Towards the integration of vector map graphics in mobile environments

Martin Waldén

Department of Communication Systems
Lund Institute of Technology
Lund University
Abstract

The world is turning towards globalization and mobility. The notion of mobile internet has hurried the evolution of mobile phones, PDA:s, and hybrids called smart-phones. As technology advances more attention has been given towards the supply and development of commercial value-added services in mobile environments. One example of value-added services is location-based services, which refer to applications in mobile devices reacting to a geographic trigger. Some of these applications allow a user to visualize its location on a map. However, the mobile environment introduces limitations such as slow transfer rates in wireless connections, and limited memory and processor capacity. This study aims to derive a set of design proposals when providing spatial data for visualization in mobile devices in order to overcome the limitations. Since spatial databases are quite large, the data needs to be represented in an efficient manner to enable a fast visual feedback. The study defines a number of functional, cartographic, design, and performance requirements on the representation. The impact of the requirements was analyzed by performing two activities; an evaluation of existing formats in order to derive a number of design proposals for implementing a vector representation, and an evaluation of performance in a real system providing an authentic location-based service. The result of the study is ten implementation-ready proposals in order to meet the requirements. Some of the proposals were found to be feasible to implement in the system providing the location-based service. Test and validation results from the implementation were used in deriving the proposals.
Sammanfattning

Contents

1 INTRODUCTION ............................................................................................................ 1
   1.1 BACKGROUND ........................................................................................................... 1
   1.2 PROBLEM DISCUSSION .............................................................................................. 2
   1.3 PURPOSE .................................................................................................................... 2
   1.4 DELIMITATION ............................................................................................................ 2
   1.5 OUTLINE .................................................................................................................... 3

2 METHODOLOGY ........................................................................................................... 5
   2.1 DEFINITIONS ................................................................................................................ 5
      2.1.1 Vector data and vector format ........................................................................... 5
      2.1.2 Geospatial maps ................................................................................................. 5
      2.1.3 Location-based service ....................................................................................... 6
      2.1.4 Location-based service system ........................................................................... 6
   2.2 METHODOLOGY OF CHOICE ......................................................................................... 7
      2.2.1 Case study - Wayfinder Systems AB .................................................................... 8
   2.3 REQUIREMENTS ......................................................................................................... 9
      2.3.1 Cartographic criterion ....................................................................................... 9
      2.3.2 Functional criterions ........................................................................................ 10
      2.3.3 Performance criterions .................................................................................... 11
      2.3.4 Design criterions .............................................................................................. 11
   2.4 ACTIVITIES ............................................................................................................... 12
   2.5 PERSPECTIVE ............................................................................................................ 13
   2.6 DATA COLLECTION .................................................................................................... 14
   2.7 VALIDITY AND RELIABILITY ...................................................................................... 14

3 VECTOR MAP GRAPHICS .......................................................................................... 15
   3.1.1 Metafiles ........................................................................................................... 18

4 EVALUATION OF FORMATS OF VECTOR GRAPHICS ............................................ 19
   4.1 SVG .......................................................................................................................... 19
   4.2 MOBILE SVG .......................................................................................................... 20
      4.2.1 Richness of graphical presentation ..................................................................... 22
      4.2.2 Possibility to identify objects .............................................................................. 22
      4.2.3 Possibility of progressive data transmission ..................................................... 23
      4.2.4 Size of data, transfer time via wireless networks ............................................. 23
List of figures

Figure 1. Outline of the thesis................................................................. 3
Figure 2. Overview of the LBS application and system.......................... 6
Figure 3. The Wayfinder application visualizing the location of the user in a map. ........ 7
Figure 4. Development process.............................................................. 8
Figure 5. The Wayfinder application in a Nokia 6600............................ 9
Figure 6. To the left the original image is displayed. In the middle a vector graphic version of the image is zoomed. To the right a bitmap version of the image is zoomed........... 16
Figure 7. Illustration of traditional and recent trends in location-based service systems. ...... 17
Figure 8. Illustration of the relations between SVG Mobile profiles, SVG 1.1, XML and SGML............................................................................................................................... 20
Figure 9. To the left an illustration of client side and to right the server side architecture of the SVG generation................................................................................................................................. 25
Figure 10. Software architecture of the SVG gateway................................ 25
Figure 11. Illustration of RaveGeo™....................................................... 28
Figure 12. Illustration of MapTP™.......................................................... 36
Figure 13. Illustration of the layers in MapTP™ platform............................ 37
Figure 14. Resolutions of the spatial data transmitted to the client in order of increasing detail................................................................................................................................ 46
Figure 15. Illustration of the vertex reduction algorithm......................... 49
Figure 16. Illustration of the selection of the Douglas-Peucker algorithm......................................................................................................................... 50
Figure 17. Illustration of recursion of the Douglas-Peucker algorithm........... 51
Figure 18. Illustration of the hull of a polyline. Hull vertex indeces are stored and reused in the algorithm. ................................................................................................................... 52
Figure 19. Object a) qualifies as a homogeneous network and b) and c) as heterogeneous networks. ....................................................................................................................... 53
Figure 20. Illustration of coordinate quantization by point predicting............ 55
Figure 21. Illustration of bounding regions.............................................. 56
Figure 22. Reusing of query answers...................................................... 57
Figure 23. Cache strategy for retrieving spatial objects in a map................... 59
Figure 24. Clusters that form the dictionary of the FHM algorithm............... 61
Figure 25. The distribution of a) $\Delta_{i-1}$ and b) $\Delta_{0}$ dictionary.................... 62
Figure 26. Map generalization design..................................................... 63
Figure 27. Extension of the map generalization model.............................. 66
Figure 28. Screenshot of the application simulating the mobile phone........... 67
Figure 29. Strategy scheme for lifecycle validation of software performance......... 74
List of tables

Table 1. A comparison between SVG Basic and SVG Tiny. The ‘*’ indicates that the feature is reduced in functionality compared to SVG 1.1. ........................................................... 21
Table 2. Examples of features available in the SlimMap format. .................................................. 31
Table 3. Criterions for including objects in map image in the SlimMap format. ............................. 32
Table 4. Data transfer rates of wireless connections and allowed size of image in SlimMap format. .............................................................................................................................. 33
Table 5. Description of the layers in MapTP™ platform. ............................................................... 36
Table 6. Comparison between SVG Basic, RaveGeo™, SlimMap, and MapTP™. The ‘*’ means that data is compressed with Zlib. The GfxFeatureMap is left out due to risk of compromising the business case of Wayfinder System AB. .................................................. 41
Table 7. Complexity in the ordo-notation for polyline simplification algorithms. ....................... 52
Table 8. Examples of the reduced number of bits with the different representations. .................... 55
Table 9. Probability of each use case defined in the LBS system. .................................................. 82
Table 10. Methods for each software module derived from UML class diagrams. ...................... 82
Table 11. The client/server structure and multiplicity of the modules of the considered system. .......................................................................................................................................... 86
Table 12. Nodes of the precedence graphs grouped into subset for each actor. ............................ 88
Table 13. The software tasks of the LQN and their roles as pure client, active server, or pure server. ........................................................................................................................................... 95
Table 14. Identification of entries of software tasks. ....................................................................... 96
Table 15. Device calls of each software task. .................................................................................. 97
Table 16. LQN parameters of the LBS system. .............................................................................. 98
1 Introduction

The following chapter describes the background leading to a discussion concerning the emphasis of the thesis. The discussion leads to the formulation of the problem that is to be analyzed. The formulation of the problem results in the purpose of this thesis. Finally, delimitations are presented and a complete outline of the thesis will be presented.

1.1 Background

The evolution of mobile phones, PDA – Personal Digital Assistants, and hybrids called Smartphones allows millions of users the possibility of reaching the Internet. The notion of Mobile Internet has set the path of developers of mobile devices, operators, and other service providers by the introduction of GPRS and later UMTS. The attention towards Mobile Internet has mainly focused on future market trends and technological developments. However, recently more attention has been given towards the supply of commercial services. Successors to WAP and SMS services are establishing, which will enable full exploitation of the Internet.

The market penetration of Mobile Internet services has been crippled by lack of standards on terminals and networks. As more developers tend to turn towards platform development this hold-back ought to be temporary. It is becoming quite clear that the revenue is shifting from the actual connection and transmission to value-added services within mobile networks.¹ The service providers to Mobile Internet can appreciate the synergy effects of traditional Internet services plus the competitive advantage of being mobile. Thus, the demand for applications and services in mobile devices is heavily increasing. A great deal of new applications providing new services, are being introduced. Online applications may be identified and sorted in categories such as content-based (information providers such as news etc.), commerce-based (commercial transactions in products etc.), community-based (e-mail, chat etc.), and service-based applications (value-added services such as search engines, maps, route planners etc.).² One example of new service-based applications is location-based services – LBS, which refer to applications reacting according to a geographic trigger. Such trigger could be the input of a town or the position of a mobile phone. Now it is possible to take part of up-to-date, interactive, informative graphical presentations directly in mobile phones and PDAs, instead of carrying around a laptop or even considering a printout.

² Rangone, A. et. al, Mobile Internet: An emperical study of the evolution of the supply of B2c Mobile Internet Applications in Italy (2002).
1.2 Problem discussion

It becomes more evident that the need for graphics in applications evolves as more and more mobile phones and PDAs are equipped with colour displays and powerful CPUs. It is possible to develop applications like LBS with advanced functionality and graphical presentation in mobile devices. Still these devices have limited resources. Each mobile device has different characteristics in terms of CPU speed, memory size, and colour support. In order to develop attractive applications the software developer must satisfy at least three critical demands. First is the demand for functionality. The application must satisfy a customer’s needs regarding functionality or create a need that the customer does not already have.3 Secondly, the appearance of the application, i.e. the graphical presentation, must be attractive. For example in an application for LBS, the most important part might be the presentation of maps. Finally yet importantly is the responsiveness of the application. Regardless of how impressive the functionality or the appearance of the application is, it becomes unattractive to a customer if it takes several minutes to display a single map. Thus, the performance aspect is extremely important. The most critical part is the performance and responsiveness of the server providing the application with graphical data.

1.3 Purpose

The purpose of this thesis is to analyze an efficient vector representation of geospatial data suited for visual presentation and interpretation in a mobile environment. In addition, the representation, or format, should be suitable for an efficient transfer of geospatial data over wireless connections with limited transfer rate. This study aims to derive a number of design proposals of a satisfactory format in order to concreticize the areas of improvement and their impact on the LBS system. The proposals will include functional, cartographic, design, and performance demands.

1.4 Delimitation

This study sets its focus on the derivation of design proposals in order to meet certain requirements. The representation of geospatial data is restricted to form the shape of a map, due to the outline of the existing LBS application of Wayfinder Systems AB. The vector formats chosen for evaluation is selected based on user experience, internal information at Wayfinder Systems AB, and industry awareness. Some of the formats are designed to meet

---

3 Many examples of products, that have created a need and a demand accordingly, exist. One example is SMS in mobile phones.
the properties of LBS, while some are not. In addition, standards that are more recent are selected prior to older, e.g. why SVG is chosen instead of CGM and WebCGM.

### 1.5 Outline

The thesis starts with a discussion of the problem leading to the definition of the objective and purpose of the study. Next, a number of requirements of the vector representation are defined and two activities derived in order to meet the requirements. The first activity involves an evaluation of existing vector formats in order to determine and evaluate useful properties when representing spatial vector data in mobile environments. From the evaluation, a number of design proposals are derived. Some proposals were found feasible to implement in an authentic LBS system and are verified and validated. The second activity involves a performance evaluation of an authentic LBS in order to estimate the impact of implementing some of the design proposals in the LBS system. A performance model of the LBS application provided by Wayfinder Systems AB is generated and used to estimate response time and to identify possible bottlenecks. The results from the activities are summoned and presented along with existing research in a number of fields identified as relevant to meet the requirements of the study. Finally, conclusions and a discussion of future work are presented.

![Outline of the thesis](image-url)
2 Methodology

This chapter begins with the definitions of frequent used terms and expressions in order to reduce the possibility of misinterpreting their context. Next, an explanation of the methodology to fulfill the purpose of this thesis will be presented. A number of criterions, subjects for evaluation have been selected. Further, the procedure for gathering data, information and theory needed, will be presented. Eventually a discussion concerning the reliability and validity of the study will close the chapter.

2.1 Definitions

To help clarify and avoid misinterpretations, definitions of frequently used terms will be presented.

2.1.1 Vector data and vector format

In this thesis, the terms vector data and vector format will be used frequently. Vector data, or rather vector spatial data, refers to object-oriented spatial data usually defined as sets of spatial entities. The vector spatial data typically includes spatial inter-relations, which describe a geographical area. A vector format refers to the way the spatial data is represented or a defined purpose. For example, a vector format would describe how entities are formed of points, lines and polygons with additional meta-data. However, the definition of a vector format could be expanded to include the definition of a transfer protocol, and data extraction algorithms such as selection and reduction of data.

2.1.2 Geospatial maps

A geospatial map may be represented as a vector or as a raster model according to the Geographic Information Systems - GIS community. The vector model refers to an entity-based view, where space constructed from objects that fill space, while the raster model refers to a space oriented view, where each point in space has some properties. This thesis will assume that geospatial data is stored in accordance to the vector model.

---

2.1.3 Location-based service

There is no true definition of the term *location-based service – LBS*, thus the term has given rise to several interpretations. A common description include activities such as estimation of the location of the mobile user, based on estimated location a service is produced, and finally deliverance of location-enhanced service to the user. However, the definition of the LBS used here refers to services, which react according to a geographic trigger. Here the notion of LBS systems mainly refers to applications in mobile devices presenting geospatial data, usually in the shape of a map. The services are accessed via a wireless network represented as *wide area network – WAN* in Figure 2.

![Figure 2. Overview of the LBS application and system.](image)

2.1.4 Location-based service system

The LBS application and system subject of investigation is developed by Wayfinder System AB. The system involves an application in mobile devices connected to a server via a wireless network. The application allows, among other features, a user to visualize a location on a map or to search for the shortest route between two locations and displaying it on a map. The user is able to pan, resize, rotate and zoom in or out in the map. The application also enables information to be linked to objects in the map, allowing the user to receive additional information of points of interest. A more detailed description of the system is presented in 8.1.
2 Methodology

Figure 3. The Wayfinder application visualizing the location of the user in a map.

2.2 Methodology of choice

Software engineering refers to the study of software processes, development principals, techniques and notation. Further, it includes production of quality software, delivered on time, within budget and foremost satisfying users' requirements. Software development is often an iterative and incremental development process, where new functionality is frequently added. The development process usually exists of a number of phases to distinguish planning, requirements gathering, production, and testing and validation. The process adopted in the thesis, illustrated in Figure 4, corresponds to the waterfall model. The first phase is inception, where the scope of the thesis is decided i.e. the enhancement of map visualization and user-map interactions. The purpose is stated in 1.3. The elaboration phase is where detailed requirements are collected and are presented in the following sections. The definition of criterions summons the most critical aspects of the analysis. The criterions include functional, cartographic, design, and performance demands and is presented in 2.3. In the construction phase the first of the two activities will occur. To investigate the impact of the functional, cartographic, and design demands an evaluation of existing formats and standards will be

---

performed as well as evaluating existing and related research e.g. map generalization etc. The result of the evaluation is a number of design proposals found critical in order to fulfill the requirements. Proposals that are found feasible to implement due to time limitations of the study, will be implemented in the server at Wayfinder Systems AB (see 2.2.1) and a Java application simulating the application in mobile devices. The last phase is the transition phase. The work here includes the second activity; investigating the impact of the performance demands. A performance model will be generated in order to evaluate performance issues for adding new functionality in the system. A further description of the activities is presented in 2.4

![Development process](Figure 4. Development process.)

2.2.1 Case study - Wayfinder Systems AB

One company developing LBS applications to mobile devices is Wayfinder System AB located in Lund in Sweden. Wayfinder provides wireless traffic-aware navigation, routing services and mobile location services that incorporate maps, proximity searches, directions, and routes for multiple forms of transport. This study be performed as a case study and will include information and data gathered from Wayfinder Systems AB. In addition, the LBS application developed by Wayfinder will be used throughout the study as a model of a typical system providing an LBS. In their application for handheld devices, digital images of maps as a way of representing geospatial data are delivered over wireless connections such as GPRS. The application allows the user to pan, resize, rotate and zoom in or out in the map. The application also enables information to be linked to objects in the map, allowing the user to receive additional information of points of interest.
2.3 Requirements

In order to concretise the critical requirements on the representation of geospatial in an LBS system, a number of evaluation criterions are identified. The criterions are derived from the unique properties of LBS and mobile environment. The criterions are a selection of the most critical key success factors. In the following sections a short discussion of the requirements leading to the criterion will be given. The statement of each criterion is presented after the short discussion.

2.3.1 Cartographic criterion

The graphical presentation of the spatial data must provide a certain richness of graphical features to be considered attractive to a user. A suitable metric to measure how effective the contents of the application are is the ratio of time spent waiting for content versus the time spent viewing content. Users are willing to wait a little extra time to view something particularly compelling, but there is an upper limit. In applications for LBS, the presentation of maps is key.

---

The intelligence handling the graphical presentation may differ depending on the developer’s preferences. Two extremities can be derived, based on where the intelligence is allocated. The first extremity is where the application located in the mobile device is handling the graphical interpretation of the geospatial data. The other is where the representation format includes not only geospatial data but also cartographical data. Cartographical data describes the geospatial data and how it is presented in the map.

- Richness of graphical presentation.

### 2.3.2 Functional criterions

It should also be possible to identify objects in the map to be able to associate additional information to objects in the map. For example, restaurants, hotels or other points of interest should be able to be linked to information stored in a database, allowing the user to interact with the contents of the map. The format should therefore allow an object-oriented representation of the map including their topological relations. It is necessary that each object may be identified by a unique identification or a name, implying the use of a vector format for representing the geospatial data.

- Possibility to identify objects and topological relations.

In order to minimize the waiting time for the user progressive data transmission, usually called streaming technique, may be used. Streaming content is a technique to minimize the waiting for data versus viewing data ratio. By adopting an object-oriented approach, the streaming technique is made easier. However, not all vector graphics formats allow this technique since the content needs to be validated before being displayed.

- Progressive vector data transmission.

Wireless connections like GPRS offer limited transfer rates compared to broadband connections available to desktop computers. Datasets of geospatial data are usually quite large and methods to limit and/or reduce the size need to be investigated in order to meet the limitations of the mobile environment. Methods to reduce size would include intelligent data selection (generalization), compression, and to avoid data redundancy.

- Size of data, transfer time via wireless networks.
2.3.3 Performance criterions

The performance of the system providing the LBS is a critical factor for the service to become attractive to a user. Typically, LBS systems struggle with limited bandwidth and large amounts of geographic data being processed. Software and hardware contention models may deliver performance bottleneck identification and utilization predictions as guidelines for hardware partitioning and modules mapping. In addition, performance models may provide supporting foundations when deciding on additional system functionality. The visualization of data is only a set of subtasks within the server, which if designed poorly could result in a system bottleneck. Accordingly, the way the geospatial data is represented is of utmost importance when extracting and converting the geospatial data in the server. A performance evaluation can support the design decisions.

- **Performance, response time of the system.**

Since mobile devices typically are limited in terms of memory, it is not possible to store large amounts of data for visualization in the mobile device. Therefore, actions must be taken to cover the event of updating data and requesting new. Nor is it possible to allow for a complicated extraction of data and rendering algorithm in terms of memory or MIPS in the processor. For instance, data processing such as unnecessary structuring and ordering as well as complex compression algorithms should preferably be avoided.

- **Memory and CPU demands in mobile device limiting data processing.**

2.3.4 Design criterions

A fundamental demand on the vector format is that it should be possible to extract and convert the data from the existing format used to store geospatial data to the vector format. For example, the server running on Wayfinder Systems AB uses an internal map format, which can be created from almost any digital map format. However, the server format is not optimal for transferring over wireless connections or graphical presentation. Instead, it is designed to perform extremely fast geospatial searching and route optimization based on journey time predictions. The implementation of a vector format means that algorithms for extracting and converting data stored in the server, transfer protocol, and extracting and rendering in the client application must be developed. It might also mean that additional hardware with a supporting software module must be installed. This raises economical issues in terms of extra hardware or license cost etc.

