, their family, their political opinions and so forth. Theese blogs are normally not read by many people. In the hand of a good writer with a profiled persona, however, moblogging could be a powerful weapon. For example, demonstrations could be covered live, without the bias of the large networks and war-crimes could be captured and published right away. Of course there would be a new breed of paparazzis also. Moblogging could therefore break down the barriers between the content providers and the public.

2.1.5 Location Based Services

Apart from mobligging there are variuos other services which could be realized around a location component. Most often these are information-services, aimed at providing relevant data to the user. Such a service could provide information about restaurants in your vicinity, information pertaining to the traffic-situation in your area, weather services, etc. Most such services would be user-initiated; a user specifickly asks for a certain

type of information. User added content is not necessarily needed in such systems, but later in this report, we will se how the Growser framework facilitates for a hybrid between user-added and publisher-specific information.

2.1.6 KXML

KXML [10] is a small pull parser especially made for mobile devices. It has a small footprint and is designed to be easy and fast to use. In contrast to DOM based parsing, it is not necessary to build the whole xml-tree to use events. KXML is implemented in java, and is ideal for mobile devices and j2me.

2.2 Related work

Even though mapservices on cellular phones is a field in rapid development, there is far between the good examples. A few good projects exists, though, and these will be described briefly here.

2.2.1 Traditional way of handling POI-information

A typical example of delivering POI-information is the tourist map, were the user will be presented with a two-dimensional map of a given area. The map will contain static POI-data such as restaurants, shops, museums, and so on. This way of handling POI-information is good, but the user is not allowed to add their remarks to the current data, or their own POI as they travel through a given area.

Perhaps the most important difference between the traditional and the new way of handling POI-data is that the latter provides for dynamic representation. This way new recommendations could be deployed easily when conditions change.

2.2.2 Tag and Scan

The UK-based ”tag and scan”-application[11] is a modern take on the tourist map. Each user can add, view, and edit location-tags. Users can also browse and search for POI-information within a given radius of their current location. The ”tag and scan” application facilitates for private groups were users can choose who else are allowed to view their tags. This kind of functionality opens up for many new uses of the map. For example when buying a new house or apartment, visited locations could be tagged with images or video. When done you could sit down and look over the tagged POI-information before you decide which to buy.

2.2.3 Urban Tapestries [1]

This project, in addition to the technological challenges, sets out to describe social patterns. By giving users the opportunity to add observations, stories and experiences to a map, the creators hope to explore the way that technology is used, and in which way it influences our social relations. Urban Tapestries seems to be an interesting mix of technology and sociology, with many interesting uses.

2.2.4 Map and Go [2]

Map and go is an Italian geo-service that combines web solutions and mobile clients. The service includes map, traffic information and POI information. According to the web pages all services is available on mobile phones and PDA but little specific information can be found.

2.3 Methods

The Growser project is set out to explore the current possibilities of mobile devices coupled with geo-aware applications. Our main goal was to create a proof-of-concept implementation of a geographitti framework. We have used a methodology called extreme programming to create results in a relative short timespan.

Extreme programming was developed by Kent Beck, who wrote the original book on the subject. The method not very rigid, but nonetheless features twelve guidelines for developing a project[12]. We have followed some of these, but not all. Most important, we have worked in pair, which have made us work faster and more efficiently. This has also led to interesting discussions and generally driven the development onwards.

Chapter 3

The growser framework

The framework consists of three parts; the server, a web-based administrator interface, and a mobile client.

The server does all the background work with rescaling images, fetching maps from a compatible WMS-service, and holding the POI and user-data. It also contains the functionality for filtering and searching the information.

The administrator-interface holds the functionality for making changes to the system. This could be to add more categories, edit or delete POI-data, create or delete users, or changing WMS-settings.

A main part of the framework is the POI-definitions. POI are divided up into categories, and not all POI are user-editable. Examples of static categories are: hospital, library, academic institution, and airport. User-tagged points are added in a general category.

