Usable and well scaled maps for consumers

Work Package 1

Delft, September 2007

RGI 233 Report Nr 1

Usable and well scaled maps for consumers

Work Package 1

Delft, September 2007

RGI 233 Report Nr 1

Summary

Today we see a huge increase of the use of geo-information in mobile devices. All current solutions are based on static copies that are stored on the mobile device. This makes dynamically adapting the map to new information and to the changing circumstances of the user impossible. With the availability of high bandwidth wireless connections (such as UMTS) better, more dynamic, solutions are possible: The server generates a proper, up-to-date map of the region of interest at the right level of detail for display and adjusted to the needs of the user. For a mass market (consumers of mobile maps) the human factors aspect is very important. The currently available mobile maps solutions still have insufficient user-interfaces. Extremely important is the issue of context as the user gets 'lost' very easily on the small mobile displays when zooming and panning. Based on a selection of use cases (navigation, tourist support, etc.), User-Centered Design techniques will be applied to develop small prototypes / simulations and the interaction and the quality of the maps in these prototypes / simulations will be evaluated.

The project runs from 2006 to 2008 and is organized in the following Work Packages:

- WP 1 'Preparation' (month 4-9, 2006);
- WP 2 'Prototype development' (month 10, 2006-6, 2007);
- WP 3 'Evaluation of first prototypes' (month 7-12, 2007);
- WP 4 'Improved prototypes' (month 1-6, 2008);
- WP 5 'Evaluation of improved prototypes' (month 7-12, 2008).

This report contains the combined results (originally separate reports were planned), of this preparation. It contains a description of the use cases, an analysis of user requirements, and overviews of relevant literature and available or proposed technology.

Project leader	:	Section GIS Technology OTB Research Institute for Housing, Urban and Mobility Studies Delft University of Technology
Address	:	Jaffalaan 9 2628 BX Delft
Contact	:	prof.dr.ir. P.J.M. van Oosterom
Tel.	:	+31 (0)15 278 6950
Fax	:	+31 (0)15 278 2745
E-mail	:	p.j.m.vanoosterom@tudelft.nl
Website	:	http://www.rgi-otb.nl/pages/uwms2

Contents

1 Introduction

"Usable and well scaled mobile maps for consumers" (UWSM2) fits perfectly the intentions of RGI. The project starts with questions, users of geo-information would like to see answered: ANWB is interested in an application for mobile tourist information; Amsterdam would like to have a mobile application for parking in the city. Science (DUT, TNO, ITC) and industry (ESRI, 1Spatial) are working together, and in cooperating with the users, will try to provide answers.

Work Package 1 of UWSM2 is defined in the project plan as the preparation phase. This report contains the combined results (originally separate reports were planned), of this preparation. It contains a description of the use cases, an analysis of user requirements, and overviews of relevant literature and available or proposed technology.

In chapter 2, "User requirements" by "TNO Defense, Security and Safety", the user requirements are translated into a scenario that will be the target of the first prototype (Work Package 2). Descriptions of the initial use cases, provided by Amsterdam and ANWB, can be found in Appendices 1 and 2. Early on in the project it was decided to make a clear distinction between the two prototypes to be developed. The first prototype should concentrate on the generic aspects of the use cases, and it will be developed in a simulated ('desktop') environment. The second prototype (Work Package 4) will focus more on the specific requirements of the intended applications, and it must be available on real mobile devices. So in the scenario only the generic elements of the use cases are included. In the remainder of chapter 2 some techniques to describe and model use cases are introduced.

Chapter 3, "Usability of mobile maps" also by "TNO Defense, Security and Safety", is essentially a literature review of "Usability Engineering", concentrating on the use of maps on small mobile devices. The current state-of-the-art of the research in this discipline is described, and an overview of the challenges for mobile mapping is presented, together with a number of possible solutions. In the next phase of the project the most promising solutions will be implemented in the first prototype, and following that evaluated by user tests (in Work Package 3).

The emphasis in chapters 2 and 3 of this report is on the user side of the project: the functionality and look-and-feel of the applications the end-users will have available (eventually). But to make this possible some underlying technology is required, the assumption within the project is that current technology is not sufficient to provide a completely satisfactory solution. So the need exists for the development and deployment of new technology, this is the subject of chapters 4 and 5 of this report. As with any modern, distributed (networked) application the UWSM2 applications can be seen as client-server applications. On the mobile device the client part of the application runs, which primarily takes care of user interaction and visualization. The maps and other information to be displayed are retrieved from a server (or servers) by means of (wireless) network connections. Chapter 4 deals with server side technology, in chapter 5 client side technology is described.