- **Design of implementation and its resource demand.**
The last criterion refers to the effort of implementing the extraction and conversion algorithms in the software and hardware architecture of the system providing the LBS, transfer protocol, and extraction algorithms in the mobile device. It might prove to be more cost effective to implement a less advanced vector format, if it still enhances the expectations of a user to some extent. The return of investment of new functionality and design alternatives depend on a number of factors and should be taken under consideration. The time to market is a valuable measure, which is influenced by competitive pressure and product introduction barriers set by competitors as well as the market situation. Qualitative estimations of user expectation should also be performed.

- Demand on time resources of development.

### 2.4 Activities

In order to initiate the evaluation of a suitable format for representing geospatial data, two activities have been chosen. The first activity involves an evaluation of existing vector formats and standards. The evaluation focuses on whether the formats are suitable for representing spatial data, and provides useful insights when developing a new or updating a vector format. The formats are then compared to each other resulting in reflections of which features of the formats that are useful and ought to be introduced in a system providing an LBS. The chosen vector formats are a pre-selection of formats based on user experience, industry awareness and internal information at Wayfinder Systems AB. The vector formats chosen for evaluation is the open standard Mobile SVG – a subset of SVG 1.1, RaveGeo™ – a patented format developed by Idevio, SlimMap and GfxFeatureMap – formats developed by Wayfinder Systems AB, and MapTP™ - a format developed by NETSOLUT. In addition to the evaluation, existing research in a number of fields identified as relevant to meet the requirements of the study are investigated. The results from the evaluation and research are a number of design proposals of a vector representation of spatial data in a mobile environment. The proposals that are found to be feasible to implement is implemented and integrated in the system at Wayfinder System AB. The implementation is tested and validated in a simulated environment.

The second activity involves a performance evaluation by generating a performance model of the LBS system. The evaluation focuses on determining the response time and identifying bottlenecks in the system by simulating the impact of adding new functionality provided by the design proposals. The performance estimation is also a way of describing and getting a fundamental understanding of the behaviour of the LBS system and its critical parts. Hence, it is particularly important to understand the dependencies between software modules and on which hardware devices they are executed. The LBS system will be modeled as a *layered queuing network – LQN*. The LQN is an adaption of queuing models of systems, which takes
into consideration client/server relations among software and hardware servers and resources. The LQN is closely linked to software specifications, which makes it easy to develop and understand. The model will be generated and derived by using a method providing a step-wise performance model derivation from existing development documents.\textsuperscript{7} A set of key success factors of an LBS system (a more thorough description of the system is given in 8.1) can be derived, whereas responsiveness, throughput and scalability are critical factors. The responsiveness of the server is determined by the ambition where the user experiences that the maps are stored in the mobile device rather than on a server. In a sense, almost all of the criterions mentioned earlier are affecting the performance and responsiveness of the application and the server. Results from the evaluation are presented in chapter 9.

\section*{2.5 Perspective}

The criterions given in the previous section are merely a selection of some of the possible requirements one could derive when determining a suitable or implementing a representation of spatial data. A complete thorough analysis of some of the criterions mentioned is beyond the scope of this thesis, and therefore only qualitative discussions will be given. However, some criterions are more thoroughly analyzed resulting in quantitative measures. A chosen strategy of this thesis is to integrate software performance engineering (see 7.1) in decisions of development, e.g. adding new functionality to an existing application and system. It is however possible to use different approaches involving other criterions, which might lead to other conclusions.

Some of the criterions may contradict each other. Seeing that the speed of the wireless connection should be transparent to the user, the size of the data transferred must be small byte-wise. This might result in a heavy compression of data in the implementation. A time and resource consuming extraction and conversion algorithm in the server as well as in the mobile device, enabling a fast transfer via the wireless network, might however prove not to be desirable. If implementing the ability to stream the map content, the number of calls to and callbacks from the server increases drastically, i.e. the load on the server increases drastically. The implementation must also allow for a minimum of richness of the format restricting the level of compression. Thus, the decision is somewhat of a paradox, implying that the properties must be balanced in sensible manner.

\textsuperscript{7} Cortellessa, V. et. al, \textit{Automatic derivation of software performance models from CASE documents} (2001).
2.6 Data collection

The objective of this study is to derive a number of design proposals of a representation of spatial data suitable for interpretation and visualization in a mobile environment. The study is performed as a case study of Wayfinder Systems AB – a provider of LBS. The purpose and discussion are of explorative nature, and voucher for an analysis of real-life scenario. The primary data is collected from Wayfinder Systems AB by interviewing design experts to gather information of system and software design, user behaviour, and information on LBS. Data is collected by performing measurements on the existing system at Wayfinder Systems AB, and going through legacy documentation produced for the LBS application.

Secondary data is defined as data referring to primary data, but is not collected at the source of the object of study, i.e. the LBS application. The secondary data is collected from published research material and literature, such as books and articles in Geographic Information Systems, Spatial Data Handling, Mobile Internet, Computer graphics, Software Engineering, Software Performance Engineering, and Data mining. Information concerning vector formats such as RaveGeo™ and MapTP™ is gathered from interviews with personnel at Idevio and NETSOLUT.

2.7 Validity and reliability

The validity of a study describes the resemblance of collected data and the application of data from which conclusions are drawn. Since the primary data of this study is collected mainly from an existing LBS application and system, the data guarantees partial validity before being processed. In addition, the ambition of the data collection is to be able to apply the conclusions to any LBS application with similar properties. The reliability of the primary data is improved by performing semi-structured interviews with design experts. The secondary data is gathered from well-quoted articles and published materials in research disciplines mentioned in 2.6. However, since the resulting design proposals are not yet fully implemented, the performance estimations cannot be properly validated. Still the estimations may provide useful insights affecting the choice of proposals. At last, the feasibility of the study is reflected upon. The feasibility refers to if there is some sort of contribution to existing research or if a company implementing the proposals will benefit from the study in any way.
3 Vector map graphics

This chapter explains the difference between bitmap and vector graphics and the integration of vector graphics and vector spatial data resulting in the term vector map graphics.

When discussing graphical presentation of data, two significantly different types of graphics must be explained. The first is bitmap graphics, or raster graphics, and the other vector graphics. Until recently most digital graphics were of the first type. Due to limitations to the properties of bitmap graphics the world is turning towards vector graphics. Vector graphics is however not a new phenomenon, it has been around since the 1950s. The first popular PC vector graphics format was the Computer Graphics Metafile standard – CGM, which the International Organization for Standardization and American National Standards Institute adopted in as late as 1987. Since all modern displays are raster-oriented, the biggest difference between bitmap and vector graphics comes down to where they are rasterized. In the case of vector graphics the image is rasterized in the client, as opposed to as bitmaps already rasterized on the server.

An image based on bitmap graphic, often called a raster image, is composed by pixels. This means that the image is broken down into a grid where the grid cells constitute as picture elements. The grid cells are called pixels and are able to contain only one colour each. Bitmap images are accordingly resolution dependent, where resolution refers to the number of pixels in an image and is usually stated as dpi (dots per inch) or ppi (pixels per inch). Another type of data visualization is vector graphics and is not based on pixels but made up with individual, scalable objects. These objects are defined by georeferenced points connected with lines rather than pixels, which allows them to always render at the highest quality. Thus, objects may consist of lines, curves and shapes with editable attributes such as colour, fill and outline. The attributes are linked to an object by the same identifier. Changing the attributes of a vector object does not affect the object itself. It is even possible to change the number of attributes without destroying the basic object. Points are represented as single coordinate pairs. Points linked together create lines and are stored in a list containing start coordinates,

---

vertices and stop coordinates. Shapes are stored as the lines by which it is surrounded. The objects are represented with spatial inter-relations such as topological order.

Furthermore, vector graphics is resolution independent, which results in scalability and resizability without any distortion what so ever see Figure 6. This feature makes vector graphics appear very smooth looking. Since elements in a vector image are stored as an object, it is possible to link information associated with the object. Another advantage with vector graphics is that the images are not restricted to a rectangular shape as bitmaps. Vector objects can be placed over other objects, and the object below will show through.

![Image](image_url)

**Figure 6.** To the left the original image is displayed. In the middle a vector graphic version of the image is zoomed. To the right a bitmap version of the image is zoomed.

Vector images can quite easily be converted to bitmaps. This process is called *rasterizing* and is performed by calculating the coordinate values to the corresponding position in the raster. Rasterizing a line starts with calculating the equation of the line. The raster cells are then marked as the line crosses them. When a vector image is converted to a bitmap, the output can be a bitmap image of any size. If a bitmap image of a particular size is rendered and a different size is requested by, for instance a map application, it is necessary to export a new bitmap image from the original data. Vector images also have an advantage over bitmap images in that the contents are searchable. The textual content of bitmap images can be seen by humans but is not available to search engines. In vector images, depending on format, the textual content is searchable.

Traditional location-based service – LBS systems rely on extracting geospatial vector data in a server, and rendering a bitmap image of map. The bitmap image has the advantage that it presents compression and produce small files if the image resolution is low. Additionally, bitmap images may be displayed anywhere including mobile devices, since the rendering ability is well attached to existing platforms. However, in recent developed LBS systems the presentation of geospatial data relies on vector data, see Figure 7. *Vector map graphics* represents the integration of vector graphics and vector spatial data. Spatial data, as described in 2.1.2, is usually stored as a vector model, which makes the connection to visualization through vector graphics evident. Objects in a map are stored as chains of coordinates with
topological relations. Application areas like LBS will benefit significantly by using vector graphics, since the technology’s ability to allow users to zoom in on maps to high magnification levels without loss of quality. This allows the user to interact with the map while being offline, i.e. a new map must not be downloaded with every interaction such as zooming. With the introduction of object-oriented vector map graphics, the ability to link additional information to objects in the map is made possible. LBS often include maps with animated objects and hyperlinks associating information of points of interest etc. Since vector images may consist of objects and layers it is easy to identify each object allowing it to be associated with further information. In addition, an object-oriented vector format allows the ability to progressively transmit the map. This streaming technique allows objects to be downloaded in lower resolution first and be displayed while downloading the same objects in higher resolution. This technique minimizes the waiting time for data versus viewing data ratio. In Figure 7, the dotted line in LBS systems today represent transparent callbacks from the server enabling the progressive transfer of data.

Different formats and standards of vector graphics suitable for LBS are evaluated and presented in chapter 4.

---

3.1.1 Metafiles

Metafiles are graphics that contain both raster and vector data. For example, a vector image that contains an object, which has a bitmap pattern applied as a fill, would be a metafile. The object is still a vector, but the fill attribute consists of bitmap data. The object is still a vector, but the fill attribute consists of bitmap data. The most common format is SVG – Scalable Vector Graphics. Other common metafile formats include WMF – Windows Metafile, EMF – Enhanced Metafile and WebCGM – Computer Graphics Metafile.
4 Evaluation of formats of vector graphics

The chapter presents an evaluation of different existing formats and standards for vector graphics. The evaluation aims to provide useful insights on which features are valuable in developing a new or updating an existing vector format. Standards followed by different formats are described and advantages and disadvantages as well as useful features are presented.

4.1 SVG

SVG stands for Scalable Vector Graphics and is a language for describing two-dimensional graphics. It is an open standard of the XML family developed by the World Wide Web Consortium – W3C. It reached recommended status by the W3C on 14 January 2003 versioned 1.1. This section is meant as a brief description of SVG. The following sections will describe two mobile profiles of the SVG 1.1, SVG Basic and SVG Tiny.

By being based on XML, SVG gains many advantages such as a sound basis for internationalisation, powerful structuring capability, an object model, and last but not least a world wide acceptance. Most vector graphics are often structured as nested groups or layers. A map is by nature a layered representation of the earth. By its characteristics, the SVG specification allows the same layering concept. It is natural to express this structure in XML. Besides XML, SVG is also based on other W3C technologies, such as Cascading Style Sheets – CSS and Document Object Model – DOM.

---

11 CSS is a stylesheet technique developed and used for HTML, but is also possible to use with documents in the XML family.
12 DOM is a specification of interfaces, enabling platform independent access to elements and attributes in an XML document, using the objects organized in a hierarchical structure called document tree.
4 Evaluation of formats of vector graphics

4.2 Mobile SVG

However, SVG 1.1 is not designed for use in mobile environments, leading to the development of a profile specification that addresses mobile devices. The profile aimed to allow SVG to render on mobile devices with limited memory, CPU speed and bandwidth. A single such profile is however not sufficient to deal with the variety of mobile devices, since each mobile device has different characteristics in terms of CPU speed, memory size and colour support. Thus, two profiles were defined: SVG Basic and SVG Tiny. The first profile is suitable for higher-level mobile devices such as PDAs, while the second is suitable for lower level mobile devices with very limited resources. The mobile SVG profiles introduce limited content compared to SVG 1.1. To ensure interoperability between content and software tools compliant with different profiles, the SVG Tiny is designed as a proper subset of SVG Basic and SVG Basic as a proper subset of SVG 1.1.

By being an open standard, the interoperability between products from different developers is ensured. The main benefits are lower cost of development and wider scope of usability. Thus, there is no cost of license like in many commercial formats. Since Mobile SVG is a true subset of SVG 1.1, products developed for mobile device will be presented correctly in desktop computers. This is a big advantage as the mobile and desktop world may continue to function in symbiosis. However, to benefit from the fact that Mobile SVG, as well as SVG 1.1, is open standards, the format must have industrial support. Today, support for SVG only exists in a few browsers for the World Wide Web such as Amaya and Batik. Amaya is open source project initiated by W3C including a browser and editor.13 The Batik Squiggle browser uses SVG to describe documents to be rendered in the browser.14 For other browsers, plug-ins are available allowing the browser to display SVG images. The most common plug-in is developed by Adobe.15 Future versions of browsers are most likely to include support for SVG. Mobile SVG is not supported directly by any OS, like Symbian, in mobile devices. However, a number of SVG viewers are available with or without license such as BitFlash

Mobile SVG Player. The 3rd Generation Partnership Project – 3GPP is a collaboration agreement developing technical specifications for third generation mobile system. The 3GPP recently decided that the SVG Tiny profile should be supported in its recent MMS standard.

As mentioned earlier the Mobile SVG consists of two profiles SVG Basic and SVG Tiny, where SVG Tiny is the most limited profile. In Table 1 a comparison of the most important features regarding representation of maps between the two profiles is presented. Evidently, the SVG Basic profile is better suited for representing a map.

<table>
<thead>
<tr>
<th>Features</th>
<th>SVG Basic</th>
<th>SVG Tiny</th>
</tr>
</thead>
<tbody>
<tr>
<td>References</td>
<td>Internal, external</td>
<td>Internal</td>
</tr>
<tr>
<td>Bitmap images</td>
<td>JPEG, PNG</td>
<td>JPEG, PNG</td>
</tr>
<tr>
<td>Styling CSS*</td>
<td>Yes, restricted</td>
<td>No</td>
</tr>
<tr>
<td>Transformations</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>- Matrix, translate, scale, rotate, skew</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Paths</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Basic shapes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Symbol</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Text Unicode</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>- No text selection or clipboard operations</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Text on path</td>
<td>Yes, not on text</td>
<td>No</td>
</tr>
<tr>
<td>Filling, stroking, marker*</td>
<td>Yes</td>
<td>Yes, not on text</td>
</tr>
<tr>
<td>Color*</td>
<td>sRGB etc</td>
<td>sRGB</td>
</tr>
<tr>
<td>Gradient, pattern</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Filter effects</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>- Gaussian blur, opacity etc.</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Interactivity, events</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Linking*</td>
<td>Yes, not within SVG document</td>
<td>Yes</td>
</tr>
<tr>
<td>Scripting</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Animation*</td>
<td>Yes</td>
<td>Yes, not through scripting &amp; DOM</td>
</tr>
<tr>
<td>Metadata</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 1. A comparison between SVG Basic and SVG Tiny. The ‘*’ indicates that the feature is reduced in functionality compared to SVG 1.1.

---

16 BitFlash, BitFlash SVG Viewer & SDK (2003).
17 3GPP, Multimedia Messaging Service (MMS); Media formats and codecs (2003).
4.2.1 Richness of graphical presentation

The SVG document consists of graphics elements and container elements. Graphics elements are corresponding to objects that are to be drawn in the image. Container elements are used for structuring the document, containing graphics elements or other container elements. Graphics elements consist of text, basic shapes, paths, bitmap images and the “use” element. The text is stored as fonts and not as multiple lines describing each glyph. By that, the text is very flexible to position and is also searchable. SVG makes it possible to provide fonts embedded into the content, for text in uncommon or minority languages. These fonts are not installed on the client system and disappear after the content has been viewed. The basic shapes are rectangle, circle, ellipse, line, polyline and polygon. The use of the ellipse element is restricted in SVG Basic. The path element is a flexible tool for creating shapes. The lines between different points can be drawn as elliptical or circular arcs, or cubic Bezier curves. Elements in an SVG document have an implicit drawing order, with the first elements in the SVG document drawn first. Subsequent elements are painted on top of previously painted elements, taking into account opacity settings. Thus, it is possible to hide elements by simply drawing another on top.

The container elements are used to structure the SVG document. It is also possible to attach similar attributes, such as titles and descriptions, to several shape objects. Examples of container elements are an element, “g”, for creating symbols and the element, “a”, used for hyperlinks. The SVG document may be structured in a hierarchical tree defined by a DOM. Scripts, such as Javascript, may be added in order to quickly access and manipulate elements and their attributes. Elements may therefore be added, deleted and re-ordered within the tree. Scripting is an important feature for creating interactivity and animations. The execution of scripts is triggered by events, allowing attributes of the elements to be altered. Animations may be created with scripts. To create an animation, a timer function in the script language is used, making it possible to call functions after certain time intervals. These functions define the behaviour of the animation by for example moving, resizing or hiding objects. The SVG element "animation", which is another technique for creating animations, has inherited its animation functionality from SMIL – Synchronized Multimedia Integration Language.

In addition, SVG Basic provides features typically associated with bitmap images. Such features are filter effects like Gaussian blur, opacity and shadowing.

4.2.2 Possibility to identify objects

An SVG map image consists of elements in the SVG document, which corresponds to objects in the map. In SVG Basic it is possible to identify objects by assigning labels. It is also possible to store metadata of each object. SVG Basic is able to define symbols and link them
to a specific URL for all symbols. Each symbol may be associated with a bitmap image, which can be drawn and distributed in the SVG document. In order to make the transference of SVG document more efficient, it may be preferable to cache frequently used symbols locally in the client. The symbols can be resized, their orientation can be changed and the style can be adjusted to match the rest of the vector image.

### 4.2.3 Possibility of progressive data transmission

SVG is part of the XML family and therefore inherits a few differential properties. As XML, SVG requires that all start tags must be matched with corresponding end tags. An SVG document cannot be validated until the last tag is read. Due to this property of SVG documents, progressive transmission is difficult. The viewer of the SVG document could assume that the content is valid but there would still be problems rendering the content as it is loaded. There is no guarantee in SVG (or in XML for that matter) that an identifier is defined before it is used. An SVG document may "use" an element before it is defined. A text on a path on a map may reference a path that has not been loaded yet. It would be possible to restrict the format converted in the server, that all identifiers must be defined before a reference is passed onto them. Even with these restrictions, the content may not lend itself well to incremental rendering. Objects are rendered from front to back based on the order that they appear in the document. It might be useful to take advantage of this feature by hiding parts of a map with objects drawn in front. If the application in the mobile device naively would draw objects, as they were loaded, it would expose these techniques.\(^{18}\)

### 4.2.4 Size of data, transfer time via wireless networks

SVG is vector based, and for all vector graphics this means that the size of the file does not depend on the size of the image. Instead, the size depends on the complexity of the image. The more elements the image consists of, the bigger the file. The difference between a static map and an interactive map is minimal. An interactive map increases in file size of maybe two percent. However, SVG is still text based making file size consequently larger than a binary counterpart. Also, as an SVG document describes all cartographical data, the file size becomes considerably larger than if only geographical data would be described. But since it is text based, it is suitable for compression with compression algorithms like Zlib.\(^{19}\) The W3C recommends the use of Zlib for compressing the SVG document. The algorithm is licence free and provides loss less compression and cyclic redundancy check value for detecting data corruption. Another compression alternative is the XMill, which is an effective compressor for XML. It should therefore also work SVG. XMill separates the structure and the content in an

---

XML document and uses different compression methods for these two different partitions. Hence, XMill typically achieves much better compression rates than conventional compressors such as gzip. The XMill is free but still licensed invoking some restrictions of derived commercial products.\textsuperscript{20} It is possible to develop custom made compression by defining a protocol for interpreting the SVG tags. The alternative might provide satisfying compression level and allow for less demand on the mobile device while decompressing.