Each POI contains the following information: category, title, description, user ID, timestamp, image, small-image, thumbnail, latitude and longitude. In addition to permission-settings a category works as an ordering element, setting correct POI-icon in the mapbrowser. The icons are much like the ones you would see on a normal map; hotels, restaurants, gas-stations, etc.. The title is the name of the POI. For example L̈eo's Ke-

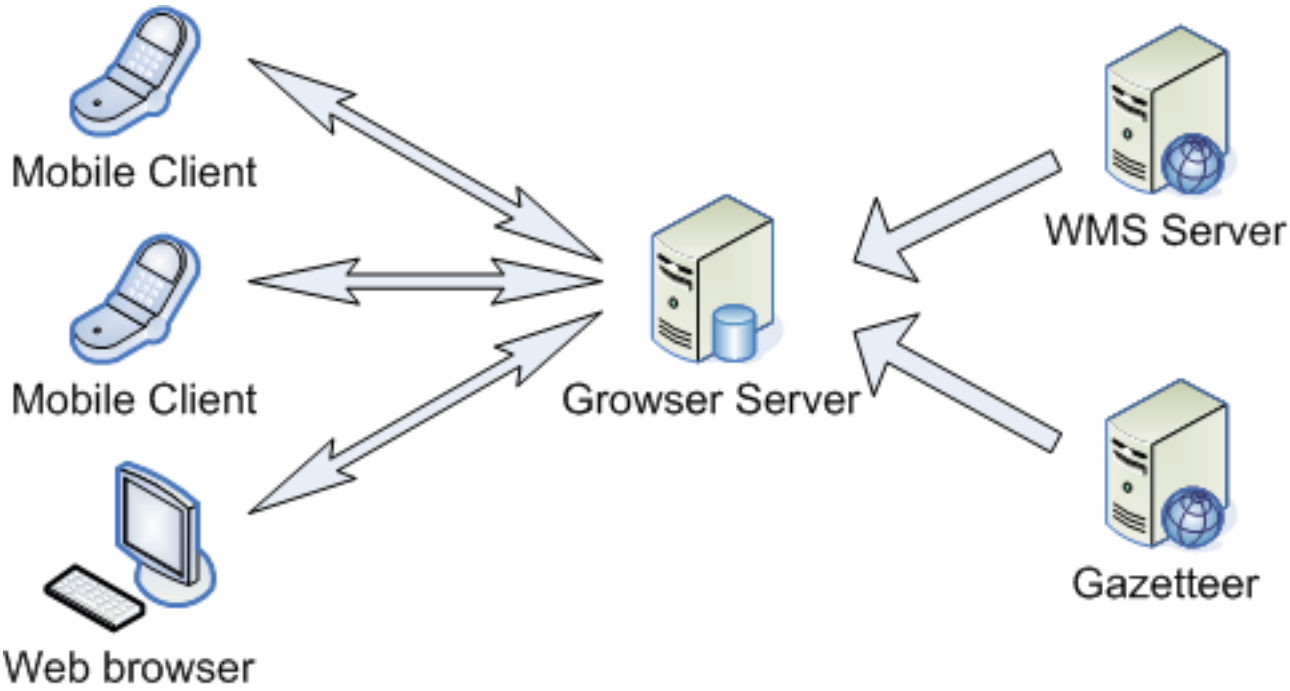


Figure 3.1: Overview of the system

bab¸, ÜIO¸, Ëemo Bar¸, etc. The title is used in the search-function and for point-browsing. A point-of-interest also contains a description which is more detailed information. This can be if the restaurant has good food or other information which could be of interest. The uid is used for tracking-operations, permissions and profile-based functions. It is also necessary to keep track on the time when the POI-information was inserted into the system, and a timestamp is automatically added to all POI-information. This can also be used for points that should be invalidated after a certain amount of time has passed. The images are used when the point is presented to the user, and the lat/ long-values are used for positioning.

3.1 Overview of the system

The Growser system functions as shown on figure 3.1. The only practical deviation from that figure is that the Gazetteer is implemented on the mobile client and not the server. This was done on the client for test purposes and never moved to the server. The gazetteer support on the server was planned implemented along with making the server location aware but never done due to lack of time.

The backbone of the Growser framework is the server. It stores all information and functions as a service provider for the clients. All information gathering and data storage is managed server side. It conceals all data handling from the clients and supplies them with the functionality they need.

The mobile client is the main focus of the Growser system and utilizes the server functionality and supplies the users with a GUI. The mobile client is meant to be the main provider of information in the Growser system.

The location data in the Growser system is represented on graphical maps supplied by a WMS server. The WMS server used in the current version of Growser is supplied by Statens Kartverk (the Norwegian map bureau). This limits the use of the Growser system in Norway. Extending the coverage area of the Growser system can be done simply by using a WMS server with a different coverage area.

3.1.1 Use cases

To illustrate use of the Growser system we'll present some specific use case scenarios. The examples used here is more or less the scenarios used when developing the main functionality of the system.

John is staying at Grand Hotel in Halden and wants to inform other people that this is truly a grand place and the place to stay if you are plan-

ning a trip to Halden. He then starts the Growser application on his mobile phone, finds the location of Grand Hotel, writes a recommendation, takes a picture out of his hotel window and sends the information to the Growser server for all to see.