In chapter 4, "Server side technology" by "Delft University of Technology", a new and specific solution for the creation and use of generalized maps is introduced. Small screens with limited resolution characterize mobile devices. Still a user needs to switch between scales (detail maps and overviews) with preferably smooth panning and zooming in between. This translates to a heavy use of generalized maps at various scales. Two solutions exist to support this. One is the multi-scale approach: generalized maps are created for a limited number of selected scales. For displays at intermediate scales a map must be 'interpolated' from the available scales above and below the required scale. In this project an alternative solution is proposed and developed: vario-scale mapping. The essential idea is that generalized maps can be stored in advanced, topological data structures (tGAP: topological Generalized Area Partitioning) in a database on a server. For any requested scale the map can be retrieved efficiently and without redundancy from the server. The tGAP structures and how to use them for vario-scale mapping are introduced and explained in chapter 4.

Finally in chapter 5, "Client side technology" by "ESRI NL", a short introduction is provided into some possible technical solutions for the client side. Partly the solution will be based on existing software technology. But if full use must be made of the potential power of tGAP structures, e.g. the use of progressive transfer of data, additional client side software will have to be developed (in later phases of the project).

2 User requirements

An important aspect of all usability engineering method is the early and strong focus on the user. We use a scenario-based approach to achieve this. Scenarios have been widely used in design in general and user interface design in particular to capture and represent tasks (Rosson et al., 2001). Although some writers have used the term more or less interchangeably with use cases, scenarios are generally quite different from use cases. Scenarios are typically extended narratives forming a plausible vignette or story-line. They tend to be rich, realistic, concrete, and specific, full of detail for enhanced realism. Scenarios are verbal descriptions of possible uses of the system, describing the user and his or her tasks, including his or her goals, desires, and the context of use. It describes how the user interacts with the device at a high level and what functionality is provided, but does not go into details about the implementation of user interfaces. Because the scenario presents a story about the use of the system without technical details and the use of complex diagrams, it provides a perfect means to discuss with users and to elicit their desires.

The scenario is the starting point for further specification of the desired functionality and interaction. Many different types of methods, approaches and models have been developed to do this, originating either from software engineering practice or from the human factors research field. The models that are used in this project are shown in the figure below.

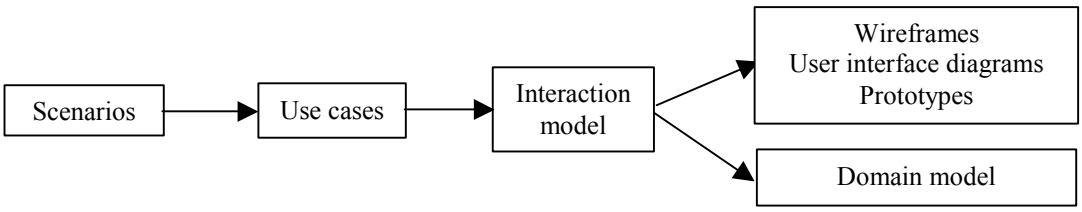


Figure 2.1: Models used in the project Usable Mobile Maps for Consumers.

Based on the scenario, use cases will be derived. Use cases describe frequently executed or complex tasks that need further specification because they are vital for the success of the to be developed system. Use cases can be an aggregate of smaller use cases, and can be related to each other. Next, the interaction between the system and the user that is required to complete a use cases is specified in more detail. Based on the results, interaction designers can design wireframes or (paper) prototypes that can be presented to stakeholders for comments or evaluation. Required system components and their architecture are specified in a domain model. A popular set of diagrams to model these different parts is provided by UML, the Unified Modelling Language. In this project, we will use use case diagrams to model our use cases, sequence diagrams to model the interaction between the user and the system, and class diagrams to describe the domain model. UML does not include specific diagram for user interfaces, hence we will most likely use wireframes or flash or paper prototypes. Another option is a visualization called user interface diagrams, which have been developed as an informal addition UML.

2.1 Scenario

The scenario is derived from two use cases defined by the local authority of Amsterdam and the Dutch Automobile Association (ANWB), see Appendices 1 and 2. In the scenario the requirements (table 1) that are derived from the sequence diagrams are filled out between brackets.