SVG provides other features to limit file-size. One feature is interpolation, which enables automatic construction of in between frames, rather than explicitly stating the contents of each and every frame in an animation. This has a large effect on the size of the content compared to binary alternatives. In addition, interpolation also allows the framerate to be smoothly adjusted for the available computing power, without having to make multiple copies of the content for different devices.\textsuperscript{21} By supporting the use of CSS, rather than traditional styling, also enables the file-size to be smaller. However, the SVG Tiny profile allows for a limited use of CSS. The use of Bezier curves instead of many short line-segments may also limit the file-size, as well as SVG’s support for fonts instead of drawing each glyph.

### 4.2.5 Design of implementation

There are two possible ways to render SVG documents. The two approaches differ where the software handling the conversion from map data to SVG is located. The conversion may be located either on server side or client side. As illustrated in Figure 9 an SVG Gateway in the Graphics Module could carry out the conversion of the data from the MapModule before being delivered to the client. The GUI would be responsible for the rendering of the SVG document. The client side approach is similar. If the client has the necessary capabilities in terms of CPU power and memory there is no reason for the server to handle the conversion process. The client could request the data and perform the conversion locally. This is however not likely due to the limitations of CPU and memory in mobile devices as well as bandwidth of wireless connections. Thus, the server side approach is preferable. A third way is possible by using XML as an intermediate step. It is possible to write a mark up directly inside an SVG document using Namespaces. A scripting language such as JavaScript could handle elements within the intermediate XML document. The JavaScript code would be responsible for formatting and displaying by generating SVG elements dynamically by addressing the DOM of the XML document.\textsuperscript{22} However, the intermediate step is unnecessary if there is no interest in describing the map data in XML for further use in the server.

\textsuperscript{20} XMill, \textit{An Efficient Compressor for XML} (1999).
\textsuperscript{22} Sagar, M.S., \textit{An SVG Browser for XML Languages} (2003).
The architecture of the SVG gateway is divided in the server interfacing with clients and a software module responsible for the conversion, see 8.4.2. The server is handling the request from the user and getting the necessary map data. The software module processes the data by parsing the data, arranging the layout and adding scripts to the resulting SVG document. The layout process refers to arrangement of the most significant objects as well as the selection procedure of which objects to include. Optionally the module compresses the document before returning it to the server. The architecture is illustrated in Figure 10.

Since SVG is text-based, it is easy to use a scripting language that feature good support for manipulating text strings and querying databases, such as PHP, Perl or Python. However, depending on the implementation of the server other solutions might prove to be better suited.
The SVG language offers a large amount of graphical features, many of the not relevant in the representation of a map. When implementing the SVG Viewer in the application the only features necessary to implement is the features needed to represent a map. In many SVG Viewers today, a validator for the SVG document is implemented to determine if the SVG document is valid or well-formed. By removing unnecessary features and moving the validation feature to the server creating the document, allows for the viewer to be more slimmed.

### 4.2.6 Memory and CPU demands in mobile devices

When the pure SVG document is delivered to the mobile device it needs to be cached, in order to be processed by the application. If the document is compressed using Zlib or other compressing algorithm it must be decompressed before being processed. The application is responsible for the rendering of map image on the basis of the SVG document. The rendering is carried out by implementing rendering algorithms or by installing a plug-in. Accordingly, available memory in the mobile device must contain the compressed document as well as the decompressed document, and the rendering algorithm or plug-in.
4.3 RaveGeo™

RaveGeo™, a product developed by Idevio\textsuperscript{23}, is a format for representing geographic spatial data. The format makes it possible to access geographical vector data very quickly in many resolutions. The format is designed to suit distributed environment, however, not explicitly mobile environments. RaveGeo™ consists of three main parts:

- RaveGeo Compiler – a stand-alone application.
- RaveGeo Server – a software that executes in a web server
- RaveGeo Reader – a software module located in a mobile device or a desktop computer.

The product is illustrated in Figure 11. The RaveGeo Compiler reads some of the most common spatial data formats and creates a data set stored as the RaveGeo data format in a database. The RaveGeo data format is the key ingredient in the RaveGeo™ concept. The format is a multi-resolution storage structure for vector-based spatial data. Multi-resolutions of the spatial data are created by a preprocessing task. The multi-resolution process is defined by specifying data generalization parameters in the RaveGeo Compiler. This allows objects in the map to be hidden in low resolutions or be viewed as highly detailed in high resolutions. The multi-resolution feature is a key factor for the streaming ability. The RaveGeo™ vector format is highly compressed due to the generalization features. The RaveGeo Reader extracts and restores the data into an object-oriented format, which is reachable from an API. The vector data can be read by the RaveGeo Reader either from disc or from the RaveGeo Server. The reader is a runtime module and is easy to integrate in applications in desktop computers as well as in mobile devices. It is also possible to integrate the RaveGeo™ in different GIS products using a plug-in.

4.3.1 Richness of graphical presentation

The RaveGeo™ format is first and foremost developed to represent spatial data, not cartographic data. Thus, RaveGeo™ leaves the visualization of the data and rendering process to the application built around it. The content of a map is accessed through objects with appropriate attributes, which are created by the RaveGeo Compiler. The topological relations between the objects are preserved in the format. The objects are described as points, lines and polygons along with attributes. All other graphical features are beyond the scope of RaveGeo™. Instead, the designer of the application is responsible for the graphical presentation as well as rendering the map image. The main advantage is that the amount of data transferred to the application is drastically smaller, than if cartographic data would be included. If the developer wishes to change the appearance of the maps the application must be updated. However, it would be possible to design the rendering software so that the look of each object type could be set by defining colour, line thickness, text font and size. The styling data could easily be transferred in a table describing each object’s styling attributes.

4.3.2 Possibility to identify objects

Each object in the map created by the RaveGeo Compiler is assigned to a unique id, making it possible to link additional information to the object. Other objects such as points of interest, which is beyond the scope of the compiled maps, must be separately handled by the application see 4.3.5.
4.3.3 Possibility of progressive data transmission

The possibility of displaying data while downloading the map is one of RaveGeo’s biggest advantages. When requesting map data, the RaveGeo Server transmits the data in the lowest resolution first. If higher resolution is requested initially, more detailed data of higher resolution is transmitted subsequently. In that way, a low-resolution version of the map may be displayed first, while downloading objects in higher resolutions. As objects in the map are downloaded in higher resolutions, they may be displayed while waiting for even higher resolution data and so on. Thus, the user tends to view a basic map, which gets more and more detailed within seconds.

4.3.4 Size of data, transfer time via wireless networks

The RaveGeo Compiler is responsible for the conversion of spatial data to the object-oriented RaveGeo™ vector format. By avoiding the storage of redundant data and by exploiting similarities in geometries and attributes the RaveGeo™ achieves a high compression rate, typically reducing the file size to about 90 per cent to most common geographic data formats. The compression is lossless and may be further reduced at the cost of resolution. Due to the high compression rate and the ability to extract data in varying resolutions the transfer time is minimized, making the format suitable for transfer via wireless connections with restricted bandwidth. In order to minimize the number of map requests to the server, the RaveGeo Reader provides a non-persistent cache, allowing map data to be stored in between user sessions. Due to the compression rate of RaveGeo™ a large area should be able to fit into the cache depending on the amount of memory dedicated to the application in the mobile device, thus minimizing the amount of request to the server effectively.

4.3.5 Design of implementation

The RaveGeo™ product, as described earlier, consists of three main parts, the RaveGeo Compiler, the RaveGeo Server, and the RaveGeo Reader. The compiler is a stand-alone application responsible for the conversion from map data to the RaveGeo™ vector format. In order to achieve the desired compression rate, the conversion is applied to the entire map database. The conversion process is quite extensive, making it impossible to do real-time conversion of spatial data. The RaveGeo Compiler is optimized for fast access and compression of large amount a static data sets, such as spatial data, with updating frequency of weeks or months. The conversion process should therefore be performed prior to the use of the data. Data sets that are updated more often, such as route data and points of interest, should be separated from the scope of RaveGeo™. The size of such data is typically smaller than the map data, making it possible to transfer separately. By separating the data sets, better module mapping is achieved resulting in less inter-dependency among modules. The
separation of modules also enables an easier integration process. However, a new module, i.e. the RaveGeo Compiler, must be added to the system adding to the complexity of the system.

The RaveGeo Server is responsible for gathering data from the RaveGeo database and providing the data requested to the application. In the RaveGeo™ product, the RaveGeo Server is executed in a web server. However, the appearance of the server is flexible.

The RaveGeo Reader, integrated in the application located in the mobile device, is managing the extraction and structuring of data. Since the rendering is excluded in the reader, such functionality must be added in the application.

Since RaveGeo™ is a commercial product the integration in a location-based service system raises economical issues in terms of licence. This is a disadvantage compared to using an open standard. However, the specific characteristics of RaveGeo™, such as heavy compression and streaming ability, might not be provided by other formats.

4.3.6 Memory and CPU demands in mobile devices

The RaveGeo Reader is responsible for extracting and restoring the map data in an object-oriented data set. The reader, as a software module, is integrated in the application and requires 160 kB of memory space. Depending on the number of features the module could be slimmed down 120 kB. The RaveGeo Reader also allows map objects to be cached in the mobile device, in order to minimize the number of request to the server. The size of the cache is not persistent, allowing the user to match the cache with available amount of memory. As mentioned before the RaveGeo Reader is not responsible for rendering the image. In this way the designer of the application may develop the rendering algorithms suitable for the current platform adding necessary intelligence for cartographic features. The developer of the application may add intelligence corresponding to the amount of memory available.
4.4 SlimMap

The SlimMap vector format is developed by Wayfinder Systems AB for use in their location-based services. The format is designed for transferring map images via wireless networks with limited bandwidth, such as GSM Data. More specifically, the format is designed to represent maps with data transfer time limit of one second regardless of the transfer rate of the connection. Due to this restriction, the map is intended to be viewable in a mobile device, and not in devices offering large displays.

4.4.1 Richness of graphical presentation

The SlimMap format contains not only information of geographical data, but also how the data should be presented. Features that are available are defined by a protocol describing how the application should render the features and in which colours. Examples of features and their supposed interpretation by the application are illustrated in Table 2. The text feature allows for text to be aligned with other features. However, the text may only be presented in a straight line in an angle, and not aligned to an arbitrary polygon. Points of interest such as hospitals, gas stations and schools may also be represented as a feature. A symbol in the shape of a bitmap image may be associated with a point of interest feature. The symbol image must be transferred beyond the scope of the SlimMap format and should be cached locally in the mobile device to minimize data transfer.

<table>
<thead>
<tr>
<th>Features</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thin line</td>
<td>1 pxl thick</td>
</tr>
<tr>
<td>Medium line</td>
<td>2 pxl thick</td>
</tr>
<tr>
<td>Bold line</td>
<td>3 pxl thick</td>
</tr>
<tr>
<td>Dotted thin line</td>
<td>1 pxl thick, 1 pxl filled and 2 pxl gap</td>
</tr>
<tr>
<td>Dotted medium line</td>
<td>2 pxl thick, 1 pxl filled and 2 pxl gap</td>
</tr>
<tr>
<td>Dotted bold line</td>
<td>3 pxl thick, 1 pxl filled and 2 pxl gap</td>
</tr>
<tr>
<td>Thin line polygon</td>
<td>1 pxl thick, unclosed polygon</td>
</tr>
<tr>
<td>Medium line polygon</td>
<td>2 pxl thick, unclosed polygon</td>
</tr>
<tr>
<td>Bold line polygon</td>
<td>3 pxl thick, unclosed polygon</td>
</tr>
<tr>
<td>Circle</td>
<td>Empty circle</td>
</tr>
<tr>
<td>Rectangle</td>
<td>Empty rectangle</td>
</tr>
<tr>
<td>Tiny text</td>
<td>8 pt text</td>
</tr>
<tr>
<td>Medium text</td>
<td>12 pt text</td>
</tr>
<tr>
<td>Big text</td>
<td>16 pt text</td>
</tr>
<tr>
<td>Route</td>
<td>4 pxl line open</td>
</tr>
</tbody>
</table>

Table 2. Examples of features available in the SlimMap format.
A change in the SlimMap protocol forces an update in the application. It should however be possible to change the appearance of a map image if the objects in the map are represented as different features. By stating which features an object is represented by in the server, the application could render the maps differently.

When a map is requested in the SlimMap format, the coordinates of the map is sent to the server. Due to the restrictions in size all objects in the map probably cannot be included in the map transferred to the application. Thus, a selection of prioritized objects must be made. Objects to be included first are city limits, coastline, lakes, forests, parks and so on. Objects may only be included if they are big enough. The criterions are stated in Table 3 and are defined as a percentage of the map area. To determine which streets and roads to include, a percentage of the map image is not suitable. Instead, properties like road class and speed limit are taken into consideration. When creating the map image objects are included until the limit in file size is reached. If the last object is not completely included it should be removed.

<table>
<thead>
<tr>
<th>Features</th>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coastline</td>
<td>&gt; 5 % of the map area</td>
</tr>
<tr>
<td>City limits</td>
<td>&gt; 5 % of the map area</td>
</tr>
<tr>
<td>Forest</td>
<td>&gt; 5 % of the map area</td>
</tr>
<tr>
<td>Lake</td>
<td>&gt; 5 % of the map area</td>
</tr>
<tr>
<td>Streets, roads</td>
<td>Sorted by road class, speed limit etc.</td>
</tr>
</tbody>
</table>

Table 3. Criterions for including objects in map image in the SlimMap format.

### 4.4.2 Possibility to identify objects

The SlimMap format is designed as simple format in order to not add unnecessary intelligence in neither the client application nor the server. The graphical representation of objects in the map consists of dots, lines polygons and text. The format is object-oriented in the sense that it is possible to extract data in an object-oriented manner. However, there is no unique id attached to the objects and therefore the format does not offer the possibility to identify objects. However, it may be possible to define an area containing an object within the application.\(^{24}\) It is not possible to link additional data stored in a database to each object.

### 4.4.3 Possibility of progressive data transmission

The SlimMap format does not offer streaming abilities. The total data transfer time for a map is one second and additional data is not supported. Due to the implementation of the format

---

\(^{24}\) Such a feature would correspond to the `map`-operation in HTML.
each request to the server is stateless, i.e. the server does not keep track of data already sent to the application. Complimenting data cannot be transferred in order to enhance the appearance of the map.

### 4.4.4 Size of data, transfer time via wireless networks

The size of data of a map image in the SlimMap format is limited due to the restriction that the transfer time cannot exceed one second regardless of transfer rate. Thus, the size of the map image depends on the available wireless connection. The number of bytes allowed to represent a map in the SlimMap format can easily be calculated from the data transfer rate of the wireless connection. Different types of connections and corresponding transfer rates are presented in Table 4.

<table>
<thead>
<tr>
<th>Connection type</th>
<th>Data transfer rate [bits/s]</th>
<th>Map size [kB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSM Data</td>
<td>9600</td>
<td>1</td>
</tr>
<tr>
<td>HSCSD</td>
<td>28800</td>
<td>3</td>
</tr>
<tr>
<td>GPRS</td>
<td>53400</td>
<td>5.5</td>
</tr>
<tr>
<td>Bluetooth</td>
<td>108800</td>
<td>11</td>
</tr>
<tr>
<td>UMTS</td>
<td>384000</td>
<td>46</td>
</tr>
</tbody>
</table>

*Table 4. Data transfer rates of wireless connections and allowed size of image in SlimMap format.*

In order to further minimize the amount of data, it is possible to store coordinates in fewer bits at the prize of lower precision. Depending on the size of the display in the mobile device, this may be acceptable.

### 4.4.5 Design of implementation

The software associated with the SlimMap format consist of conversion and selection algorithms located in the server, and rendering algorithms located in the GUI of the application.

### 4.4.6 Memory and CPU demands in mobile devices

The memory requirements in the mobile device set by the SlimMap format is relatively insignificant. The size of the map image is between 1 – 46 kB depending on the transfer rate of the wireless network. The CPU demand of rendering a SlimMap image depends on the rendering abilities of the operating system.
4.5 GfxFeatureMap

The GfxFeatureMap is developed and used internally by Wayfinder Systems AB. The format is a protocol for representing maps in binary data. The format is not specifically developed for transfer via slow wireless networks, but as it is a binary format the field of its application may be expanded. Since the GfxFeatureMap is used internally at Wayfinder Systems AB, a thorough description may not be provided. The evaluation is given in Appendix A.
4.6 MapTP™

MapTP™ is a product developed by NETSOLUT located in Germany. MapTP™, short for Map Transfer Protocol, enables multi-resolution maps transferred to applications in distributed and mobile environments. It is a client/server software platform ready to be integrated in software and hardware architectures. The product is compliant and may be used in combination with other software and standards such as RDBM, XML, SVG, SOAP and many GIS systems and GIS formats. This enables MapTP™ to easily be integrated with a large range of data types customized to specific needs.

The MapTP™ product consists of three main components:

- MapTP – a vector format, or protocol, for representing geographical data.
- MapTP Geocoding – provides fuzzy address and incomplete address matching etc.
- MapTP Routing – provides complex itineraries including road signs, traffic circles etc.

The component of interest in this evaluation is the MapTP vector format. Spatial data is converted into multi-resolutions in the MapTP Server by a static conversion process. The process uses input data from most common geographical data formats. However, the MapTP Server also supports import of any third party vector data. The server offers a number of external APIs such as XML, PHP, and HTTP in order to provide flexibility when accessing the data. When requested, data is fetched from the server in the MapTP vector format. The MapTP vector data is transferred to the MapTP Reader in incremental packages, which allows the application to process the data immediately. A systematic description of the product is given in Figure 12.

---

The MapTP™ platform may also be described as layers in an OSI fashion. A description of the layers is provided in Table 5 and illustrated in Figure 13. In order to integrate third party data the MapTP™ platform includes an agent framework. The framework uses a number of internal protocols, such as DMEPTP, ServiceTP, ChannelTP and CGITP, depending on types of services and interfaces. On top of the OSI hierarchy is the LBS application.

<table>
<thead>
<tr>
<th>Layers</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSession</td>
<td>Provides functionality for connecting to the MapTP Server using underlying network transport protocols, like TCP/IP.</td>
</tr>
<tr>
<td>MapTP</td>
<td>MapTP is responsible for retrieving the map.</td>
</tr>
<tr>
<td>DMEPTP</td>
<td>This protocol is used for sending and modifying maps to and on the server.</td>
</tr>
<tr>
<td>ServiceTP</td>
<td>This protocol is used for attaching distributed spatial services, such as routing services, to the MapTP system.</td>
</tr>
<tr>
<td>ChannelTP</td>
<td>This protocol is used for sending highly dynamic or even real-time data, such as traffic information, and pushing it instantly to mapping clients.</td>
</tr>
<tr>
<td>CGITP</td>
<td>This protocol is used to quickly process huge amounts of HTTP requests, for example XML and SOAP data.</td>
</tr>
<tr>
<td>MapTP DC</td>
<td>MapTP DC contains all that is needed for drawing maps.</td>
</tr>
<tr>
<td>MapTP Reader</td>
<td>Responsible for requesting maps and structuring the data set.</td>
</tr>
<tr>
<td>Agent Framework</td>
<td>Classes MapTP Agents that integrate spatial data, services and algorithms into the system.</td>
</tr>
<tr>
<td>Application</td>
<td>Customized application.</td>
</tr>
</tbody>
</table>

Table 5. Description of the layers in MapTP™ platform.
4 Evaluation of formats of vector graphics

Figure 13. Illustration of the layers in MapTP™ platform.

4.6.1 Richness of graphical presentation

The MapTP module located in the application allows the developer to access all objects in the map through a complete object-oriented data structure of the map. The objects are represented as dots, lines and polygons. The module includes displaying methods in the API, but they may be overridden allowing the developer to take control of the complete look and feel of the map. However, it is also possible to modify resource files in order to change the appearance of the map without any updates of the MapTP client module in the application.