Paul is planning a visit to Halden and wants to find out where the local restaurants, hotels and interesting tourist attractions are located. He accesses the Growser application on his web browser and searches for Halden. He then gets a list of POI information in Halden and clicks on Grand Hotel. He gets the information on Grand Hotel in Halden along with a street map.

George is new to Halden and wants to find a place to eat. He starts his Growser application and finds his way to Halden. He then turns off all information but restaurants and navigates his way throughout the town till he finds a promising restaurant. He ends up in the trailer selling Chinese food on the town square but that is not really our problem.

Ringo is in Halden late night on Demo bar and sees George not feeling well in a corner. (Due to the previous mentioned Chinese food) He wants to notify Paul and John about this and adds a POI with a picture with his Growser application.

3.1.2 Information flow

There are three different information types in the Growser system, user added information, information added by system administrators, and location information provided by a Gazetteer service. Regular users can add information on geographical location with a mobile client or a web browser. System administrators and power users must use a web browser to add and edit information restricted form normal users.

3.1.3 Information restriction

The user added information is stored on the server associated with the user by a user id. This enables information filtering in future application. This is vital in both in areas of the map with high information density and restricting information access. Those problems represent two of the major problems in a system like this.

Restricting the number of people allowed to view information added by users is tricky. On one hand sharing information with other users is the primary functionality of this system. On the other hand, allowing users to add information not meant for the public in general is vital. Functionality for restricting the information added and filtering the information viewed is not implemented inn the system as of now. The client allows the user to turn of user added information all together but this is no final solution. Restricting information viewed in an area can only be done by turning off desired categories on the mobile client. This is not enough by a long shot. Several extensions to the system are necessary to make it usable for a large number of users. Adding user groups to restricting the number of users that you want information from is the most essential. Restricting information by only getting the most recent additions and grouping information on locations are issues that need consideration before the system can be offered to a large user group.

3.1.4 Technological considerations

The technologies used for the implementation of the Growser system were chosen for rapid development. The mobile client is developed using Java and the server is build width php and a MySql database. This is simply the technologies that we had experience with at the start of the project. This does not mean that those technologies are inadequate for this type of application just that they were never seriously compared to others.

The use of Java as programming language on the mobile phone has proven to be the right one for this application. It offers some serious problems with the different implementations on different phones and some problems with the emulator but that was suspected form the start. It is however, a relative easy language to work with and has supported all the functionality we needed.

The use of php and MySql on the server side opposed to other similar technologies is not done without considerations. Php and MySql run on both the Linux and Windows platform and make them good choices for web server tools. Using other databases supported by php as Oracle, PostgreSQL, Sybase or a Windows ODBC connection can easily be done.

3.2 The Server

The server of the Growser framework serves two main functions, supplying clients width information and providing an administrator tool. All clients communicate width the server using the HTTP protocol. The administrator tool is web based.

3.2.1 Choosing technology

When we started designing the framework two types of servers was considered. A standalone application written in Java and a web based server written in php. Both solutions have their benefits and disadvantages. A standalone server offers much in the way of two way communication and was thought to be the only solution to some of the required functionality. A web server is much simpler to make and offers scalability that have been tested. Upon closer analyse we found that all necessary task could be solved using a web server and the standalone server was abandoned.

3.2.2 Functions

Providing the clients with information needed to display and alter maps is the server's highest priority. For testing purposes two types of clients have been used a mobile client and a web client. It is plausible that other clients could be supported in the future and the foundation for implementing such clients has been made. There is however changes needed in the functionality of the server for full support of several different clients.

The client server communication is discussed in a later chapter and will be only briefly covered here. The current version of the server supports the following:

- user handling
- map retrieval
- POI search
- adding POI information
- retrieving POI information

This covers the most the most basic functionality and can be used as a future framework.

3.2.3 Caching

When you browse the map width the client all map images are stored on the server. The location and level is stored in a database along width the location of the picture. This is then used if the same position is requested by a client.

The basic unit of the coordinate system used is one meter. This resolution is too high to be of any practical use in the application on all but the

highest level of detail. Since this result in a high number of almost identical pictures cashed, all requests will be rounded to an appropriate level of accuracy.

3.2.4 Improvements needed

The need for improvement of the server to support an application used by a high number of clients is paramount. The server has all the needed functionality but lacks many important aspects.

The most important improvements lie in privacy issues. As it is now most of this has been neglected for the benefit of testing. The aspects of privacy in this type of system is to big a subject to handle in this report so it will be neglected here as well as in the program itself.