Sara wants to go to the Rijksmuseum in Amsterdam. She drives her car to the Museumplein car park, but the car park is full when she arrives. To search for another parking space she looks at her mobile device [Iv]. She would like to know where she can park her car in Amsterdam near the Rijksmuseum and how much it will cost [Is] [POI].

The map shows the parking tariffs in the surrounding of the Rijksmuseum [F] [Z]. Sara has indicated before by answering questions that she does not mind to walk for a little while when the sun shines (3 km) and with bad weather she still doesn't mind to walk 1 km [UM].

The system has looked up the weather forecast for today and it is going to rain. Sara thinks the rain is not that bad and indicates to the system that she would also like to see the tariffs on 1 to 3 km walking distance [UM]. The system now looks at parking tariffs in the neighbourhood [F]. It shows on the screen that there is a cheaper tariff within 3 km walking distance [Sosv]. Sara sees that only at the other side of the Van Baerlestraat she can park for € 1.40 per hour less. She reviews the distance to the Rijksmuseum and decides to go there [Iv] [Z]. On her map she looks where it is one-way traffic and drives to the other side of the Van Baerlestraat. After parking her car she buys her ticket using her mobile phone. Because her phone has GPS she only has to call the service number. She buys her ticket and stores the location of her car [F] [Lst]. Now she likes to know where she is with respect to the Rijksmuseum and how to come there [Iv] [F] [Z].

After leaving the museum Sara goes shopping, but eventually she wants to go back to her car. Because she did walk pretty far she would like to see the direction of where her car is, instead of a map with her position and that of the car on it [Is]. Sara walks to her car [Iv] [Z] and leaves Amsterdam.

Table 2.1 shows the user requirements that were derived from the use-cases. Some general user requirements are that the icons and letter fonts should be readable and all the information should be easy to interpret.

User Requirement	**Description**
POI selection (POI)	The user must be able to select a POI in different ways (i.e. address, name, kind of POI)
Information selection (IS)	The user should have the ability to select information that has to be showed on the map
User model (UM)	Both user and device should be able to adapt the user model.
Location storing (Lst)	The user should be able to make her/his own POIs
Feedback (F)	The device should give feedback about the users actions
Zooming/panning (Z)	The user must be able too zoom/pan, but the device should do this also according to the user model
Show off screen visualization (Sosv)	The device shows the POIs that fall outside the screen
Information visualization (Iv)	The device shows e.g. the location of the user on the map, or the parking rates.

Table 2.1: User requirements for the parking support system.

2.2 Use cases

Figure 2.2 shows the main use case taken from the scenario, "go to point of interest (POI)". This use case is composed of five smaller use cases, each of which needs further specification. For more complex use cases, additional descriptions of use cases should be made, but because in this case everything was rather straightforward, we decided to start with the interaction modelling.

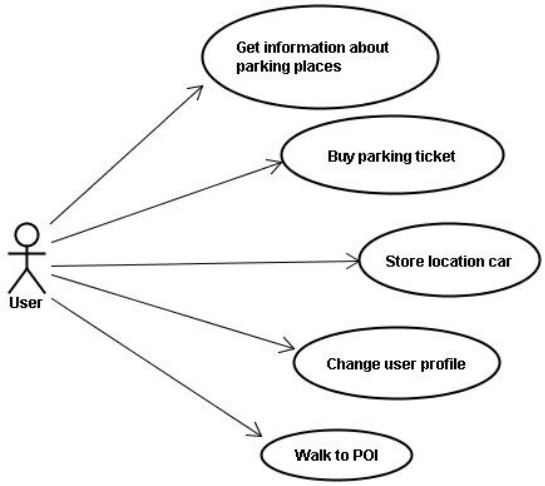


Figure 2.2: Use case diagram "Go to point of interest", consisting of 5 other use cases.

2.3 Interaction model

A first step that can be taken after the use cases have been described is the modelling of an activity diagram. Although it is possible to construct sequence diagrams to model interaction at a more detailed level straight away, starting with a more global activity diagram can provide some additional guidance in the process.
Figure 2.3 shows the activity diagram that was constructed based on the scenario and the use-cases. It provides a visual representation of the scenario, with a focus on the interaction between the user and the system. The sequence diagrams take this one step further, because different modules are distinguished inside the system. To model the system, the popular and well-known design pattern Model-View-Controller (MVC) was used. This pattern divided the system in three components:

1. Controller, which provides the means of input for the user,
2. Model, which contains the business logic of the system,
3. View, which provides the output of the system.