The provided API of the client module of MapTP™ enables the use of user interactions by supporting events. Animations may be triggered while downloading data in order to minimize the waiting for data versus viewing data ratio. Since all communication with the server is performed in the background, the downloading of data is transparent to the user by displaying animations, such as head up/north up rotation of the map, or other features in the foreground.

4.6.2 Possibility to identify objects

The MapTP™ format allows for an object-oriented data structure of a map. Each object is associated with a unique id and may contain generic object descriptions for presentation in the map. It is also possible to attach symbols to a set of objects with similar statistics or group objects in layers. By supporting the development and use of “agents”, MapTP™ also allows linking objects in the map to internal as well as external databases. This technique allows the MapTP™ on one hand to connect to external data sources using any API, and on the other hand to connect to the MapTP server in order to supply the data to the application.
4.6.3 Possibility of progressive data transmission

Instead of relying on isolated vector extractions, MapTP™ uses a complex vector protocol to send a stream of incremental updates to the map client software. All communication is performed in the background, allowing the application to display any incremental update of the map immediately. The server uses session control to establish the appropriate incremental data and minimizes any redundancy in map data.

4.6.4 Size of data, transfer time via wireless networks

The MapTP™ offers a high compression rate using a set of compression algorithms specific for vector data. The compression is applied when data is extracted from the MapTP database. The size of the data transferred to the application is also minimized by the use of intelligent session management of the MapTP Server. The session management eliminates redundancy of data up to 99 per cent. The redundancy check is possible as the server keeps track of which state the client currently is in.

In order to meet the demands of wireless environments, NETSOLUT has developed a MapTP mobile API. The mobile API is designed to minimize memory and CPU demands as well as demands created by slow wireless connections. Thus, the mobile API reduces the data throughput by twenty per cent compared to the traditional API. Still it is fully compliant with the MapTP protocol. The client module designed for mobile devices offers persistent map storage, enabling offline work. The cache size may be configured in order to match available memory in the mobile device. However, when additional data is needed the module also provides a seamless switch between online/offline modes.

4.6.5 Design of implementation

The MapTP™ platform consists of both client and server software. The server consists of the MapTP database where multi-resolution spatial data is stored in the MapTP vector format. The server performs static conversion of input geospatial vector data, making real-time conversion impossible. Since MapTP™ offers agent framework, route data may be included within the scope of MapTP™. By using a real-time push channel, ChannelTP, additional data such as route data or fleet management data may be pushed to the application. The MapTP Reader is integrated in the application, and manages the structuring of the object-oriented data structure of the map. The application is responsible for rendering the map image using the provided or overridden display methods.

The MapTP™ platform is a commercial product, which requires the software to be licensed if integrated into a commercial LBS. Thus, economical issues must be added in the evaluation.
4.6.6 Memory and CPU demands in mobile devices

A footprint of the MapTP module in the application is 300 kB, with a memory consumption of around one megabyte or more depending on the data cache size. The cache containing map data may be configured in order to match available memory. However, the use of the extent of the cache may reduce the number of map requests to the server, i.e. minimizing the waiting time for the user as well as enhancing the performance of the server. The mobile API of the MapTP is designed to demand less of memory and CPU. The developer of the application may choose to discard the rendering methods provided by the MapTP client module, and choose a more efficient implementation. Thus, the memory consumption may be further decreased.
4.7 Comparison & summary

In this section a comparison between the different vector formats is presented. As some criterions are difficult to compare, an effort to quantify the criterions is made. Each criterion is quantified in the ranged from one to five, where five denotes the ability to completely fulfil the criterion.

A few interesting aspects regarding limitations in comparing the formats on the basis of the defined criterions is worth mentioning. It is clear that the criterions are not equally weighted in the comparison. For instance, the lack of ability to identify objects makes a format not desirable although it might be preferable when comparing other criterions. SVG, RaveGeo™ and MapTP™ are the only present formats that offer object identification.

The size of data becomes less critical if progressive transmission is possible. The streaming feature allows for short transfer time initially and still offers large amount of data to be transferred. RaveGeo™ and MapTP™ are the only formats that offer the streaming feature based on multi-resolution transmission. GfxFeatureMap offers the ability to stream content in correct drawing order. The GfxFeatureMap is object-oriented and it is possible to design the rendering software to draw each feature as soon as it is downloaded. The streaming feature in multi-resolutions results in a more complex implementation of the format, and therefore also a more complex integration. The SVG and SlimMap are less complex but lack the advantage of streaming.

The graphical richness of the format also becomes less critical only if the format allows for the data to be stylable without resulting in an upgrade in the application software. Depending on the way the styling attributes are represented in the format, the application may interpret the cartographic features. This would result in smaller amount of data to be transferred to the application. The SVG format and RaveGeo™ format are two extremities in offering cartographic data. On one hand, SVG offers many cartographic features, many of them irrelevant when representing a map. On the other hand, RaveGeo™ offers no cartographic data at all leaving such features to the application displaying the map. The SlimMap and GfxFeatureMap both offer some cartographic data but may be configured to allow the appearance of the maps to be determined in the server.

One interesting observation is that the selection process is beyond the scope of SVG and data redundancy processes are beyond the scope of all formats except the RaveGeo™ and MapTP™ format. Thus, the size of data criterion is not ideal when comparing the formats. It is quite possible to design such algorithms and apply them to data prior to the conversion to the vector formats. A more interesting measure would be the amount of overhead in the map
format. It is obvious that the SVG format offers the largest amount of overhead since it is text based. The SlimMap offers a selection process where the number of features is restricted by the predetermined file size. It is quite clear that the heavy restriction on the data size of the SlimMap format results in too heavy limitations in desirable features of a vector format. The GfxFeatureMap also requires a selection process of the geographic data in order to limit the file size. But neither SlimMap nor GfxFeatureMap offers data redundancy check. This process typically uses large amount of data to identify and reduce redundant data. The reduction of redundant data is a time consuming task, which makes it inappropriate in real-time systems. Thus, the RaveGeo™ and MapTP™ formats rely on pre-processed datasets, resulting in an additional database and hardware resource. This adds to the complexity of the system and may raise maintenance costs. In addition, the server in the MapTP™ platform performs real-time redundancy checks by creating sessions for each user. The use of sessions is not preferable as seen in 9.2. The utilization of the server decreases drastically, compared to a state-less server.

The qualitative properties of each format are quantified according to the discussion above, to enable a comparison between the different formats and their features. The result is presented in Table 6. The size property is determined by representing a similar geographical area in each vector format. In RaveGeo™ and MapTP™ the data size is estimated based on the number of coordinates in the geographical area and the compression rate. It is quite clear from the comparison that MapTP™ and RaveGeo™ are the most sophisticated vector formats and are designed and best suited to represent spatial data in mobile environments. The MapTP™ offers a great variety of integration interfaces, while the RaveGeo™ provides a more slimmed interface, yet enhances customizations.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SVG Basic</td>
</tr>
<tr>
<td>Graphical richness</td>
<td>5</td>
</tr>
<tr>
<td>Identify objects</td>
<td>Yes</td>
</tr>
<tr>
<td>Progressive trans.</td>
<td>No</td>
</tr>
<tr>
<td>Size of data</td>
<td>70kB*</td>
</tr>
<tr>
<td>Transfer time</td>
<td>1</td>
</tr>
<tr>
<td>Memory, CPU m.d.</td>
<td>4</td>
</tr>
<tr>
<td>Real-time extraction</td>
<td>Yes</td>
</tr>
</tbody>
</table>

*Table 6. Comparison between SVG Basic, RaveGeo™, SlimMap, and MapTP™. The ‘*’ means that data is compressed with Zlib. The GfxFeatureMap is left out due to risk of compromising the business case of Wayfinder System AB.*
The chapter presents a number of design proposals derived from the evaluation of existing vector formats in previous chapter. The design proposals present areas of improvement and are implementation-ready proposals.

In the execution of the first activity presented in chapter 4, some interesting properties have been observed. In summary, a number of design proposals of a vector format in order to fulfill the defined criterions have been derived. In addition to the evaluation, existing research in a number of fields identified as relevant to meet the requirements of the study are investigated. The research is applied in the derivation of the design proposals. The proposals are easy to translate into any LBS system, which incorporates vector maps, and provide a more straightforward summary of the demands on the desired format. The design proposals are:

- Object identification.
- Progressive vector transmission.
- Generalization of geospatial data.
- Coordinate representation.
- Spatial database storage.
- Semantic caching of received data.
- Threaded data processing.
- Compression techniques.
- Cartographic appearance.
- Design approach.

Each proposal will be further explained in the following sections.

5.1 Object identification

A format, which does not offer the ability to identify objects in the map, cannot be a contestant. The identification is needed to link additional external data to each object in the map. It might be desirable to, for example, enable a user to click on the location of a company and receive information like phone number, address or URL to its website. Since a map may contain unlimited amount of data, it is preferred to group different objects with similar
statistics in *layers*. Objects may be railroads, hotels and gas stations. Each user of an map application might want to visualize different objects by choosing layers. Certain objects, like hospitals, should be able to be presented as symbols. It should also be possible to align text to roads and paths in other directions than vertically or horizontally. For example, it is desirable to align text with irregular shapes like rivers. Objects should therefore be represented as *abstract cell complexes*. A geospatial map is presented as a sequence of layers \((l_1, ..., l_n)\), where each layer is associated with a set of attributes. Objects in the map own a unique identifier \(o_i\) and each object belongs to one and only one layer as part its domain. A number of values \(v = (v_1, ..., v_j)\) accessed through a set of attributes \(a = (a_1, ..., a_j)\) are attached to each objects.\(^{26}\) Each object may consist of one or more spatial entities \(g = (g_1, ..., g_k)\) described as *abstract cell complex* of three orders embedded in the Euclidean plane, with topological relations.\(^{27}\) Points are entities of order 0, lines of order 1, and regions of order 2. A line can be either open or closed, and is described by a sequence of vertices. Extreme vertices in the sequence, called endpoints, form its boundary. If the line is closed, the boundary is made up of a single point. Regions can have holes, cuts, and punctures. An instance of a map consists of a set of layers \(l\) with attaching objects \(o\) defined by \((a, v, g)\). For example, a lake could be represented by \(a = \{\text{id}, \text{name}\}\), \(v = \{5, \text{Väner}\}\), and \(g\) being a region defined by a number of coordinates in topological order as well as topological relations to other objects such as bounded to road etc. The objects make up the contents of a map, and are datastructures to be transferred to any application.

Object identification is also required to enable data of higher resolution to be added to each object in the application when zooming etc. Since spatial data is not completely static, it may be necessary to change an object’s id as it is modified. Thus, the server must provide a check for the validation of cached data in the client application upon session start. If validation fails, the application is urged to clear its cache. To avoid the risk of escalating number of ids, the uniqueness could be limited to layers or even submaps. If each object is assigned a unique id the ids will need to be represented in a large number of bits, since the spatial data may consist of millions of objects. The overhead of transferring the ids would be tremendous; therefore ids must be allowed to be reused.

### 5.2 Progressive vector transmission

Progressive transmission of bitmap images for visualization of data has successfully been used on the World Wide Web by providing the viewer with coarser versions before the bitmap image is completely downloaded. However, in the vector domain progressive transmission is somewhat more delicate. The idea is to display downloaded objects in the map while

---


downloading the remaining parts of the map, i.e. progressive vector transmission. Traditionally, transmission of vector data has generally been done by means of a one-step long process. However, when large datasets are accessed remotely or transmitted across connections with limited transfer rate, progressive vector transmission is needed. Each packet can be processed by the client application instantly, allowing a user to get an immediate visual feedback even on slow connections. The progressive transmission can be executed in terms of drawing order and/or resolution order. As seen in the evaluation of the RaveGeo™ and MapTP™ formats progressive transmission is possible with impressive results. Progressive vector transmission can be accomplished by adopting two different approaches presented below. The latter one is more sophisticated than the first.

### 5.2.1 Progressive transmission of vector data in predefined zoom-levels

The progressive transmission in predefined zoom-levels refers to transmitting vector data suitable for display at a certain zoom-level. The amount of objects in the map is selected according to predefined rules for cartographic visualization (certain types of objects are excluded in higher zoom-levels). The progressive transmission refers to the transmission of objects in a specific order, i.e. the most significant objects is transmitted first allowing them to be visualized first see 5.7.

### 5.2.2 Progressive transmission of vector data in predefined resolutions

A more sophisticated approach is the progressive transmission of vector data in predefined resolutions. When requesting a map image lower resolution data providing a rough outline of the map is delivered first, allowing the viewer to interpret the map immediately. If higher resolution is requested, more detailed data is transferred providing a more detailed image sequentially. The progressive transmission is illustrated in Figure 14. When zooming the map each object in the map is scaled until a higher resolution version of the object is available. Thus, the map image may be updated instantly allowing the user to get an immediate feedback on the zooming.

When stored in spatial databases, geospatial data is traditionally captured at a certain resolution and accuracy. Resolution refers to amount of spatial entities and the information of such entities, i.e. their geometry, and accuracy refers to the precise approximation of the extent and location of an entry. Variations of resolutions may change the topological order, while variations of accuracy may not. The point of separating these metric and topological aspects is the clarification that these properties may coexist in a single map. Critical entities in a map, e.g. roads in a roadmap, may be presented in high-accuracy while other entries may

---

be presented in low-resolution, e.g. lakes. By merely focusing on visualization of geospatial data the accuracy aspect becomes less important, as long as the approximation error is kept within an acceptable tolerance. It is important not to think of resolution as zoom level, however they are correlated. A high zoom level usually requires high resolution. However, when visualizing geospatial data the stored level of detail is often much too high, i.e. data is too large, for its purpose. Another disadvantage of single-resolution data is that if the user does not zoom the detailed data is wasted, and thus unnecessary data is downloaded. In order to reduce the amount of data needed by the application, thus reducing the transfer time and processing effort in the mobile device, methods called generalization or simplification may be used see 5.3. By generalizing the data in quantum steps, multi-resolution data is achieved.

Let $\mathbf{M} = \{M_i | i = 0, \ldots, n\}$ denote the family of maps in multi-resolution sequence, where $M_0$ is the map with maximum resolution, and let $\mathbf{F} = \{F_i | i = 0, \ldots, n\}$ denote the corresponding simplifications in a sequence of tolerances $\epsilon_0 < \epsilon_1 < \ldots < \epsilon_n$. Then the pair $(\mathbf{M}, \mathbf{F})$ is called a layered multi-resolution model. Creating a multi-resolution model is not a trivial process. Distinctions must be made between topological and metric aspects to ensure constancy in both domains. The progressive transmission of vector data in multi-resolutions, illustrated in Figure 14, presents data handling on the server side as well as on client side. On the server side, methods for building, manipulating, and transmitting a sequence of representations at different resolution levels must be implemented. To build resolutions require a preprocessing task not suited for real-time processing. On the client side, functionality to integrate and visualize the transmitted resolution levels must be implemented.

![Figure 14. Resolutions of the spatial data transmitted to the client in order of increasing detail.](source)

5.3 Generalization of geospatial data

Wireless connections offer limited transfer rates compared to broadband connections available to desktop computers. Datasets of geospatial data are usually quite large and methods to limit and/or reduce the size need to be investigated in order to meet the limitations

---


of the mobile environment. A simple and natural approach is to select small amounts of data to transfer to the mobile device. The downside to this is that either content is less compelling or lasts for a very short time. In addition, each request for data comes with a certain amount of overhead. Thus, the ratio of overhead and useful geospatial data being transferred would increase and decrease the efficiency. In addition, smaller data packages would result in increased number of accesses to the server via the wireless network. The purpose of generalization of geospatial data is to obtain a graphical representation of a map suitable for interpretation of a user. Generalization aims to organize and select the proper amount and kind of data to facilitate a clear visualization. Generalization also reduces the amount of data representing a map. This is extremely important as the size is correlated to the transfer time to the application. The wireless connection is identified as a bottleneck in 9.2. As defined in 2.1.2 a map consists of spatial entities described as abstract cell complex of three orders embedded in the Euclidean plane. Points are entities of order 0, lines of order 1, and regions of order 2. Objects in the map may consist of one or more entities with topological relations. Resolution refers to amount of spatial entities and the information of such entities, i.e. their geometry. However, the two maps must be consistent in their spatial relationships. Objects that meet in the high-resolution map may not be disjoint in the low-resolution map. As the main purpose of the vector format is to represent the spatial data visually, the level of accuracy is not as demanding as if the data would be used for spatial computation. Three possible generalizations of an entity exist:31

- Preservation: the entity still carries its characteristics and order but with simplified features.
- Reduction: the object is represented in lower order, such as a single point represents a region.
- Immersion: the object disappears, i.e. it is immersed in a larger object.

These generalizations functions operate on entities and may be used to create map generalization operations, \( F \). Since this entity generalization functions are atomic and continuous, a composition of several functions (resulting in map generalization operations) is also continuous. As a result, all map generalizations can be obtained as a sequence of atomic entity generalization functions.32 However, in order to perform map generalization operations the description of spatial entities as abstract cell complex must be increased to include topological relations. Entities can be related to each other by the following relations: disjoint, meet, contains, covers, equal, overlap, inside, covered by, bounds, and bounded by.33

topological relations among objects may be summoned in a single map to include a subset of situations:

- Two regions are either disjoint or they meet at a common boundary.
- Given a line $l$ and a region $r$ following relations are possible: $l$ and $r$ are disjoint; $l$ and $r$ meet at a single point $p$; $l$ bounds $r$; $l$ is inside $r$; $l$ is covered by $r$ ($l$ is inside but one endpoint bounds $r$).
- Given a point $p$ and a region $r$ following relations are possible: $p$ and $r$ are disjoint; $p$ bounds $r$; $p$ is inside $r$.
- Two lines are either disjoint, or they meet at a common endpoint.
- Given a point $p$ and a line $l$ are either disjoint, or $p$ bounds $l$.
- Two points are disjoint.

### 5.3.1 Map generalization

As mentioned in previous section, map generalization operations can be defined from combining entity simplification and topological relations. Map generalization refers to six operations: simplification, selection, symbolization, exaggeration, displacement, and aggregation.\(^{34}\) The operations can be divided into model-oriented generalization and cartographic generalization. The model-oriented generalization includes simplification, selection, symbolization, and aggregation. Displacement, aggregation, and additional simplification belong to cartographic generalization.

Simplification is the most common simplification operation and the easiest to understand. Simplification results in objects with simplified features. This can be accomplished by polygon and polyline simplification. The simplification process adds to the complexity of the extraction of data in the server but more or less sophisticated algorithms may be used. The idea of using multi-resolution data is that when zooming low-resolution data can be used to immediately display the rough outline of the map, while higher resolution data is being downloaded. The filtering process of geospatial data should preferably be performed in real-time. However, the complexity of the process might prove that a more static simplification process is necessary. Instead of actually requesting data and performing simplification, the server would only need to query a database responsible for storing the multi-resolution data see 5.5. Still it might be possible to implement a less complex simplification process to be performed “on the fly”. In addition, it would be most preferable to filter data that has already been transmitted to the client application in order to eliminate redundancy. That is, the filtering process would need to receive input if lower resolution data of a certain area already has been transmitted to the client.

\(^{34}\) Ibid.
A fair amount of research has been done and has lead to a number of reliable simplification algorithms. The most simple algorithm referred to as *vertex reduction* is of immediate evidence. It simply discards a vertex along a polyline that has a distance less than some minimum tolerance $\varepsilon > 0$, from a prior initial vertex. When no vertex fulfills this condition then the algorithm continues the iteration. The tolerance $\varepsilon$ may be set so that no vertex will be rendered to the same pixel on the display, or any other acceptable error e.g. a percentage of the scale level. The algorithm is illustrated in Figure 15. The algorithm must traverse all vertices resulting in a worst-case complexity of $O(n)$. Due to the simplicity of the vertex reduction algorithm, it might not be ideal for polyline simplification. However, in a real-time filtering process it might prove to be acceptable. In a static filtering process it would be suitable as a preprocessing stage.