The chasing of map pictures that is implemented is a simple one. There is no functionality for automatic removal of the pictures and there is no distinction on the different images. This routine was made to speed the downloading of maps to the clients by avoiding the call to a WMS server for all map tiles. It needs serious improvement before the system can be fully operational.

The search function available to the clients obviously needs improvement. It only offers search for text and phrases in the title and describing text of POI. When the number of POI increase the need for more specified searches becomes paramount. As it is now, the search function becomes less useful the more information in the database.

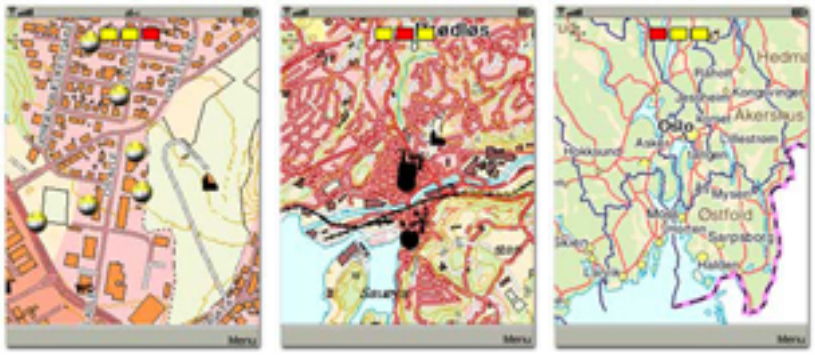
3.3 The Client

As the Growser client is built for mobile devices, it is essential that it is easy to use, and does not require many clicks to perform an action. It currently implements all features of the underlying system, and works as a proof-

of-concept implementation. The features can be divided up into four main categories: browsing, searching, layer-management, and tagging [13]. The client can also talk with gazetteer-services to fetch coordinates for a given location.

3.3.1 Browsing the map

This is the feature that is most used in the client application. The browser have three main features; panning, which lets the user navigate the two dimensional map; zooming (fig. 3.2), lets the user zoom in and out; and POI-view, which displays when the user is hovering over a point.



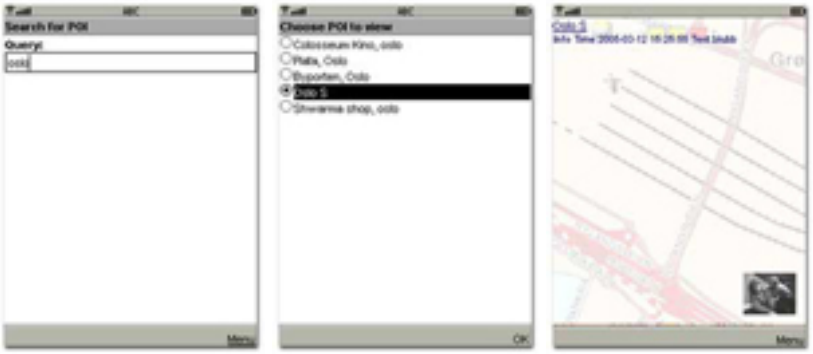
(a) 'local' (b) 'regional' (c) 'national'

Figure 3.2: The Growser client showing the three zoomlevels

3.3.2 Searching for points

POI-searching is done from within the mapbrowser (fig. 3.3). The user enters a query, which is sent to the server, that in turn sends back the relevant data. The client then presents this to the user, who can choose between the

results. When a search-result is chosen, the mabrowser coordinates is set to those of the POI.



(a) search query (b) serch results (c) view choosen

Figure 3.3: The intitial process of searching

3.3.3 Filtering layers

When the mapbrowser gets crowded with POI-information, it is essential to be able to filter out the information which is irrelevant to the user. The Growser client includes an option for enabling/ disabling POI-layers, so that the display doesn't get crowded with unwanted content (fig. 3.4).

3.3.4 Adding tags

Adding tags to the system can be done from the mobile client. The user chooses a point to tag, enters information, attaches an image, and send the information to the server. The point will then come up in the mapbrowsers connected to the system (fig. 3.5).