A more complex algorithm of polyline simplification is the well-known *Douglas-Peucker algorithm*. Whereas the vertex reduction algorithm uses closeness of vertices as a rejection criterion, the Douglas-Peucker algorithm the closeness of a vertex to an edge segment. The algorithm starts with a crude initial guess at a simplified polyline, namely the single edge joining the first and last vertices of the polyline. Then the remaining vertices are tested for closeness to that edge. If there are vertices further than a specified tolerance, $\varepsilon > 0$, away from the edge, then the vertex furthest from it is added in the simplified polyline. This creates a new guess for the simplified polyline. Using recursion, this process continues for each edge of the current guess until all vertices of the original polyline are within tolerance of the simplification.\(^{35}\) The selection condition is presented in Figure 16.

More specifically, the two extreme endpoints of a polyline are connected with a straight line as the initial rough approximation of the polyline. Then, how well it approximates the whole polyline is determined by computing the distances from all intermediate polyline vertices to the finite line segment. If all these distances are less than the specified tolerance $\varepsilon$, then the approximation is good, the endpoints are retained, and the other vertices are eliminated. However, if any of these distances exceeds the $\varepsilon$ tolerance, then the approximation is not good enough. In this case, a new point that is furthest away is chosen as a new vertex subdividing the original polyline into two shorter polylines, as illustrated in Figure 17. The complexity of the algorithm is known to be in worst-case $O(n^2)$ and best-case $O(n \log n)$ when $\varepsilon$ is small.\textsuperscript{36}

\textsuperscript{36} Ibid.
The Hershberger-Snoeyik algorithm is an attempt to improve the complexity of the Douglas-Peucker algorithm by a so-called convex hull speed-up. The algorithm speeds up the selection of the farthest intermediate vertex from the initial vertex by noting that the farthest vertex must be on the convex hull of the polyline chain from $V_i$ to $V_j$, see Figure 18. The hull is computed in $O(n)$ time using Melkman’s algorithm for the hull of a simple polyline. The vertex farthest from startvertex can be found by using an $O(\log n)$ binary search on the hull vertices. The hull information is reused when a polyline chain is split at the farthest vertex. The algorithm is complex to implement but in the final analysis it is shown to have worst-case complexity of $O(n \log n)$. However, the algorithm introduces a few restrictions at the cost of higher complexity. First, it depends on using the Melkman $O(n)$ time hull algorithm, and thus the polyline must be simple, i.e. unclosed. Second, since the algorithm depends on a two dimensional convex hull algorithm, it only applies to planar polylines, whereas the prior
The mentioned algorithms can be used for any dimension.\textsuperscript{37} This is however quite acceptable in two-dimensional map applications. The algorithm was implemented in the system at Wayfinder Systems AB with impressive results. However, the simplification requires that geometric data is separated from topological data. Otherwise, inconsistency may appear in the map. The inconsistencies may be avoided by allowing endpoints of the polyline to represent topological connections. For example, a t-crossing may be represented by three polylines which connect in a single point. The possible inconsistency is avoided since endpoints are always left untouched by the simplification algorithms. One disadvantage is that the number of vertices reduced is smaller if the polyline contains a small amount of vertices.

![Figure 18](image.png)

**Figure 18.** Illustration of the hull of a polyline. Hull vertex indeces are stored and reused in the algorithm.

Research as been done in attempt to further simplify polyline simplification algorithms. Hershberger & Snoeyik questions whether there are practical, linear-time solutions to this problem.\textsuperscript{38} In that case a static simplification process may be the only solution in order to further minimize response time when requesting map data. Table 7 presents the complexity and the suggested filtering process type.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Complexity</th>
<th>Filtering process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertex reduction</td>
<td>$O(n)$</td>
<td>Real-time</td>
</tr>
<tr>
<td>Douglas-Peucker</td>
<td>$O(n^2)$</td>
<td>Static</td>
</tr>
<tr>
<td>Hershberger-Snoeyik</td>
<td>$O(n \log n)$</td>
<td>Static/real-time</td>
</tr>
</tbody>
</table>

Table 7. Complexity in the ordo-notation for polyline simplification algorithms.

These polyline simplification techniques require non-cyclic polylines, whereas regions cannot be simplified. It would be possible to cut a region in a number of polylines. The start point, endpoint, and points relevant to the preservation of topology must however be untouched to obtain consistency in the region.\textsuperscript{39}


\textsuperscript{38} Hershberger, J. & Snoeyik, J., *Cartographic Line Simplification and Polygon CSG Formulae in O(n log*n) Time* (1998).

\textsuperscript{39} See resemblance to Splines, which demands of continuity in connecting vertices.
The selection operation is the second map generalization operation, which refers to the elimination of entities considered irrelevant of cartographic means. This operation affects the topology of a map and must listen to the topological relations of entities. Symbolisation refers to the case when an object is reduced in characteristics. Instead of being represented as its actual size, the object is represented by a suitable symbol, e.g. a line is reduced to a point. The exaggeration operation attempts to preserve and exaggerate objects or entities that would be too small for the context of a map, but are semantically important. An example is the exaggeration of road classes in a road map. The exaggeration operation influences the shape of objects and entities but does not influence the topology. In addition to the exaggeration operation, objects can be emphasized by the displacement operation. The operation refers to move objects and entities in attempt to avoid overlapping or indistinguishable space between entities. An example of where displacement might be useful is when two cities share the same border but for reasons of clarification need to be separated. The displacement operation would add a non-existing space in between. Thus, the metrics of the entities are changed as well as the topology. The last generalization operation is aggregation, which is the most difficult one to implement. The operation aims to group entities together that individually would disappear, but is as a group still of significance. A suitable example is small lakes spread over an area. If aggregation was excluded the lakes would be reduced in large scaled maps, instead of more logical be represented as one big lake. Aggregation affects topology, metrics as well as shape, which contribute to the implementation difficulty.

Rules controlling the operations must be defined, i.e. the definition of what objects and/or entities is affected (simplified, reduced or immersed) by the operations and at which resolution level. The rules should also cover consistency aspects considering homogeneous and heterogeneous networks see Figure 19. Such networks are covered by Tryfona & Egenhofer (1996).

![Figure 19. Object a) qualifies as a homogeneous network and b) and c) as heterogeneous networks.](image)

### 5.4 Coordinate representation

Several alternatives of representing coordinates, or rather quantization of coordinates, of spatial entities may be used to reduce the size of spatial data. But first, the demand of the

---

precision of the coordinates must be stated. If the demand of precision is low, an effective way to quantify coordinates is to transform the coordinates to fit the screen of which the data is to be displayed. The resolution (i.e. amount of pixels) of the screen is limiting, a pair of coordinates cannot be rendered to less than a pixel. The coordinates \((x, y)\) could be transformed to \((x', y')\) according to:

\[
\begin{align*}
x' &= \frac{w_{res} - 1}{r_{lon, max} - r_{lon, min}} (x - r_{lon, min}) \\
y' &= \frac{h_{res} - 1}{r_{lat, max} - r_{lat, min}} (y - r_{lat, min})
\end{align*}
\]

where \((w_{res}, h_{res})\) denotes the resolution of the screen, \(r_{[lon, lat], \{min, max\}}\) denotes the limits of the reference system in two dimensions. On a screen with 256 x 256 pixels, each coordinate should not require more than 8 bits. In a map containing 40 000 pairs of coordinates the total amount bits reduced by using an 8 bit representation instead of 32 bit which is common (if represented as an integer), would be 960 000 bits or 117,2 kB.

However, depending on if a cache is used in the client side of the application and how it is implemented scaling to screen coordinates may not be possible. Since an implementation of a cache probably would use coordinates to determine if objects are already cached, the coordinates transferred to the client must be related to coordinate reference system used in the remote side. By using integers, scaling means loss of precision and consequently loss of relation to the reference system.

Another approach to coordinate quantization is to use relative coordinate systems, or differential vectors, to describe the geometrics of a spatial entity. The first alternative is to define differential vectors by determining a vector as the difference between a point on polyline and the startpoint of the polyline, i.e. \(\Delta_{i, 0} = \{x_i - x_0, y_i - y_0\}\), where \((x_i, y_i \mid i = 1, ..., n)\) are points on the polyline. The differential vectors can be quantified with fewer or in worst-case equal number of bits, since \(|\Delta_{i, 0}| \leq |x_i, y_i|\). A second alternative is to define differential vectors as the difference between a point and the previous point, i.e. \(\Delta_{i, i-1} = \{x_i - x_{i-1}, y_i - y_{i-1}\}\), where \((x_i, y_i \mid i = 1, ..., n)\) are points on the polyline. The probability of quantifying the differential vectors with fewer bits is higher than in the first alternative.

A third alternative is to use point prediction, where the differential vectors are defined as the difference between a point and a predicted point as illustrated in Figure 20. The predicted point could be calculated by adding the length of the previous differential vector to the previous point.
The probability of quantifying the differential vectors with even less bits should be higher than in the two previous alternatives. However, the pattern is more difficult to predict, as the variance of the differential vectors tend to be higher in the point prediction alternative than in the other two. Depending on the size of the display on the mobile device, the number of coordinates that make up a map may vary. Rendered in an average display of current Smartphones, a map may consist of some 40 000 coordinates. Table 8 illustrates the number of bits reduced by adopting any of the three suggested coordinate quantizations.

<table>
<thead>
<tr>
<th>Nbr coords</th>
<th>Quantization alternatives</th>
<th>[reduced nbr bits]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Screen</td>
<td>$\Delta_i, 0$</td>
</tr>
<tr>
<td>1000</td>
<td>24 000</td>
<td>16 000</td>
</tr>
<tr>
<td>15000</td>
<td>360 000</td>
<td>240 000</td>
</tr>
<tr>
<td>25000</td>
<td>600 000</td>
<td>400 000</td>
</tr>
<tr>
<td>40000</td>
<td>960 000</td>
<td>640 000</td>
</tr>
<tr>
<td>75000</td>
<td>1 800 000</td>
<td>1 200 000</td>
</tr>
</tbody>
</table>

Table 8. Examples of the reduced number of bits with the different representations.

The representation techniques described above can be used in a combination with scaling coordinates to fit the resolution of the display, resulting in a more compact representation.

### 5.5 Spatial data storage

By storing the spatial data in a database dedicated to visual presentation, more advanced optimizations is possible in order to enhance software performance in the server providing the data. The increased modularity of the server optimizes each module based on efficiency (do things right) and effectiveness (do the right things).\footnote{Grant, R. M., *Contemporary Strategy Analysis – Concepts, Techniques, Applications* (2002).} For example, important optimization issues are efficient data storage in terms of fast access and reduction of redundant data. The
later is a time and memory-consuming process, which require a preprocessing task as well relatively static datasets. The implementation of fast access to multi-resolution data also requires a preprocessing task where rules defined in 5.3 are applied with appropriate input parameters. For instance, entities that are preserved through multiple levels should only be stored once in the lowest resolution level to avoid redundant data. Higher levels should only introduce new entities and more detailed representations of entities in previous levels. In addition, objects and entities can be stored with topological relations such as disjoint, meet, contains, covers, equal, overlap, inside, covered by, bounds, and bounded by. The relations assist to perform map generalization and are required to compute multi-resolution map data. Map generalization and compression require that topological data is separated from the geometric data. By storing it separately instead of separating it in real-time, the generalization and compression is enhanced considerably.

Optimization of object representation is made easier with static processing. For example when representing bounding regions, lines describing the regions are stored redundantly. To represent the regions in Figure 21 AC, BC and CD must be stored twice each resulting in point A, B, C, and D being stored three times each. By avoiding such redundancy the size of data transmitted to the application can be heavily reduced. One solution to this problem is to via the relation operators bounds and bounded by store common points in a list. By assigning entities with references to the list of common points, great savings in file size can be achieved.

![Figure 21. Illustration of bounding regions.](image)

### 5.6 Semantic caching

Caching frequently accessed data on the client side in a client-server system is an effective technique to improve performance especially in mobile environments. Compared to traditional distributed environments, the mobile environment introduces unique characteristics due to properties of wireless communication and mobility. For instance, the wireless

---

connection is a major performance bottleneck because of the limited bandwidth. In addition, the mobile environment offers frequent disconnections both foreseeable and unpredictable. Power outage or wireless network failure is known to cause frequent non-predictable disconnections. By using a semantic caching approach, data from previous queries to a server, a forthcoming query may be answered by data stored locally instead of answered remotely. Semantic caching refers to a client maintaining both semantic descriptions of queried data (i.e., content of a map) and results of previous queries.\textsuperscript{43} The semantic caching could be applicable in an LBS application in order to cache geospatial objects or entities, instead of addressing an entire map as a cache object. By describing geospatial data semantically as abstract cell complexes rather than an entire map and storing previous queries of the complexes, content in the map could be reused in order to fully or partially answer a query locally on the client side. Figure 22 shows the reuse of previous query answers to complete a future query. A query can be separated into a probe query (answered locally) and a remainder query (answered remotely). By processing previous queries can substantially reduce the amount of data transferred over the wireless connection.\textsuperscript{44} In addition, in the occurrence of disconnections foreseeable or unpredictable data being transferred prior to the disconnection may be stored locally and reused later.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{query_reuse.png}
\caption{Reusing of query answers.}
\end{figure}

Due to limitations in memory size in mobile devices, it is impossible to cache all accessed data items in the cache. Therefore, it is necessary to implement a cache strategy based on cache replacement algorithms in order to find suitable subset of data items to store in the cache. The cache strategy aims to maximize the hit ratio, while minimizing resource usage. Each strategy attempts to foresee what data are most likely to be needed again. The main difference between cache strategies is how cached elements are selected for elimination when the cache reaches its limit. The cache replacement rules suitable for caching spatial data could be:

\begin{flushleft}
\textsuperscript{44} Ren, Q. & Dunham, M., \textit{Using Semantic Caching to Manage Location Dependent Data in Mobile Computing} (2000).
\end{flushleft}
5 Design proposals

- **LRU – least recently used**, refers to the eviction of objects in the order they were added to the cache. It is implemented as a priority queue, where elements addressed in the cache are moved to the front of the queue.
- **LFU – least frequently used**, objects are evicted based on when they were added, when they were last used, and the usage frequency.
- **FAR - farthest element**, is suitable for storing spatial objects. Objects farthest from the current location of the user of the application are evicted.

The FAR rule is very useful in tracking applications where a direction of the moving user can be detected. The rule is only applicable if the coordinates of objects are absolute in the sense that they are not scaled to fit the screen as suggested in 5.4. In addition, when a direction of a moving user can be detected automatic queries of future spatial data could be sent to the server, transparent to the user.

When using the LRU and LFU rules objects need to be associated with a time stamp. This adds to an additional dimension to the abstract cell complex used to represent objects in a map. Objects would be defined by \((a, v, g, t)\) (see 5.1), where \(v\) is a number of values accessed through a set of attributes \(a\), \(g\) is one or more spatial entities, and \(t\) being a time stamp when the object is last addressed by the cache. Any combination of the cache replacement rules may be applied to the cache. An additional rule might prove to be useful, if multi-resolutions of objects are used it would be wise to replace higher-resolution data prior to lower-resolution data. A more sophisticated cache replacement policy is proposed by Yin et al, aiming at developing cost functions for targets on caches such as minimizing query delay or minimizing downlink traffic.\(^{45}\)

A semantic cache strategy can be applied to access spatial objects in a map illustrated in Figure 23. The strategy starts with checking if the addressed map is stored in the cache by constructing a query from previous queries. The query is trimmed to a probe query and a remainder query. Three scenarios can occur: the remainder query is empty - the query is answered locally, the probe and remainder is not empty – part of answer is stored locally and remotely, and the probe query is empty – the answer is stored remotely. This operation should be of low complexity and the search effort is determined by the cache replacement rules. If data or parts of data are not stored in the cache it needs be fetched from the server, and the remainder query gets cached in the semantic cache. The probe query, if not empty, gets updated according to the prevailing replacement rule. An introduction of spatial queries is given by De Floriani et al (1993).\(^{46}\)

Since the cache relies on object ids, it is necessary to validate the cache when a user starts a session in the application. When the geospatial database is updated, the probability of object ids being changed is not negligible.

### 5.7 Threaded data processing

To make efficient use of the limited hardware resources in mobile devices, a specialized form of multitasking called multithreaded programming may come in handy. Threads are lightweight tasks within an application allowing context switching from one thread to the next at a low cost.\(^47\) By allowing processes in the application to be executed in parallel, idle time can be kept at a minimum. Multithreading is especially suitable for interactive environments such as an LBS application, where user input determines the behaviour of the application. In addition, as the transmission rate of the wireless connection to the server providing the spatial data is limited a single-threaded application would have spent most of the time waiting for the download task to finish before it can proceed to the next i.e. displaying. By dividing the tasks into different threads the utilization of the hardware can be enhanced.

To further simplify the rendering process the data should be organized when extracted in the server in order to minimize the processing effort in the mobile device. It is more feasible to allocate the cost of structuring data where hardware resources are less limited. The organization refers to both ordering within data packages and among data packages. For instance, when transmitting multi-resolution data referring to the notation of a map \( M = \{ M_i \mid i = 0, \ldots, n \} \) as a multi-resolution sequence, where \( M_0 \) is the map with maximum resolution,

---

when transmitting in multi-resolutions the least detailed data should be transmitted followed by more detailed data, i.e. $M_n, ..., M_0$. The transmission of the multi-resolution data should be performed in transparently callbacks to the user. The data should also be ordered within each data package, e.g. resolution, in a correct rendering order. These organizations performed on the server side, facilitates the management of data in the resource limited client side. This feature makes partially progressive display possible, in which each layer of the map is rendered as soon as it is fully downloaded and interpreted. When a user is performing heavy zooming in the map, the callbacks from a previous zoom-level must be interrupted to avoid the “outdated” data to be transmitted. Still each request to the user should be state-less to avoid sessions in the server. Sessions would limit the server capacity by limiting the number of simultaneous processes.

## 5.8 Compression techniques

To further reduce cost of communication and the size of the vector data transmitted to the mobile device, compression techniques can be an effective approach. Compression techniques can allow larger subsets of maps covering a larger area than merely the visual area of the subset to be transferred to the mobile device. Larger subsets would result in more efficient cache hit ratio, e.g. when the user is panning the map. A traditional LZW compression technique, which is a loss-less algorithm may be applied, but other compression techniques might prove to be more effective. Depending on the choice of compression technique, distortion may be added. However, as cartographic generalization is used approximation errors are already introduced. Therefore, lossy compression algorithms are acceptable if the error can be limited. In addition, the decoding effort should be kept simple due to the limited computational resources of mobile devices.

An alternative compression technique recently applied to vector maps is a dictionary compression technique.\(^\text{48}\) The technique focuses on identifying differential vectors (see 5.4) in data to construct a dictionary. To identify the differential vectors an algorithm called *K-mean clustering algorithm* is used, which takes as input a fixed number (K) and generates that many *clusters*, i.e. vectors, as output. The clusters become the dictionary. The FHM algorithm chooses the clusters statically according to Figure 24. The size of the clusters is decided by the Fibonacci sequence and delta values.

Each differential vector is encoded to the index of the spatially closest dictionary entry. By limiting the size of the dictionary approximation errors are introduced. To apply dictionary compression to vector data, the data must be separated into topological data and geometric data. Since the topological parts of the data needs to be preserved (as the case in polyline simplification), the compression technique can only be applied to the geometric data. The pseudocode for the dictionary compression algorithm is presented below:

1) for each polyline do 
2) Separate start point and sequence of differential (delta) values
3) end;
4) do
5) if CCM algorithm do
6) K-mean clustering on sequence of delta values, each cluster is a dictionary entry
7) else
8) Set size of cluster from delta values
9) end;
10) for each polyline do
11) Encode polyline using dictionary
12) end;
13) end;

The representation of coordinates affects the appearance of the dictionary. When choosing differential vectors based on the first point ($\Delta_i, 0$), the distribution of the dictionary entries is more optimal than if differential vectors based on the previous point ($\Delta_i, i-1$) is chosen. This is
due to the lengths of the \((\Delta_{i,0})\) vectors are generally larger than the lengths of the \((\Delta_{i,i-1})\) vectors. Hence, the approximation error is lower when representing coordinates according to \((\Delta_{i,0})\). In addition, since each differential vector in the \((\Delta_{i,i-1})\) representation is related to the previous one the approximation error is accumulating. This is also true for the prediction representation of the coordinates. Figure 25 illustrates a typical distribution of the dictionary entries of \((\Delta_{i,0})\) and \((\Delta_{i,i-1})\) representation.

![Image](image.png)

**Figure 25.** The distribution of a) \(\Delta_{i,i-1}\) and b) \(\Delta_{i,0}\) dictionary.

The compression ratio of the technique depends on the number and size of the reference vectors. To use a reference vector for types of spatial entities rather than for each spatial entity would increase the compression ratio. However, it might also increase the approximation error since the differential vectors of entity types could vary significantly.

### 5.9 Cartographic appearance

To be able to interpret spatial data visually, it has to be associated with metadata describing how each object should presented on the display. For example, textual descriptions, colours, and paths need to be attached to objects to allow the viewer to separate objects and types of objects. By adding cartographic metadata to the spatial data, the result could be quite large. If added to the data being transferred to the application, the same metadata would be transferred repeatedly. This is wasteful and is the reason why cartographic data should be kept separated from the spatial data. One approach to avoid the transfer of cartographic metadata is to add the interpretation of the spatial data in the application. However, the interpretation should be implemented to allow flexibility of the cartographic appearance of a map. That is, the cartographic metadata should be able to be changed without updating application software. By defining cartographic settings as configuration items, the application could perform version checks of its current setting. The settings would define attributes of each each layer of the map.
5 Design proposals

5.10 Design approach

The key success factors given earlier are non-trivial in the sense that they require extensive implementation efforts. The ideal solution would to be able to perform the data processing such as generalization, redundancy checks, and compression in real-time. However, these operations are time and resource consuming tasks. Therefore, pre-processing of the spatial data is necessary. A map generalization design proposal is presented by Dettori & Puppo (1998). It can be extended to not only include map generalization but also object identification, compression, and spatial data storage (coordinate quantification and redundancy check).

5.10.1 Map generalization design approach

As mentioned earlier, map generalization aims to organize and select the proper amount and kind of spatial data to facilitate a graphical representation suitable for interpretation of the viewer. The generalization process may be divided in model-oriented generalization and cartographic generalization. The former refers to decisions of what information to be included in the map, and the latter refers to decisions how to display such information.49

In Figure 26 a map generalization design is presented.50 The design proposal includes a layered library, which describes the integration of model-oriented generalization and cartographic generalization. The library consists of a kernel at the lowest layer, transformation layer, which consists of three independent modules devoted to semantic, topological and metric transformations, model-oriented generalization layer, cartographic generalization layer, drawing layer, and application layer.

![Figure 26. Map generalization design.](image)

---

50 Ibid.
The kernel is responsible for the construction, modification, and inquiry of the spatial database representing a map. The kernel includes atomic functions such as:

- **Graph creation**: operations to build topological structure, associating semantic information to entities etc.
- **Inquiry functions**: retrieval of information from the map.
- **Transformation**: operations performing atomic topological generalization such as contractions (line-to-point, region-to-point, region-to-line), merge (line-merge, region-merge), and abstractions (point-abstraction to line or region, line-abstraction to region).

The transformation layer consists of three independent modules. The semantic module contains semantical information of spatial entities such as generalization rules. The rules include description of which resolution levels an entity exists and its state and order, e.g. if a region is contracted or abstracted. The topological include high-level operations built as sequences of the topological atomic operations in the kernel, e.g. contractions and abstraction of a complex object, i.e. consisting of entities of different orders (points, lines, regions). The module only manipulates topological relations between entities, and leaves shape and semantic changes to the other modules. The metric module is responsible for computing shape transformations based on numerical and geometrical algorithms. It includes operations like polyline simplification as well as shape modifications induced by contractions and abstractions. For example, the metric module calculates the placement of the point in a line-to-point contraction.

The three modules work independently, but need coordination in performing operations, which include a combination of semantic, topological and metric operations. The coordination is supervised in the model-oriented generalization layer. Its primary task is to call on metric and semantic operations when topological manipulations are performed.

The cartographic generalization layer uses a map model created by the model-oriented generalization layer and produces another map model modified to aesthetic needs. The layer performs modifications, which facilitates the visual interpretation of the map. Typical operations performed are displacement and exaggeration. Since displacement and exaggeration, as described in 5.3.1, may affect the shape of entities the layer must be granted access to the metric module. The layer may also access the atomic topological operations in the kernel, as displacement may produce topological modifications. The output of the cartographic generalization layer should be a multi-resolution model of the map, suitable to be stored in a database.
The drawing layer is responsible for the visual presentation of the maps, e.g. in a mobile device. It is located on top of both the model-oriented and cartographic generalization layers, since both produce a map model. The layer needs to access the spatial data through the inquiry functions of the kernel.

At the very top of the design is the application layer. Depending on the type of application, it needs to access one or more lower level.

5.10.2 Extension

The model suggested by Dettori & Puppo could be extended to not only include map generalization but also object identification, compression, and spatial data storage. The kernel is responsible for converting raw spatial data to a topological graph. In the converting process some of the features addressed in this chapter could be included. When creating the graph each object in the map should be stored as an abstract cell complex. By definition each object should therefore be associated with a unique identification. To avoid the risk of escalating number of ids, the uniqueness could be limited to layers (types of objects) or even submaps. To make more efficient use and further facilitate map generalization, redundant data should be avoided. Therefore, a redundancy check should be performed be the kernel. Redundant data can be duplicate entities belonging to several objects and at lower level duplicate pairs of coordinates (see 5.5). The redundancy check is time and memory consuming, which makes it suitable to perform in the initial stage of producing the map model. The kernel should also be equipped with operations to perform coordinate quantification (see 5.4). The spatial data would be considerably reduced, which also would lower the access time.

The model could also be extended to include compression techniques. A compression layer is then added in top of the model-oriented generalization layer and the cartographic generalization layer, since the two layers produce a map module ready for compression. The compression layer can be implemented using he techniques presented in 5.8. To be able to perform the compression algorithms on spatial data, the compression layer needs to access the inquiry operations in the kernel. The extended model is illustrated in Figure 27.
Figure 27. Extension of the map generalization model.
This chapter describes which design proposals were found to be feasible to implement in the location-based service system at Wayfinder Systems AB. A short motivation of the selection is given along with reflections from the implementation and testing.

To be able to validate and verify the design proposals suggested in the previous chapter they ought to be implemented in an authentic LBS system. By that, the effectiveness of the proposals is easier to determine. As described in 2.4, the implementation will occur in the system at Wayfinder Systems AB. However, due to time limitations of the study it was not possible to implement all of the proposals. Therefore, a selection of which proposals found most feasible to implement preceded the implementation.

The implementation was conducted at Wayfinder Systems AB, and the test suite was composed of a cluster of the actual server. The cluster approach was chosen and provided a closed environment for testing not affecting the customers of Wayfinder Systems AB. The application in the mobile phone was simulated by a Java application. A screenshot is presented in Figure 28.

Figure 28. Screenshot of the application simulating the mobile phone.
The proposals that were implemented were *polyline simplification* according to the Hershberger-Snoeyik algorithm (see 5.3.1), *FHM vector compression* (see 5.8), *semantic caching* (see 5.6) and *threaded data processing* in the client application (see 5.7).

The polyline simplification proved to be very effective in reducing the size of data by discarding vertices on the polyline. The tolerance, or rejection criterion, may well be correlated to the current resolution of the current zoom level of the map. The simplification was implemented to allow a real-time reduction of data performed at each query. This may not be as optimal as compared to pre-processing data, since all coordinates must be traversed at least one additional time. The size of data was, however, heavily reduced whereas the transfer time was also reduced. In order to get a perfectly satisfying result from the simplification, the reduction must exclude the vertices with topological relations or else inconsistencies might be invoked. Optimally, the vertices of topological importance should be “flagged” and should be left untouched when traversed in the algorithm. Topological information of vertices should therefore be included in the data structure of objects as suggested in 5.5, since tracking topological relations in real-time would must probably be quite time consuming.

Vector compression is also an efficient tool to further reduce the size of data. The compression technique could easily be used in addition to simplification. Also the vector compression technique was implemented in order to be applied in real-time. Therefore, the FHM compression technique was chosen. The FHM provides a less intelligent encoding technique than the K-clustering technique. The biggest difference lies in the selection of reference vectors. The FHM uses a static selection whereas the K-clustering offers a dynamic selection based on statistical properties of the differential vectors to be compressed. The FHM is simpler to implement and, first and foremost, does not require a great deal of data processing, which makes it suitable for real-time appliance. The penalty is a higher approximation error. However, by choosing parameters such as the number of available reference vectors and distances between the “boxes” (see Figure 24) wisely, improvements to limit the approximation error were possible. The distances between the “boxes” was chosen in reference to the smallest and the biggest differential vector in each spatial entity. Regarding the introduction of approximation errors, the same discussion concerning topological preservation as in simplification is applicable. The implementation of the compression technique should allow preserving each differential vector, which has topological importance by adding it to the collection of reference vectors. The compression ratio then depends on the amount of topological data and level of acceptable approximation error.

The theory of the semantic cache was implemented and tested at Wayfinder Systems AB. The cache items equalled objects in the map. However, as the internal format at Wayfinder Systems AB currently does not provide object identification a workaround for requesting
objects was needed. Instead of addressing the ids of each object to determine if the object is included in the request, the objects were included if they were covered by the extreme coordinates of the map or the map was covered by the objects. The workaround reduced the performance of accessing the cache.

To improve the use of the semantic cache, threaded data processing was implemented in the client application. While instantly scaling existing objects and render cached objects while querying the server for additional objects, the visual feedback was enhanced. The Java language offers extensive support to handle light-weight threads and provides synchronization mechanisms to avoid dead-locks. Unfortunately, the rendering API offers poor rendering performance making the effect of threading less astonishing.

The selection of design proposals to implement focused on reducing the amount of data being transferred to the client. Since the transfer speed is an obvious bottleneck in mobile environments this was a choice of high priority. However, by choosing to apply the simplification and compression algorithms in real-time the execution time of software modules in the server is affected. Therefore, the second activity involving a performance evaluation of the LBS system is executed.
This chapter presents the execution of the second activity of the study – performance evaluation. Its purpose is to answer a number of interesting questions regarding capacity and responsiveness, in order to evaluate the performance requirements. A method for deriving a performance model of the location-based system is presented.

When adding functionality to parts of a software system or scaling a system drastically unexpected results may occur. Inevitably, the performance of the system is affected, where performance refers to responsiveness, throughput, utilization and scalability among others. For example, the response time and throughput of the LBS application depend, amongst other, on the interaction of software components in the server and to the hardware, to which the modules are allocated, as well as the bandwidth of the wireless connection. The performance of the server is also varying with the amount of users requesting service simultaneously. Typically, the performance worsens when the load on the server increases drastically. This chapter will present a theoretical background and tools to generate a performance model of the LBS system at Wayfinder Systems AB in order to contribute in the evaluation of the effects of implementing the design proposals suggested in chapter 5.

Software performance is an important but often neglected aspect of software development. Traditional software development methods have taken a “fix-it-later” approach to performance.51 Software Performance Engineering – SPE is a concept that accentuates management of performance concerns in the development of software systems. SPE prescribes a systematic approach for integrating performance with design and will be further described in chapter 7.1. In this thesis, a performance evaluation will be carried out within the scope of SPE. The estimation and evaluation will be managed through a concept developed from SPE. The concept, called lifecycle validation of software performance engineering, is the process of predicting and evaluating the ability of the software product to satisfy the user performance goals.52 This is a helpful tool for integrating performance from early phases of development and on. This chapter will describe a method providing a step-wise derivation of a performance model from software performance model from development documents.

7.1 Software Performance Engineering

Software Performance Engineering provides guidelines for applying well-established performance modeling techniques throughout the development process to help ensure that the resulting software system meets its performance requirements.\(^{53}\) These techniques are tools of estimating the demands on software and use performance models to predict the performance of the system. One of the main benefits is that it is much easier to correct critical performance issues when they are identified in the early stages than it is after code has been developed, tested, and implemented.\(^{54}\) Thus, most success is likely to come from those projects, which attempt to integrate and build performance analysis directly into software design methods. Due to the fact that cost of correcting errors is increased exponentially in oncoming phases, early warning systems are necessary. SPE allows you to detect and identify performance issues early in the design stage when correction costs are lowest, in extension allowing software projects to be completed with less rework. By building and analyzing models of proposed software architectures and design it is possible to evaluate their ability to meet performance objectives. While limitations of detail exist in early assessments, simple models and calculations will suffice. It is also possible to identify architecture and design not suitable or even not cost-efficient enough in economical terms to meet user demands and requirements.\(^{55}\)

7.2 Lifecycle validation

Lifecycle validation is as mentioned before a process of predicting and evaluating the ability of software products to satisfy user performance goals. In order to implement this concept in a systematic fashion a strategic scheme is presented in Figure 29.\(^{56}\) The strategy scheme acts as a framework for performance validation and evaluation. The term *artifact*, which is used in the scheme, here means a final or intermediate version of the software program. The addition of new functionality in the LBS system such as implementation of vector graphics will be addressed as a phase of the development of the LBS. A further explanation will be given in chapter 8.

The strategy scheme may be applied to iterations of the construction phase of the development process. In the validation of the current iteration \(i\) it is assumed that the artifact


already validated in \((i-1)\)th iteration is available and received as input. Based on the previously artifact a new tentative artifact is produced. The artifact is validated by comparing its predicted performance with the user performance goals. If the validation results are satisfactory the artifact is delivered to the \((i+1)\)th iteration. In the case of rejection and if it is considered cost-effective, a new tentative artifact is produced for better performance. Else, if not considered cost-effective the user performance goals need to be revisioned before moving on to the next phase.

The performance evaluation in the strategy scheme may be performed in several ways. Still three main activities can be identified:\(^{57}\)

1. Performance model production.
2. Performance model evaluation.
3. Comparison of evaluation results with user performance goals.

The first two activities will be explained in this chapter and will be applied to the considered system in chapter 8 and 9 respectively. The third will however not be addressed until the results of the analytical calculations has been presented.

---

7.3 Performance model

Before commencing the performance evaluation, a description of the system is necessary. The model of the system is based on the system description and definition. The description of the system is derived from use case analysis and software development documents. Use case analysis summons use cases into an external picture of the system.\(^{59}\) Class diagrams and interaction diagrams such as sequence diagrams provide a deeper knowledge in a conceptual perspective. The purpose of the performance model is to describe the reality in simple terms. The model must further provide acceptable explanations of the behaviour of the system. It is common to define the system from high-level design documents. However, it might prove that not all aspects of relevance to the performance model are covered by the high-level design. If properties of the system are not explained or represented by the model, it has to be revised or exchanged. An acceptable and reliable model must be consistent, that is it must be free from


contradictions concerning the behaviour of the system. Therefore, unessential elements of the reality must be discarded.\textsuperscript{60}

### 7.3.1 Method description

The first activity of performance estimation derives a performance model from existing development documents, such as analysis or high and low-level design documents. This requires that the software be developed according to an object-oriented approach. To generate a performance model the method illustrated in Figure 30 is used. The model relies on a number of artifacts gathered from software development documents as input before producing a model. The top left area contains artifacts from the elaboration phase of the software development. The documents are usually class diagrams, sequence diagrams, platform data and operational data. The class diagrams refer to the whole system, while the sequence diagrams refer to a particular scenario. The platform data contains platform configuration and resource capacity data. The operational data describes the operational profile of different users and their workload data. In the case of distributed software as in this thesis, the top right area of the method is also used. This area contains artifacts from the preliminary design phase and relates to documents such as software modules architecture, client/server structure of modules and modules-platform mapping. Based on these input documents it is possible to generate a performance model. The generation can be described in five steps:\textsuperscript{61}

1. First, a set of precedence graphs – PG is generated based on sequence diagrams and class diagrams. Each precedence graph is related to a specific scenario and is therefore called scenario precedence graphs. They are used to identify the execution flow and the interaction between actors in the system.

2. The different precedence graphs are merged together into a global precedence graph. The global precedence graphs contain information of all scenarios and represent the behaviour of the whole system.

3. Based on the global precedence graph, the workload data from the operational profile documents, the software modules architecture and client/server structure, and an extended flow graph – EFG is generated.

4. An extended execution graph – EEG is generated from the extended flow graph. The EEG is weighted with resource demands derived from the platform configuration and platform data documents, experience data and the modules-platform mapping document.

5. Finally, a layered queuing network - LQN performance model is generated based on the extended execution graph and resource capacity data.

\textsuperscript{60} Arvidsson, Åke, Simulering (1988).

\textsuperscript{61} Cortellessa, V. et. al, Automatic derivation of software performance models from CASE documents (2001).
The LQN performance model is then evaluated and is described in the following section.

![Image](image.png)

**Figure 30.** Method outline for performance model generation.\(^{62}\)

### 7.4 Theory of Layered queuing networks

Software systems often consist of software processes, acting as clients and software servers, and hardware servers. Processes compete among themselves for access to software and hardware resources. Queuing network models successfully describe traditional time-sharing resources, but often fail to capture complex interactions among various software and hardware components in distributed client-server systems.\(^ {63}\) Due to client-server relations, contention will appear between software and software components as well as software and hardware components. This will give rise to delays, limiting response time as well as throughput of the

---

system. These performance issues are of particular importance in distributed and multiprocessor systems. LQN is an adaption of queuing models of systems with software and hardware servers and resources. A model in LQN is closely linked to software specifications, which makes it easy to develop and understand. It is well suited for systems with parallel processes running on a multiprocessor or on a network, such as client-server systems. The requests for service amongst processes and for devices can be described as a software process architecture. The performance of the system is often determined by its software process architecture. A number of parameters originating from the architecture such as the number of instances of processes, extent of process interaction, and the software-hardware mapping, have a significant impact on the performance. The use of performance modeling refers to the understanding of the behaviour of the system and discovering which of the system’s features constrain performance. Often features of the system do not already exist, as in this case. This leads to estimation of parameters as input in the predictive model. The estimations might be based on measurements on similar systems, experience data or data corresponding to worst-case scenario. A layered queuing network estimates the contention for resources and process throughputs based on each process’s resource demand such as CPU time requirements and request for service from other services. The predicted performance measures and typical output parameters from LQN solvers are process throughputs and response time, process utilizations and the average number of processes queued at each serving process, and device utilizations and mean queue lengths. It is possible from the layered queuing network model to determine bottlenecks in the system. In its extension, it is possible to determine the optimal software-hardware mapping.

The main actors in an LQN model are tasks and devices. Tasks request service from devices as well as other tasks. The service and queuing discipline may be FCFS, Priority, preemptive, Head of line, Random etc. The model is illustrated by letting a parallelogram denote tasks, and a circle denotes a device see Figure 31.

![Figure 31. Illustration of a layered queuing network model](image)

---

The requests for service amongst processes are described as a directed graph. Tasks that do not request service from other process are defined at the lower levels, while other at higher level. Each task is by definition located at only one level. A task may offer more than one kind of service, each modelled by an entry. Entries have their on execution times and demands for other services offered by other tasks. However, servers (tasks called upon) with multiple entries still have a single input queue, where requests for different entries wait for service.

The LQN may be split to two separate models. The first model is the software contention model, which describes the relationships of software tasks. The second model is the hardware contention model and is used to determine queuing delays at devices. The LQN may be solved by an algorithm called Method of layers, which, combines the results of the two complementary models to provide performance estimations for the whole system. The purpose of the method of layers is to find a fixed point where predicted group idle time and utilizations are balanced so that each group in the model has the same throughput whether is considered as a client or server and the average service time required by callers of the group equals its average response time. The LQN is easily solved by Method of layers or other analytical solvers. The generated LQN in this study will be solved by using an analytical tool called LQNS.

![Software Contention Model](image1)

![Device Contention Model](image2)

**Figure 32.** Software contention model and device contention model in the method of layers.

---


This chapter describes the considered system of which the performance evaluation is based on. The prerequisites affecting the performance estimations will be presented. The theory in the previous chapter is applied to it. The chapter ends with a performance model of the LBS application.

In this chapter, a performance model will be constructed of the LBS system, using the step-wise method for generating an LQN presented in the previous chapter. The LBS system is based on the system currently running on Wayfinder Systems AB. To avoid compromising the intellectual property rights of Wayfinder Systems AB, a detailed description of the system is not provided. A description is however needed to be able to initiate a performance estimation of the impact of adding new functionality by introducing a new format for visual representation of geospatial data, and is given in a general manner without sacrificing any incentive properties. The introduction may be visualized as the i:th iteration in the framework presented as the strategy scheme of the lifecycle validation in 7.2.