(a) 'layer' option (b) choosing layers (c) viewing layers

Figure 3.4: Managing POI-layers

(a) point to tag (b) text/ media entry (c) point tagged

Figure 3.5: Adding tags to the system

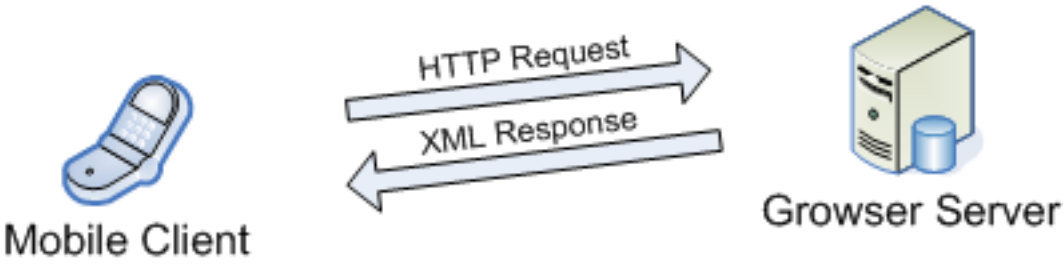


Figure 3.6: Client/ Server Communication

3.4 Client/ server communication

The mobile client is the active part of the Growser framework, all communication is initialized by the client. The client sends a HTTP request to the server and interprets the reply (fig. 3.6). The reply from the server is formatted in XML. Tough this method limits the possibility for client/server communication it is sufficient for all the basic functionality.

3.4.1 Init request

The first request the client will send to the server is an init request. This request contains a user name and password. The response of the init request contains the name of the user, user id, and information on the different categories.

3.4.2 Get map request

The get map request supplies clients width all information needed to create a map width POI information. The get map request requires four parameters, latitude, longitude, user id and level. The level variable is a positive int variable that tells the server the level of detail requested, latitude and longitude variables are at some point given to the client from

the server enabling effortless changing of coordinate system, user id originates form the init request.

When a get map request is made the server will request a map from a WMS service and cache the result. The location of the image is part of the XML reply to the client. All POI information located inside the requested map tile is part of the get map request. A relative position on the map is supplied to position the POI on the map.

3.4.3 Search request

The search request implemented in the framework is a simple word search. It takes two parameters user id and a query string. The result of the query will be any POI containing the string in its title or description. If the exact query is not found in the database it will search for strings contain some of the characters in the string.

The search request returns a list of POIs with the title and a link to the get map request width the POI in the centre of the map. The server sets the maximum numbers of POIs returned. There is need for more search options in the application but it is not implemented.

3.4.4 Tag request

The tag request adds POI information to the database. It supports all the basic information necessary to create a POI in the database including a picture associated with the POI. One major weakness of this request is that it lacks capital to ensure positive user identification from a mobile device.

3.5 Web-Interface

The web interface serves as an administrator tool of the system. It includes editing options for users, categories and POI information (fig. 3.7). The

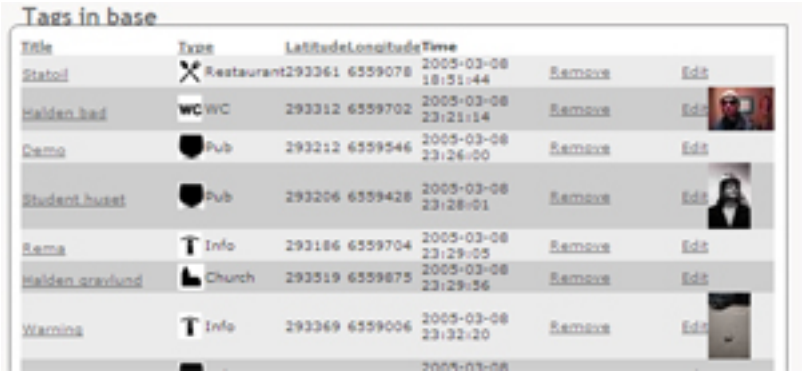


Figure 3.7: Viewing tags in the web-interface

editing options are somewhat limited and not good for a large scale user group but acceptable for test purposes.

3.5.1 POI Editing

The web interface includes functions to view, add and edit POI information (fig. 3.8, fig. 3.9). The add functionality included is made by simple HTML and offers no good ways of navigating maps. In a final application this needs to be changed to a more dynamic interface similar to the mobile client, like an applet. Once the POI is made it is not possible to alter the location.

If the POI contains an image file the server will make two downscaled images for used in mobile applications and as thumbnails. The original image will also be stored for use on web pages.

There are two ways of accessing POI information on the web. You can browse a map with a map browser or search for POI information with a web form. The map browser offers simple navigating including different zoom levels. WWhen a POI is viewed in a browser for the first time, an image will be generated on the server as shown in fig. 3.10. The image is composed of two maps with different zoom levels, and an icon repre-