8.1 System description

A system can be described by using the five view model. The LBS application and system developed by Wayfinder Systems AB can be described in similar but less formal manner by referring to the documents suggested by the method of Cortellessa. However, the information gathered summons the relevant information of the system needed in the study and relate to the views in the five view model. The views are:

- The use case view is describing the behaviour of the system as seen by the stakeholders, such as users etc., in terms of stating a number of use cases which summons the behaviour of the system.
- The design view comprises the functionality requirements by describing class diagrams, sequence diagrams etc.
- The process view is similar to the design view but emphasizes processes, threads and multiplicity of classes and objects.

---

69 A use case is a description of one small task a user could perform when using the application.
• The deployment view is describing the mapping of software components onto hardware components.

• The implementation view encompasses the software module organization by including components and files. This view is less relevant to performance modelling and will not be further addressed.

The system provides a LBS (see definition in 2.1.3). It involves an application running in mobile devices connected to a server via a wireless network illustrated in Figure 33. The application allows, among other features, a user to visualize a location on a map or to search for the shortest route between two locations and displaying it on a map. The user is able to pan, resize, rotate and zoom in or out in the map. The application also enables information to be linked to objects in the map, allowing the user to receive additional information of points of interest. These features imply that the image must be processed in the application. The server is providing the application maps and data regarding routes and information of points of interest. The most essential functionality of the application can be described in four use cases $UC_1$, $UC_2$, $UC_3$, and $UC_4$.

• $U_1$: Search for locations. The search is performed prior to the calculations of a route.

• $U_2$: Search for the shortest route between an origin and a destination.

• $U_3$: Get a map where a route is displayed. A search for the shortest route must have been initialized prior to this use case. The request is also sent when the user is panning or zooming beyond the limits of the current map containing a route.

• $U_4$: Get a map of a specific area. Used when displaying the location of the user or any other location of choice. The request is also sent when the user is panning or zooming beyond the limits of the current map.

**Figure 33.** Overview of the LBS system.

### 8.1.1 Prerequisite

The performance validation of the system is based on a prerequisite that the system is operating in its busy hour. Thus there is a constant load on the server. Assumptions regarding the extent of the routes requested by the user are also made. The parameters of a route affect
the system in two dimensions. Evidently the complexity of calculating the shortest route\(^70\) is affecting the system, but also the effort of managing the amount of map data. An average route is therefore defined by a distance within a medium sized city i.e. Stockholm in Sweden.

### 8.2 Inputs from the analysis phase

To generate a performance model inputs from earlier phases of the development are necessary. To determine the dependencies between software modules the interaction must be clarified, for example, in a sequence diagram. As stated in 7.3.1 the inputs from the elaboration phase usually consist of operational profiles, platform and software data, class diagrams and a number of sequence diagrams depending on the number of use cases defined. Each of the inputs will be described in the following sections.

#### 8.2.1 Operational profile

The operational profile aims to document the behaviour of the users of the system. With the information, it is possible to describe how the behaviour of the users affects the server. To determine the average load on the average number of calls made by the average number of users must be determined. During an average busy hour, the system gets \( N \) calls from its users. According to the operational profile and estimations, the probabilities for each use case are presented in a probability tree in Figure 34 and in Table 9.

\[ \text{Figure 34. Probability tree for the defined use cases.} \]

\(^70\) The calculation of the shortest route between two locations is based on the Dijkstra’s algorithm, which complexity is defined by the ordo-notation as \( O(n^2) \), with appropriate modifications. Thus the complexity depends on the distance between origin and destination, route length and street density.
8 Performance modelling

<table>
<thead>
<tr>
<th>Use case</th>
<th>Probability $p_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_{C1}$</td>
<td>$P_{UC1}$</td>
</tr>
<tr>
<td>$U_{C2}$</td>
<td>$P_{UC2}$</td>
</tr>
<tr>
<td>$U_{C3}$</td>
<td>$P_{UC3}$</td>
</tr>
<tr>
<td>$U_{C4}$</td>
<td>$P_{UC4}$</td>
</tr>
</tbody>
</table>

\[ \sum_{i \in U_C} p_i = 1 \]

Table 9. Probability of each use case defined in the LBS system.

8.2.2 Class diagrams

The use cases are equivalent to scenarios where objects of the LBS system through transfer of control and data flow. UML class diagrams describe the types of objects in the system and the various kinds of static relationships that exist among them. The objects are derived from class diagrams and consist of object-oriented classes grouped into software modules. The system consists of a seven modules anonymously called Module A, Module B, Module C, Module D, Module E, and Module F. The software modules architecture and the hierarchy are presented in 8.3.1. The methods of each object are summoned and presented in Table 10. The methods are derived from UML class diagrams.

<table>
<thead>
<tr>
<th>Object</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>request</td>
</tr>
<tr>
<td></td>
<td>validate_user</td>
</tr>
<tr>
<td></td>
<td>method_1</td>
</tr>
<tr>
<td></td>
<td>method_2</td>
</tr>
<tr>
<td></td>
<td>method_3</td>
</tr>
<tr>
<td></td>
<td>method_4</td>
</tr>
<tr>
<td></td>
<td>method_5</td>
</tr>
<tr>
<td></td>
<td>method_6</td>
</tr>
<tr>
<td></td>
<td>method_7</td>
</tr>
<tr>
<td>Module A</td>
<td>method_8</td>
</tr>
</tbody>
</table>

Table 10. Methods for each software module derived from UML class diagrams.

8.2.3 Platform and software data

The platform configuration of the system is illustrated in Figure 35. The system consists of a mobile device where the LBS application is installed. The mobile device is connected to Module A through a WAN. The WAN is usually a wireless connection for transferring data.

---

such as GSM data, HSCSD or GPRS but can be of other. The Module A is connected by a LAN to Module B, Module C, Module D, Module E, and Module F where different internal functions are distributed. However, in order to simplify the model all modules are allocated on one hardware $HWS$. The modules-platform mapping scheme is presented in 8.3.3.

![Figure 35. Platform configuration of the system running on Wayfinder Systems AB.](image)

### 8.2.4 Sequence diagrams

Each sequence diagram refers to a particular use case, thus four sequence diagrams will be presented. The diagrams are used to produce precedence graphs in the first step of the performance model generation. As mentioned in 8.1, descriptions of events and interactions among the modules will comprise the intellectual property rights of Wayfinder Systems and is therefore not provided. The sequence diagrams are presented below.

In the first sequence diagram, the application makes a request to Module A, which is responsible for validating the user of the application. Module A then makes three synchronous requests to Module B and Module C before providing the application with a reply. The sequence diagram is presented in Figure 36.

![Figure 36. Sequence diagram corresponding to UC1.](image)
The second sequence diagram, illustrated in Figure 37, represents the behaviour of the system based on UC2. All requests are synchronous.

![Sequence diagram corresponding to UC2.](image)

**Figure 37.** Sequence diagram corresponding to UC2.

The third sequence diagram corresponding to third use case is illustrated in Figure 38. It introduces an asynchronous call to Module F. As soon as Module D has answered the call, Module A is free to reply the initial request from the application.

![Sequence diagram corresponding to UC3.](image)

**Figure 38.** Sequence diagram corresponding to UC3.

The last sequence diagram corresponds to UC4 and is only containing synchronous calls to the modules. The sequence diagram is illustrated in Figure 39.

![Sequence diagram corresponding to UC4.](image)

**Figure 39.** Sequence diagram corresponding to UC4.
8.3 Inputs from design phase artifacts

Along with inputs from the analysis phase, the generation a performance model requires inputs from the design phase. As stated in 7.3.1 the inputs from the design phase usually consists of software modules architecture, client/server structure and modules-platform mapping. The inputs are gathered from a number of legacy documentation existing at Wayfinder Systems AB. The data will be presented in the following sections.

8.3.1 Software module architecture

The software modules are derived from class diagrams and consist of object-oriented classes grouped into software modules. An illustration of the software modules architecture is presented in Figure 40. Each software module is associated with a specific multiplicity, which refers to the number of each module running simultaneously. The configuration and the number of operational modules are based on early performance analysis and customer
demand estimations. The configuration is not static and contributes to the scalability of the system. The configuration is illustrated in Table 11. All modules, except for Module A and Module E, are running in the same amount of simultaneous instances. Module E performs executions requiring substantial CPU time and is therefore scaled by a factor of two. All modules except Module A are single-processes with FCFS queuing discipline. Module A allows multiple simultaneous processes with the same queuing discipline.

**Figure 40.** Software modules architecture and hierarchy.

### 8.3.2 Client/Server structure

The client/server structure is a description of which modules act as clients and which acts as servers. The application evidently acts as a pure client. Module A however is an active server in the sense that it acts both as client and as server depending on interfaces. While interfacing with the application, Module A acts as a server. However, while interfacing with Module B, Module C, Module D, Module E, and Module F, Module A acts as a client. Naturally, the remaining modules act as individual pure servers. The client/server structure is illustrated in Table 11.

<table>
<thead>
<tr>
<th>Module</th>
<th>Role</th>
<th>Multiplicity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Client</td>
<td>-</td>
</tr>
<tr>
<td>Module A</td>
<td>Client/Server</td>
<td>m</td>
</tr>
<tr>
<td>Module B</td>
<td>Server</td>
<td>n</td>
</tr>
<tr>
<td>Module C</td>
<td>Server</td>
<td>n</td>
</tr>
<tr>
<td>Module D</td>
<td>Server</td>
<td>2n</td>
</tr>
<tr>
<td>Module E</td>
<td>Server</td>
<td>n</td>
</tr>
<tr>
<td>Module F</td>
<td>Server</td>
<td>n</td>
</tr>
</tbody>
</table>

*Table 11. The client/server structure and multiplicity of the modules of the considered system.*

### 8.3.3 Modules-platform mapping

Software modules have specific requirements in terms of hardware resources and should be allocated on suitable platforms. The access to hardware is often restricted economically. In many cases, this results in an optimization of available or existing hardware. Modules-platform mapping data are decisions on which software module to allocate on which hardware device. In this performance estimation it is assumed that only one modules-platform mapping
exist, although in reality several alternatives exist. Further, it is assumed that all the modules to the hardware server HWS. The performance model of the LBS system will not separate the processes running on different hardware. A more modification ought to take into consideration which processes are executed on which hardware and if possible even communication delays between the hardware.

### 8.4 Precedence graphs

Precedence graphs are used to identify the execution flow and the interaction among system actors. The precedence graphs are easily derived from artifacts such as class diagrams and sequence diagrams. This section will present four scenario precedence graphs corresponding to the sequence graphs in 8.2.4. In the following section, a global precedence graph will be constructed. While the scenario precedence graphs refer to a specific use case, the global precedence graph represents the behaviour of the overall system. The global precedence graph is obtained by combining the specific scenario precedence graphs. Combination is performed with the following operation.\(^{72}\) Let \(PG(h)\) denote the result of merging \(PG(i)\) and \(PG(j)\), in symbols:

\[
PG(h) = PG(i) \oplus PG(j)
\]

where \(\oplus\) denotes the merge operation. If \(PG(i)\) consists of \((g_1, g_2, g_3)\) and \(PG(j)\) consists of \((g_4, g_5, g_6, g_7)\), then \(PG(h)\) is the precedence graph consisting of four sub-graphs \((g_1, g_2, g_3, g_4)\) connected by a CASE node as illustrated in Figure 41. The general algorithm for producing a global precedence graph, denoted \(gPG\), from the specific scenario precedence graphs \((PG(1), PG(2), \ldots, PG(n))\) is stated below:

**BEGIN**

\[
gPG := PG(1)
\]

\[
For\; i = 2, \ldots, n \Rightarrow gPG := gPG \oplus PG(i)
\]

**END**

\[\text{Figure 41. Merge operation of precedence graphs.}^{73}\]

---


\(^{73}\) Ibid.
8.4.1 Scenario precedence graphs

The derivation of the corresponding scenario precedence graphs is performed by translating the semantics of the sequence diagrams into those of the precedence graphs. The objects in the sequence diagrams correspond to the actors of the precedence graphs. Each call from or execution of an actor is substituted for a node. A complete set of nodes and descriptions are presented in Table 12, where $A_i$, $MA_i$, $MB_i$, $MC_i$, $MD_i$, $ME_i$, and $MF_i$ denote the nodes of the application and Module A through F in specified order.

<table>
<thead>
<tr>
<th>Actor</th>
<th>Node</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>$A_0$</td>
<td>Call to Module A: request</td>
</tr>
<tr>
<td></td>
<td>$A_1$</td>
<td>NOP</td>
</tr>
<tr>
<td>Module A</td>
<td>$MA_0$</td>
<td>Execution: validate_user</td>
</tr>
<tr>
<td></td>
<td>$MA_1$</td>
<td>Call to Module B: method_1</td>
</tr>
<tr>
<td></td>
<td>$MA_2$</td>
<td>Call to Module B: method_2</td>
</tr>
<tr>
<td></td>
<td>$MA_3$</td>
<td>Call to Module E: method_3</td>
</tr>
<tr>
<td></td>
<td>$MA_4$</td>
<td>Call to Module D: method_4</td>
</tr>
<tr>
<td></td>
<td>$MA_5$</td>
<td>SPLIT</td>
</tr>
<tr>
<td></td>
<td>$MA_6$</td>
<td>Call to Module F: method_5</td>
</tr>
<tr>
<td></td>
<td>$MA_7$</td>
<td>Call to Module F: method_6</td>
</tr>
<tr>
<td></td>
<td>$MA_8$</td>
<td>Execution: method_7</td>
</tr>
<tr>
<td></td>
<td>$MA_9$</td>
<td>Call to Module B: method_8</td>
</tr>
<tr>
<td></td>
<td>$MA_{10}$</td>
<td>Call to Module C: method_9</td>
</tr>
<tr>
<td></td>
<td>$MA_{11}$</td>
<td>NOP</td>
</tr>
<tr>
<td>Module B</td>
<td>$MB_0$</td>
<td>Execution: method_1</td>
</tr>
<tr>
<td></td>
<td>$MB_1$</td>
<td>Execution: method_2</td>
</tr>
<tr>
<td></td>
<td>$MB_2$</td>
<td>Execution: method_8</td>
</tr>
<tr>
<td>Module C</td>
<td>$MC_0$</td>
<td>Execution: method_9</td>
</tr>
<tr>
<td>Module D</td>
<td>$MD_0$</td>
<td>Execution: method_4</td>
</tr>
<tr>
<td>Module E</td>
<td>$ME_0$</td>
<td>Execution: method_3</td>
</tr>
<tr>
<td>Module F</td>
<td>$MF_0$</td>
<td>Execution: method_5</td>
</tr>
<tr>
<td></td>
<td>$MF_1$</td>
<td>Execution: method_6</td>
</tr>
</tbody>
</table>

Table 12. Nodes of the precedence graphs grouped into subset for each actor.

The first precedence graph illustrated in Figure 42 refers to UC1. All calls made by an actor are synchronous and no splitting of control exists. Synchronous calls refer to where a client module requests service from a server, while being blocked to continue with its next task until a reply is received. The nodes $A_i$ and $MA_{11}$ are NOP nodes merely returning the original call.
The second scenario precedence graph refers to UC2 and is illustrated in Figure 43.

The precedence graph referring to UC3 is illustrated in Figure 44. The graph introduces the use of a SPLIT node \( MA_3 \) that splits the control to the application and Module A. The call to the Module F is asynchronous allowing Module A to continue with its next task.
Figure 44. Precedence graph corresponding to UC₃.

The last scenario precedence graph refers to the last use case UC₄ and is illustrated in Figure 45.
8.4.2 Global precedence graph

The global precedence graph is derived from the scenario precedence graphs presented in the previous section, by the algorithm in 8.4. The different use cases illustrated in each scenario precedence graph are summoned in a CASE node, dividing the four use cases.
8.5 Extended flow graph

The extended flow graph describes the behaviour of the system in a more detailed manner than the global precedence graph. It illustrates, like the precedence graph, the direction of the control flow for each use case, but also gives a direct notion of calls and executions. As stated in 7.3.1 the extended flow graph is generated from the global precedence graph, the workload data from the operational profile, the software modules architecture and the client/server structure. The semantics of the extended flow graph is derived from the global precedence graph, however the notation is somewhat different. The graph is constructed by converting LOOP, SPLIT and CASE nodes from the global precedence graph into repetitive, split and...
case blocks. Each node of the global precedence graph, in the subset belonging to an actor acting as a client according to the client/server structure (see 8.3.2), is converted into a basic block. Each node acting as a server is converted to a software server block. The resulting graph is presented in Figure 47. The illustration of the graph is simplified by introducing two subroutines, which refer to similar behaviours in two different use cases. The subroutines SUB1 and SUB2 are called in the branches corresponding to UC1 and UC4. The probability of each use case from Table 9 is associated with the corresponding branch of the CASE block.

Figure 47. Illustration of the extended flow graph of the considered system.
8.6 Extended execution graph

The extended execution graph of the considered system is a direct derivation of the extended flow graph with the extension of being weighted with its resource demand. By defining hardware partitioning, modules mapping data and the estimated or predicted resource demand, a resource demand vector may be constructed:

$$\bar{d} = (d_1, \ldots, d_n)$$

for each block in the extended execution graph, where $d_i$ is the number of elementary operations the block demands from device $i$ of the platform configuration in 8.2.3. Depending on the device, the sort of the $d_i$ might be the number of program statements executed by the CPU, the number of accesses to the database or the number of accesses to a LAN or a WAN. As only one modules-platform configuration is covered in this thesis, a single EEG, which is equal to the EFG being weighted with the resource demands, exists. The derivation of the resource demand matrix is presented in the next section.

8.7 Layered Queuing Network

As described in section 7.4, the basic LQN components are easily identified from the extended execution graph. To facilitate and clarify the appearance of the LQN model, the model is separated into a software contention model and a hardware contention model. The software contention model is used to illustrate the relationships in the software architecture, whereas the hardware contention model illustrates the competition for access to hardware
amongst software tasks. The scenario, of which the LQN is aimed to provide insight, is based on:

- The variation of the number of users in the system.
- Variations in service time of modules affected by adding new functionality to the system.

The software contention model of the LQN is generated using three steps. The first step is concerned with the identification of software tasks. The second step is concerned with deciding the multiplicity of each software task. Finally, the third step is concerned with the identification of entries for each software task. Similarly, two steps are presented for generating the hardware contention model of the LQN. The first step concerns the identification of hardware devices. The second and last step concerns the identification of number of device calls.

**Step 1. Identification of software tasks**

The software tasks are derived from the software blocks identified in the EFG. The LBS system includes the following software tasks: Application, Module A, Module B, Module C, Module D, Module E, and Module F.

**Step 2. The software tasks are derived from the software blocks identified**

Each software task is associated with a degree of multiplicity. For software tasks defined as active servers, i.e. acting both as client and server, the multiplicity is obtained directly from the EFG graph. For the software tasks that are pure clients the multiplicity is obtained from the workload data. The only sole client task in the model is the Application software task, and its multiplicity is decided by the number of simultaneous users of the system, which is denoted as N.

<table>
<thead>
<tr>
<th>Software tasks</th>
<th>Role</th>
<th>Multiplicity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Client</td>
<td>N</td>
</tr>
<tr>
<td>Module A</td>
<td>Client/Server</td>
<td>m</td>
</tr>
<tr>
<td>Module B</td>
<td>Server</td>
<td>n</td>
</tr>
<tr>
<td>Module C</td>
<td>Server</td>
<td>n</td>
</tr>
<tr>
<td>Module D</td>
<td>Server</td>
<td>2n</td>
</tr>
<tr>
<td>Module E</td>
<td>Server</td>
<td>n</td>
</tr>
<tr>
<td>Module F</td>
<td>Server</td>
<td>n</td>
</tr>
</tbody>
</table>

*Table 13. The software tasks of the LQN and their roles as pure client, active server, or pure server.*

---

**Step 3. Identification of entries**

The different entries associated with each task are identified by all the called methods (client) and received methods (server) defined in the EFG. This step includes also pairing the called and received methods to each other. In the LQN this will be depicted as directed arcs to and from the entries.