Figure 3.8: Locate a poi in the web-interface

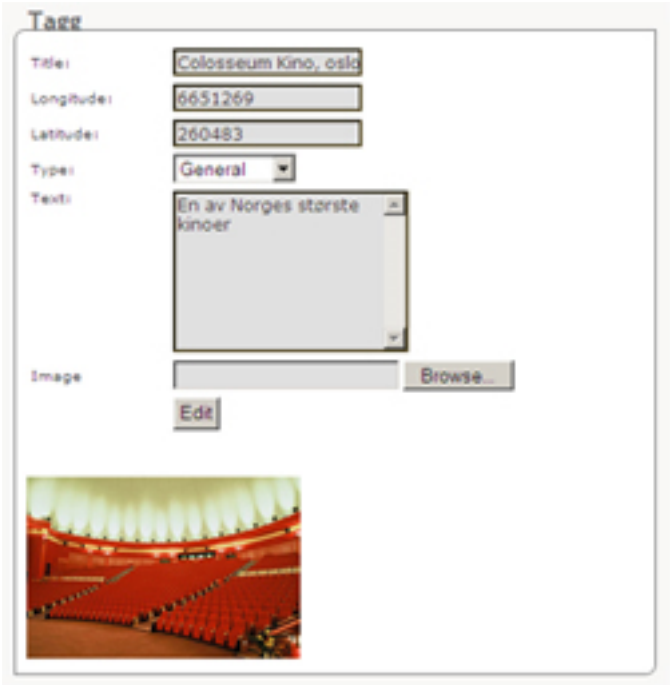


Figure 3.9: Editing a tag in the web-interface

senting the POI in the centre. Making this image when viewing from a web site and not when uploading information to the server enables easy changing of layout and saves uploading time.

3.5.2 User handling

User registration login and editing is implemented in the web interface. The register functionality ensures a unique login name and that combined with a password servers as user identification. The login name is used both from the mobile client and the web pages. In addition the web interface offers functionality for editing the user information such as name, password and home location.

The user handling on the server would be extended to include user

groups in a future application but time has been short and this functionality has been sacrificed to other implementations. Adding groups to the application can be done without changing the current application.

3.6 Future Applications

The different applications that can be made width a system as this are numerous. Some of these will be discussed briefly here.

Entertainment

An application that allows people to write info and back it up with pictures taken with the mobile phone is one of the most apparent uses. Combined with a well developed user system this could be used for a kind of geo chat program. For more private use this could be used as a photo album and travel diary. The additional functionality needed for this is easily implemented.

You could also extend the application to be a computer game. This is of course a whole other application but you could use the basic functionality. We have no concrete ideas for a game at the present but it is possible and will be done sooner or later.

Information provider

Another use is an information service such as a city guide. This could include information for tourists as well as local information that is updated daily. You could find out what's happening on the local pub or concert hall or where the different tourist attractions are. It could be a useful tool for the local business to advertise and inform the public.

There is not many towns in Norway that could support such service but it would be possible to make one client that could be used all over

Figure 3.10: Viewing a tag in the web-interface

Norway. For use in larger cities this could be an application with a marked big enough to support the cost.

Business tool

For companies width a large number of units that need directions this cold be a valuable tool. From ambulances, taxies, pizza express to farmers tracking sheep. There is work needed to build such applications but the foundation remains the same.

Chapter 4

User tests

The main objectives of the usability test on the client are to identify the major shortcomings of the system and get some indication on how end users will approach the application. The test will focus on tree central tasks; navigating the map, finding POI information and adding new POI information. Some of the problems using the client are known before the test start, but all of the main tasks should be achievable by users unfamiliar to the system.

Test group Since the application first and foremost is an application framework, little can be said for certain of the expected user group. To narrow the scope, the geo graffiti application was chosen as a base for estimate of the user group. The user group was not thoroughly examined since this is time consuming and out of our timeframe.

Most vital for a users ability to use an application such as this was thought to be experience in using application on a mobile device and general technical interest. The majority of the test users will have average mobile experience and moderate general technical interest.

4.1 Test objectives

The objectives of the usability test is to answer the following questions:

1. Are the users able to navigate the map?
2. Are the users able to find desired POI information?
3. Are the users able to add POI information?
4. Are the users interested in using a application such as this?

The first three tasks are centered round the technical challenges related to creating user interfaces using small screens. How is the information and possibilities of the application best presented to the user? Are there ways to create a more intuitive user experience?

The last question tries to measure the interest for an application like Growser, and if people are willing to use mobile devices in new ways.

4.2 Flow of the test

The users will first be given a quick introduction to the application. Then they will be presented with the first task by the tester. The time each user needs to perform each task will be recorded, and so will any errors made. When the user has completed a task, he will be given the next to the end of the task list. When the test is done, the tester will engage the user in a informal chat about the application.

Task list

The tasks will be given to the user one at the time by the tester and will be the following.

- Navigate to a given known point using the different zoom levels

- Add a new POI on a location chosen by the user.
- Find a POI by using the search function

4.3 Test Results

The results of the usability test were compromised due to several factors. First of all the tests were done on emulators and not mobile phones. This affects the tests since people have a different attitude towards working with computers compared to mobile phones. Secondly the test group had a higher experience width both mobile phones and computers than the wanted group. This means that some of the usability problems with the application are likely left undiscovered. Third, the number of people in the test group were not sufficient to get clear results. Although the test results are not satisfactory, some information was gathered during the tests and will be represented here.

4.4 Test Group

The test group (fig. 4.1) consisted of people with little in common other than being in our vicinity during the test period. We tested the application on eight users, aged from 20 to 29 years, 75% male and 25% female.

4.5 Tasks performed

All users managed to use the zoom function to view the topmost level in reasonable time. Only one managed however to use the zoom function to navigate from the topmost level. All other users believed that the zoom function used the centre of the screen and not the cursor as the point you zoomed into. As a result the user felt lost and confused.

	Modest	Less than avg.	Avg.	Above Avg.	Extensive
Mobile Experience		2	3	2	1
ComputerExperience			2	4	2

Figure 4.1: Test Group

All users managed to add a new POI from the application but three users used more time than needed. Finding a POI with the search function was also completed by all users but most found the search field ambiguous.

Some users had problems with the toggle layers function. This was mostly due to the naming of the menu function. The menu item reads ”Toggle layers” and it enables you to turn off undesired categories of information. When understanding which menu item had the desired function, there was no problem with using the function.

4.6 Possible improvements

The tests revealed some lacks in both the usability of the application and the functionality. The most apparent problems were with the zoom function. Possible solutions would be to improve the visibility of the cursor or animate the cursor before you zoom inn. Other problems were not so clearly visible but some improvements were suggested by users, and needs to be taken into consideration when future improvements are done.

The navigation system needs more testing and the design needs to be analysed. The users were not unison on this but some of the users expressed a feeling of being lost when navigating the map. This was mostly due to confusion when navigating between zoom levels, but some also had problems of connecting the map to real world locations. Reminiscing this problem can be difficult, but a test with different number of zoom

levels or some kind of indication on where you are in the world on the smaller scale maps could be a solution.

The symbols used in the application needs to be more clear in their meaning. This is a problem that was known before the test was done, but the test made it even clearer. As it is now all categories have one icon. Making icons that spans over such a wide thing as categories may prove difficult if not impossible. Opening for a wider number of categories and subcategories and thus allow for a wider number of icons may be a solution.

Chapter 5

Conclusion

After working on this project, we have realized the usefulness of geo-aware applications. Although our focus was on geographitti, we feel that the produced framework could be used in numerous ways. There are, however, many technological and social challenges to overcome for such applications to become popular on mobile devices.

While the J2ME-framework[14] has many advantages when it comes to rapid development, and prototyping, the implementation can differ hugely from phone to phone. Tailored releases are in most cases needed. J2ME, however, is a young technology, and we hope that it will mature and hardware developers will make their products more standards compliant in the future.

The other consideration when developing mobile applications is the users, and their frame of mind when using mobile phones. Mobile devices are hampered with small screens, cluttered interface, and tedious input methods. This can quickly lead to the users getting tired with a complex application. However, it is now that the foundation for next-generation mobile applications is experimented with. When mobile hardware comes of age, we most certainly will see a wide range of geo-aware applications.

The use of ordinary HTTP-protocols has been a good choice for client/server-

communication. As these protocols are easy to implement on most devices, the possibilities of further development and enhancements are made feasible.

As a project of this kind relies on user-input, there is an aspect of social science in this too. If users were given means of manipulating geographic data, it would be interesting to see which way the system would go [15]. This would provide for an interesting study in social science, but is clearly out of scope of this project.

The Growser framework shows us that geo-aware mobile applications is possible with current available technology, and we hope that it will be a small brick in the strive towards a geo-enabled future. Even if the current implementation of the framework, the methods used are suitable for further development.

Bibliography

[1] W. Nick, "Urban tapestries: The spatial and the social on your mobile," *Proboscis*, 2005. http://proboscis.org.uk/publications/snapshots.html/.

[2] "Map and go home," 2005. http://www.map-and-go.it/.

[3] "The michelin guide." http://www.viamichelin.com/.

[4] "Map24." http://www.map24.com/.

[5] M. Gunnar, "One map," *Hit*, 2002.

[6] "Web map service (wms1.3)," tech. rep., 2004. http://portal.opengeospatial.org/files/?artifact_id=5316.

[7] "Statens kartverk." http://www.statkart.no/.