<table>
<thead>
<tr>
<th>Software tasks</th>
<th>LQN parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Entry</td>
</tr>
<tr>
<td>Application</td>
<td>request</td>
</tr>
<tr>
<td>Module A</td>
<td>request</td>
</tr>
<tr>
<td></td>
<td>validate_user</td>
</tr>
<tr>
<td></td>
<td>method_1</td>
</tr>
<tr>
<td></td>
<td>method_2</td>
</tr>
<tr>
<td></td>
<td>method_3</td>
</tr>
<tr>
<td></td>
<td>method_4</td>
</tr>
<tr>
<td></td>
<td>method_5</td>
</tr>
<tr>
<td></td>
<td>method_6</td>
</tr>
<tr>
<td></td>
<td>method_7</td>
</tr>
<tr>
<td></td>
<td>method_8</td>
</tr>
<tr>
<td></td>
<td>method_9</td>
</tr>
<tr>
<td>Module B</td>
<td>method_1</td>
</tr>
<tr>
<td></td>
<td>method_2</td>
</tr>
<tr>
<td></td>
<td>method_8</td>
</tr>
<tr>
<td>Module C</td>
<td>method_9</td>
</tr>
<tr>
<td>Module D</td>
<td>method_4</td>
</tr>
<tr>
<td>Module E</td>
<td>method_3</td>
</tr>
<tr>
<td>Module F</td>
<td>method_5</td>
</tr>
<tr>
<td></td>
<td>method_6</td>
</tr>
</tbody>
</table>

**Table 14. Identification of entries of software tasks.**

Similarly, each step for generating the hardware contention model is performed for the LBS system. The steps are presented below.

**Step 1. Identification of devices**

The devices, of which the software tasks are competing for, are derived from the platform configuration of the system. Accordingly, the devices are WAN and HWS.

**Step 2. Identification of number of device calls**

Each entry resulting in an entry of execution type is identified as a call to a device. The entries to the devices are presented in Table 15:
8.7.1 Parametrization

In the parameterization values are estimated for each method in terms of:

- Number of calls for each method. According to Cortellessa this is defined as the Method Call Parameter – MCP.
- Number of units of execution for each method. This is defined as Device Call Parameters – DCP. According to Cortellessa the DCP contains the number of CPU instructions. In the modelling approach in this thesis it has been more feasible to model the execution time in each block instead. The measurements are derived from an existing version of the software application, which is already implemented. The measurements are in some cases (for new functionality) estimated or extrapolated by simple prototyping and educated guesses by system experts. The parameters are used to make the resource vectors of the EFG blocks. For the methods that are calls, the execution time is the time sending the signals. Due to the risk of compromising the intellectual property rights of Wayfinder Systems AB, the execution times are excluded from Table 16.
The derivation of the number of visits to each entry of the LQN, involves summoning each block of the EFG weighted by the probabilities of the use cases. The total service time for each entry is calculated by:

$$D_p \cdot E_{\text{Per Entry Service Time}} = \overline{p} \cdot \overline{D}$$

where $\overline{p} = (p_1, \cdots, p_n)$ denotes the probabilities of all use cases, and $\overline{D} = (D_1, \cdots, D_n)^T$ denotes the total resource demand of each entry per use case. The resource demand per use case is calculated by summoning the resource demand vectors for all blocks in the corresponding branch in the EFG, i.e. $D_i = \sum_{\text{visits}} d_j$

The software contention model is given by performing step 1, step 2, and step 3 described in the previous section. The model is presented in Figure 49. The hardware contention model illustrated in Figure 50 is given by performing step 1 and step 2 also described in the previous section.

<table>
<thead>
<tr>
<th>Modules</th>
<th>LQN parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Entry</td>
</tr>
<tr>
<td>Application</td>
<td>request</td>
</tr>
<tr>
<td>Module A</td>
<td>request</td>
</tr>
<tr>
<td>validate_user</td>
<td>method_1</td>
</tr>
<tr>
<td>method_2</td>
<td>$p_{UC2} + p_{UC4}$</td>
</tr>
<tr>
<td>method_3</td>
<td>$p_{UC1}$</td>
</tr>
<tr>
<td>method_4</td>
<td>$p_{UC3}$</td>
</tr>
<tr>
<td>method_5</td>
<td>$p_{UC3}$</td>
</tr>
<tr>
<td>method_6</td>
<td>$p_{UC2}$</td>
</tr>
<tr>
<td>method_7</td>
<td>$p_{UC2}$</td>
</tr>
<tr>
<td>method_8</td>
<td>$p_{UC2}$</td>
</tr>
<tr>
<td>method_9</td>
<td>$p_{UC2} + p_{UC4}$</td>
</tr>
<tr>
<td>Module B</td>
<td>method_1</td>
</tr>
<tr>
<td>method_2</td>
<td>$p_{UC2} + p_{UC4}$</td>
</tr>
<tr>
<td>method_8</td>
<td>$p_{UC2}$</td>
</tr>
<tr>
<td>method_9</td>
<td>$p_{UC2} + p_{UC4}$</td>
</tr>
<tr>
<td>Module C</td>
<td>method_9</td>
</tr>
<tr>
<td>Module D</td>
<td>method_4</td>
</tr>
<tr>
<td>Module E</td>
<td>method_3</td>
</tr>
<tr>
<td>Module F</td>
<td>method_5</td>
</tr>
<tr>
<td>method_6</td>
<td>$p_{UC2}$</td>
</tr>
</tbody>
</table>

Table 16. LQN parameters of the LBS system.
Figure 49. The software contention model of the LQN of the LBS system.

Figure 50. The hardware contention model of the LQN of the LBS system.
9 Results from performance evaluation

The results from the performance evaluation are presented in this chapter. The evaluation aims at identify bottlenecks and response time based on variations of the number of users and service time of software modules.

As described in earlier chapters, the performance of the LBS system is important. The main interest is to investigate the behaviour of the system when new functionality is added to the system resulting in changed execution times for software modules, and when the load is drastically increased. For instance, when the number of users is increased it is possible to identify which software modules are bottlenecks. Another interesting aspect is to provide performance boundary for the new functionality added to the system. When choosing to implement a vector representation of the spatial data, the complexity of the system increases resulting in increasing the execution time for a certain software module.

9.1 Performance scenarios

To begin the investigation two performance scenarios are chosen:

1. How does the response time vary when the number of users requesting service increases?
2. What are the performance bounds for the new software module that is added to the system?

The first performance scenario addresses the response time and to identify which parts of the system is critical and can be identified as bottlenecks. From a performance point of view, it is important to identify these limiting parts, which parts are requesting service from them, and how frequent they are requested. When the load is scaled the effects on response time is non-linear. The second scenario aims to investigate the performance bounds for adding functionality in a specific software module. If bottleneck identification discovers contention problems involving the module, the complexity of the new functionality must be addressed.
9.2 Results

The methodology for automatic generation of a performance model proposed by Cortellessa requires input to define the software and hardware architecture and their interaction amongst themselves as well as user behaviour mapping. The input data could be class diagrams, sequence diagrams, platform and software data, operational profile, client/server structures, and software/hardware mapping. The data collection from Wayfinder Systems AB is carried out through measurements, interviews with experts and archive analysis. The archive analysis is carried out by going through legacy documentation of the system. Input parameters to the LQN concerning the new functionality were based on educated estimations.

Solving the LQN model was fairly easy, the parameters of the LQN model presented in 8.7.1 were used as input to the tool LQNS either by a graphical interface of by a command line interface.\textsuperscript{75} The LQN was solved with low effort for different values of the number of users to the system (N) and different execution times for different tasks. The tool provided a flexible approach and adapted well to the desired scenarios.

Unfortunately, the figures resulting from solving the LQN model cannot be presented in the thesis due to the fact that actual figures of the execution times for different task would compromise the business case of Wayfinder System AB and their competitive advantage against their competitors. Without revealing any incentive figures, the following can be said. The module where new functionality is added proved not to be the initial bottleneck. Instead, the investigation identified that contention resulting in considerable delays appeared in another software module. However, when increasing the execution time (to simulate an increase in complexity) of the module, which concerns the implementation of the vector format, an upper performance bound was found. This indicates that resource and time consuming tasks such as redundancy checks and real-time data simplification should be avoided (see discussions 5.3.1 in and 5.5). As expected, the WAN being modeled as a device turned out to be a bottleneck by providing a delay. Therefore, it is critical to reduce the data size of the data sets or break down the data sets to smaller data packets. Other useful findings by generating the performance model was the identification and scheduling of which software tasks required service from other tasks and devices and clarification of the frequency of the requests. The information is useful when deciding on the software design of the implementation of the vector format. For instance, the uniqueness of each request proved to be important. One alternate design proposal was to introduce sessions in the server in order to simplify requests from the application. Keeping sessions of a user equals to the introduction of state recognition, i.e. the server keeps track of the user’s state and performs suitable tasks

accordingly. By using sessions, a real-time redundancy check can be performed since the server “remembers” earlier requests made by the specific client application. However, the use of sessions would drastically limit the amount of simultaneous processes, interpreted as multiplicity in the LQN model. The connection to the arrival intensity is evident, i.e. the system could only serve fewer customers per time unit. When changing the multiplicity and feeding new parameters to LQNS, the throughput was immediately affected. Therefore, instead of sessions each request should be state-less.

9.3 Issues

When deriving the performance model, some granularity issues were encountered. More specifically, the difficulties arose when deciding on the verbosity of the model. The requirements on accuracy determine the level of verbosity, but still it might prove that features in the low-level design introduce features relevant to the performance evaluation not described by the high-level design. The correlation of low-level design and LQN parameters are evident. Therefore, it is difficult to completely avoid ad hoc methods. However, it is feasible to achieve an adjustable degree of abstraction by adding or removing blocks in the intermediate graphs. The difficulty lies in determining the verbosity level from the expected answers from the performance scenarios as well as being able to perform measurements on the system.

Another issue affecting the outcome of the results of the analytical tool as well as the possibility to provide useful feedback to the design proposals was the inability to model the dynamic load balancing implemented in the server. The analytical tool LQNS introduced limitations in modeling the load balancing. It would be most desirable to be able to complement the LQN with event-driven simulations, which could be used as feedback to the LQN model.
9 Results from performance evaluation
This chapter will present conclusions from the two activities presented in previous chapters and the reflections from previous chapter. At last, a discussion of future work will be given.

Introducing vector graphics to represent maps can effectively enhance user interactions with spatial data by allowing the user to zoom instantly, rotate, and get additional data of points of interest. For instance, the ability to process the image in the mobile device makes offline work possible. The objective of the thesis was to evaluate and analyze an efficient vector representation of geospatial data suited for visual presentation in a mobile environment. Further, the study aimed to derive a number of design proposals for a representation of spatial data suitable for, in addition to visual presentation, also an efficient transfer over wireless connections with limited transfer rate. In order to derive the design proposals a number of requirements were defined. The requirements included functional, cartographic, software design, and performance demands.

To meet the defined requirements two activities were derived. The first activity involved an evaluation of existing vector formats to be able to evaluate useful features and their impact on the LBS system. The evaluation resulted in a comparison between the formats. Several useful insights were gathered, e.g. the transfer of multi-resolution data etc. The second activity was to generate a performance model of the LBS application. The model was used to investigate the performance demands and provide useful insights in the derivation of the key success factors. Due lack of time and restrictions in the analytical tool the number of useful insights from the salvage of the performance model was limited. However, the performance evaluation still provides some very interesting aspects which were reflected upon when deriving the design proposals.

The result of the study was a number of design proposals expressed as implementation-ready proposals. The proposals and a short summary are given below.

Proposal 1. Object identification
It is necessary to be able to identify each object in the map. By identification additional data can be linked to the object providing the user with traffic-aware information, phone number and address, or URL to a website providing external information of the object. A suitable data structure called abstract cell complexes are suggested.
Proposal 2. Progressive vector transmission
Progressive vector transmission reduces the time from requesting a map to the visualization of the map. Progressive vector transmission can be accomplished by adopting two different approaches. The first is progressiveness in drawing order, where the most important objects are transferred prior to objects of less importance. The second approach is more sophisticated, transmission occurs in multiple-resolutions.

Proposal 3. Generalization of geospatial data
The purpose of generalization of geospatial data is to reduce the amount of data to obtain a graphical representation of a map suitable for interpretation by a user. The generalization can be achieved by simplification, selection, symbolization, exaggeration, displacement, and aggregation.

Proposal 4. Coordinate representation
Spatial data typically consist of a large amount of coordinates to describe objects in the map. By using a clever coordinate quantization, the size of the spatial data can be drastically reduced, which enables an efficient transfer over wireless connections. The quantization proposals are suggested.

Proposal 5. Spatial data storage
By using a spatial database dedicated to visual presentation, optimizations in avoiding data redundancy, multi-resolution storage etc. may lead to minimizing the access time for spatial data.

Proposal 6. Semantic caching
Caching frequently accessed data in the application improves performance especially in mobile environments. Semantic caching refers to objects in the map being accessed individually from the cache instead of traditional caching, which treats the entire map as a cache item. Semantic caching requires an object-oriented map model.

Proposal 7. Threaded data processing
In applications in interactive environments, multithreading is an efficient approach to enhance utilization of hardware. To implement requests of data and rendering of data in separate threads, the user may get a faster visual feedback. Threads allow low cost context switching in the application.

Proposal 8. Compression techniques
The size of vector map data is typically quite large. Compression is a technique to further reduce the size and thus reducing the cost of communication in mobile environments.
Traditional techniques such as LZW may be useful. However, another technique suited in the vector domain is suggested. By using dictionary compression applied to differential vectors the size of a map may be significantly reduced by allowing an approximation error. Since approximation errors already may be introduced by map generalization, this is acceptable.

Proposal 9. Cartographic appearance
Adding cartographic data to spatial data facilitates the visual interpretation of a map. However, by adding it the size may increase. Therefore, cartographic data should be kept separate from spatial data. This can be achieved by defining cartographic map settings, which can be sent to the application more seldom. By updating the settings, the appearance of the map can be completely changed without needing to patch the application software.

Proposal 10. Design approach
A design model to implement the proposals, which are located in the server side, is suggested. The model consists of several layers; kernel, transformation layer, model-oriented generalization layer, cartographic generalization layer, compression layer, drawing layer, and application layer.

The proposals are ready to be implemented. During the evaluation some of the proposals were implemented and tested. The test suite included the server at Wayfinder Systems AB and a Java application simulating the application in the mobile device. The proposals implemented were polyline simplification according to the Hershberger-Snoeyik algorithm, FHM vector compression, semantic caching, and threaded data processing in the application. The conclusions from the implementation are purely functional, since the properties of the mobile environment were not simulated. Non-functional requirements such as performance were not tested since the limited transfer rate between application and server was not simulated. The performance modeling was used to fill that void. The Java 2D API presents a complete and straightforward implementation of the rendering as well as multithreading. However, Java is an interpreted language and the Java 2D API provides poor rendering performance compared to native graphics libraries, e.g. in Symbian OS. Java also uses garbage collection, which might cause pauses in the rendering process. However, the conclusions drawn from the test implementations are reflected in the suggested proposals.

It is interesting to reflect upon the projected lifetime of the system. The design proposals are derived based on the premises of the unique properties of mobile internet. Therefore, the implementation effort should be kept at a minimum to ensure a short time to market. The development of the mobile environment is progressing in an impressive pace. The launch of UMTS enhances the transfer rate of the wireless connection and therefore the use of
progressive transmission may soon be outdated. However, GSM will most probably possess a strong market share in the near future, which will guarantee the demand for the above proposals.

10.1 Future work

In addition to the map generalization process, an analysis of adding third party data to the spatial data should be interesting. How should the data be connected if the spatial data is highly generalized and the third party data is linked to the highest resolution level? This is very common for example in route data, GPS data etc.

To further optimize the spatial data selection process, a cognitive evaluation of the visualization of the map could be performed. The evaluation could determine and prioritize data that is most important for the visual interpretation. It could also provide useful feedback when defining map generalization rules such as abstraction and displacement operations.

It would also be desirable in future work to further evaluate performance requirements during the implementation of the design proposals. Also, it would be interesting to complement the analytical solvage of the LQN by means of simulation. The issues encountered in the performance model generation would be interesting to address. For instance, the development of a methodology to standardize the verbosity level of the performance model from design documents would be most desirable.
References

Published

Books


Articles


**Internet references**


Adobe (2003) *Adobe and SVG*, available at:


BitFlash (2003) *BitFlash Mobile SVG Player & SDK*, available at:


Network Working Group (1996) *Gzip file format specification version 4.3*, available at:

http://www.cmg.org/measureit/issues/mit04/m_4_5.html#6, 030620.


W3C (2001) *Scalable Vector Graphics (SVG) 1.1 Specification*, available at:

W3C (2003) *SVG 1.1 and SVG Mobile Frequently Asked Questions*, available at:
http://www.w3.org/2003/01/svg11-faq, 030711.

XMill (1999) *An Efficient Compressor for XML*, available at:
<table>
<thead>
<tr>
<th>Acronyms and abbreviations</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td></td>
<td><em>The main processor in a computer that executes all calculations.</em></td>
</tr>
<tr>
<td>CSS</td>
<td>Cascading Style Sheet</td>
</tr>
<tr>
<td></td>
<td><em>Stylesheet language primarily developed for use of HTML, but also used together with XML documents.</em></td>
</tr>
<tr>
<td>DOM</td>
<td>Document Object Model</td>
</tr>
<tr>
<td></td>
<td><em>A language and platform independent interface allowing access to elements inside a XML document from scripts and programs.</em></td>
</tr>
<tr>
<td>EFG</td>
<td>Extended Flow Graph</td>
</tr>
<tr>
<td></td>
<td><em>Graph used in the step-wise derivation of a performance model.</em></td>
</tr>
<tr>
<td>EEG</td>
<td>Extended Execution Graph</td>
</tr>
<tr>
<td></td>
<td><em>Used in the step-wise derivation of a performance model. A weighted EFG with the resource demand of a module-platform mapping.</em></td>
</tr>
<tr>
<td>GIS</td>
<td>Geographic Information System</td>
</tr>
<tr>
<td>GPRS</td>
<td>General Packet Radio Service</td>
</tr>
<tr>
<td></td>
<td><em>Packet switched data enhancement of GSM.</em></td>
</tr>
<tr>
<td>GSM</td>
<td>Global System for Mobile communication</td>
</tr>
<tr>
<td></td>
<td><em>Digital mobile cellular communication standard. Operational on 900 MHz and 1800 MHz.</em></td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HSCSD</td>
<td>High Speed Circuit Switched Data</td>
</tr>
<tr>
<td></td>
<td><em>Circuit switched data enhancement of GSM.</em></td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language</td>
</tr>
<tr>
<td></td>
<td><em>Markup language used for creating pages for the World Wide Web.</em></td>
</tr>
<tr>
<td>HTTP</td>
<td>HyperText Transfer Protocol</td>
</tr>
<tr>
<td></td>
<td><em>Markup language in used in the World Wide Web.</em></td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>LBS</td>
<td>Location-based service</td>
</tr>
</tbody>
</table>
| LQN     | Layered Queuing Network  
*An extension of Queuing Networks, which capture complex client/server interactions in distributed systems.* |
| MMS     | Multimedia Message Service  
*An extension of SMS to include multimedia features.* |
| PDA     | Personal Digital Assistant |
| SD      | Sequence diagram  
*UML interaction diagram.* |
| SMS     | Short Message Service  
*Technique for transferring short text messages in GSM networks.* |
| SOAP    | Simple Object Access Protocol  
*Protocol based on XML for exchanging information in a distributed environment.* |
| SPE     | Software Performance Engineering  
*Discipline within software engineering, which focuses on integrating software performance in software development.* |
| SVG     | Scalable Vector Graphics  
*Markup language in the XML family, developed for presentation of vector graphics.* |
| SVG Basic | *SVG profile based on SVG 1.1 for higher-level mobile devices such as PDAs.* |
| SVG Tiny | *SVG profile based on SVG 1.1 for lower-level mobile phones.* |
| W3C     | World Wide Web Consortium |
| WAP     | Wireless Application Protocol  
*Set of specifications for wireless Internet access.* |
| UML     | Unified Modelling Language |
| XML     | eXtensible Markup Language  
*A metalanguage for developing markup languages.* |