[8] "Opengis location services (openls): Core services [parts 1-5] (ols core)," tech. rep., 2005. http://portal.opengeospatial.org/files/?artifact_id=8836.

[9] "Jsr179 api," tech. rep., 2004. http://www.jcp.org/aboutJava/communityprocess/final/jsr179.

[10] C. Robert, "Kxml: A great find for xml parsing in j2me," *DevX*, 2003. http://www.devx.com/xml/Article/11773/0/.

[11] "Tagandscan home," 2005. http://www.tagandscan.com/.

[12] K. Beck, *Extreme Programming Explained*. 2004.

[13] D. Rushkoff, "Honey, i geotagged the kids," *TheFeature*, 2005. http://www.thefeature.com/article?articleid=101490&ref=7375623.

[14] "Java - j2me." http://java.sun.com/j2me/docs/j2me-ds.pdf/.

[15] J. W. Schuyler Erle, Rich Gibson, *Mapping Hacks*. O'Reilly, 2005.

Appendix A

Twelve rules of Extreme Programming

1. **Planning**, sometimes called the Planning Game. The XP planning process allows the XP ”customer” to define the business value of desired features, and uses cost estimates provided by the programmers, to choose what needs to be done and what needs to be deferred. The effect of XP’s planning process is that it is easy to steer the project to success.

2. **Small releases**. XP teams put a simple system into production early, and update it frequently on a very short cycle.

3. **Methaphor** XP teams use a common ”system of names” and a common system description that guides development and communication.

4. **Simple design** A program built with XP should be the simplest program that meets the current requirements. There is not much building ”for the future”. Instead, the focus is on providing business value. Of course it is necessary to ensure that you have a good design, and in XP this is brought about through ”refactoring”, discussed below.

5. **Testing** XP teams focus on validation of the software at all times. Programmers develop software by writing tests first, then software

that fulfills the requirements reflected in the tests. Customers provide acceptance tests that enable them to be certain that the features they need are provided.

6. **Refactoring** XP teams improve the design of the system throughout the entire development. This is done by keeping the software clean: without duplication, with high communication, simple, yet complete.

7. **Pair Programming** XP programmers write all production code in pairs, two programmers working together at one machine. Pair programming has been shown by many experiments to produce better software at similar or lower cost than programmers working alone.

8. **Collective Ownership** All the code belongs to all the programmers. This lets the team go at full speed, because when something needs changing, it can be changed without delay.

9. **Continuous Integration** XP teams integrate and build the software system multiple times per day. This keeps all the programmers on the same page, and enables very rapid progress. Perhaps surprisingly, integrating more frequently tends to eliminate integration problems that plague teams who integrate less often.

10. **40-hour Week** Tired programmers make more mistakes. XP teams do not work excessive overtime, keeping themselves fresh, healthy, and effective.

11. **On-site Customer** An XP project is steered by a dedicated individual who is empowered to determine requirements, set priorities, and answer questions as the programmers have them. The effect of being there is that communication improves, with less hard-copy documentation - often one of the most expensive parts of a software project.

12. **Coding Standard** For a team to work effectively in pairs, and to share ownership of all the code, all the programmers need to write the code in the same way, with rules that make sure the code communicates clearly.

Appendix B

XML Formats

B.1 Getmap

```
<?xml version="1.0" encoding="utf-8"?>
<poiinfo>
<map>Relative link to picture file</map>
<maplink>
<north>Relative link to get map request for north tile</north>
<west>Relative link to get map request for west tile</west>
<east> Relative link to get map request for east tile</east>
<south> Relative link to get map request for south</south>
</maplink>
<tag>
<cat>Category id</cat>
<posx>X position in pixels on the picture</posx>
<posy>Y position in pixels on the picture</posy>
<title>Title</title>
<descr>Describing text</descr>
<timestamp></timestamp>
<image_link>Relative link to small image</image_link>
<thumb>Relative link to thumbnail image</thumb>
</tag>
</poiinfo>
```

B.2 Init request

```
<?xml version="1.0" encoding="utf-8"?>
<userinfo>
<name>User name</name>
<uid>User id</uid>
<loginid>Unique session id</loginid>
</userinfo>
<categories>
<cat>
<id>Category id</id>
<type>Category type</type>
<descr>Description of category</descr>
<name>Name of category</name>
<visible>1 or 0</visible>
<icon>Relative link to icon file</icon>
    </cat>
</categories>
```

B.3 Search request

```
<?xml version="1.0" encoding="utf-8"?>
<taginfo>
<tag>
<id>POI id</id>
<pos>latitude ,longitude</pos>
<title>Title of poi</title>
<maplink>Getmap request for map location of POI</maplink>
</tag>
</taginfo>
```