One of the main advantages of this model is that when a more advanced logic module is added, only the model component needs adjustment. The same holds when a device is used with a different screen, because then only the view component needs alterations. MVC has also been used in a slightly adjusted form in the domain of usable maps on mobile devices (Reichenbacher, 2003). A component called Context was added to contain the logic to make the map adaptive, although this could have been included in an enhanced Model component.

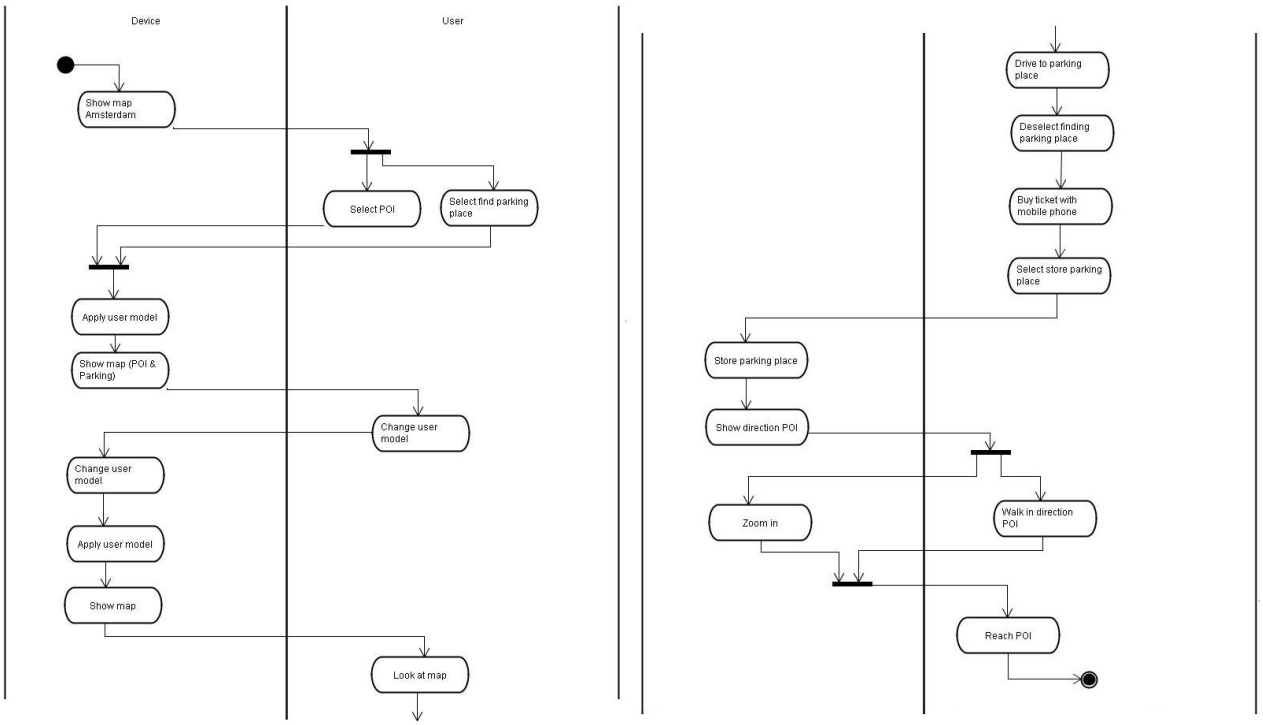


Figure 2.3: Activity diagram "Go to point of interest".

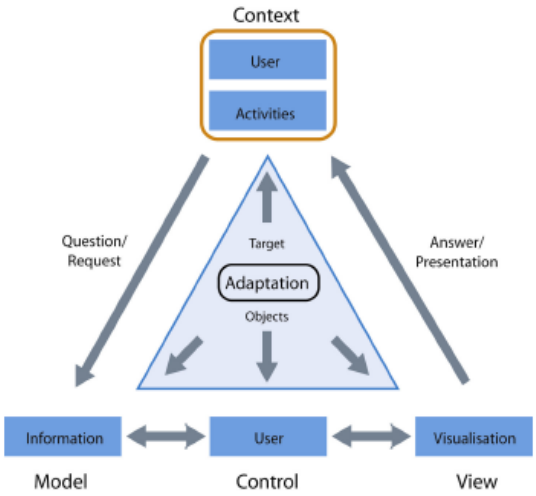


Figure 2.4: MVC by Reichenbacher (2003). Picture taken from Nivala (2005).

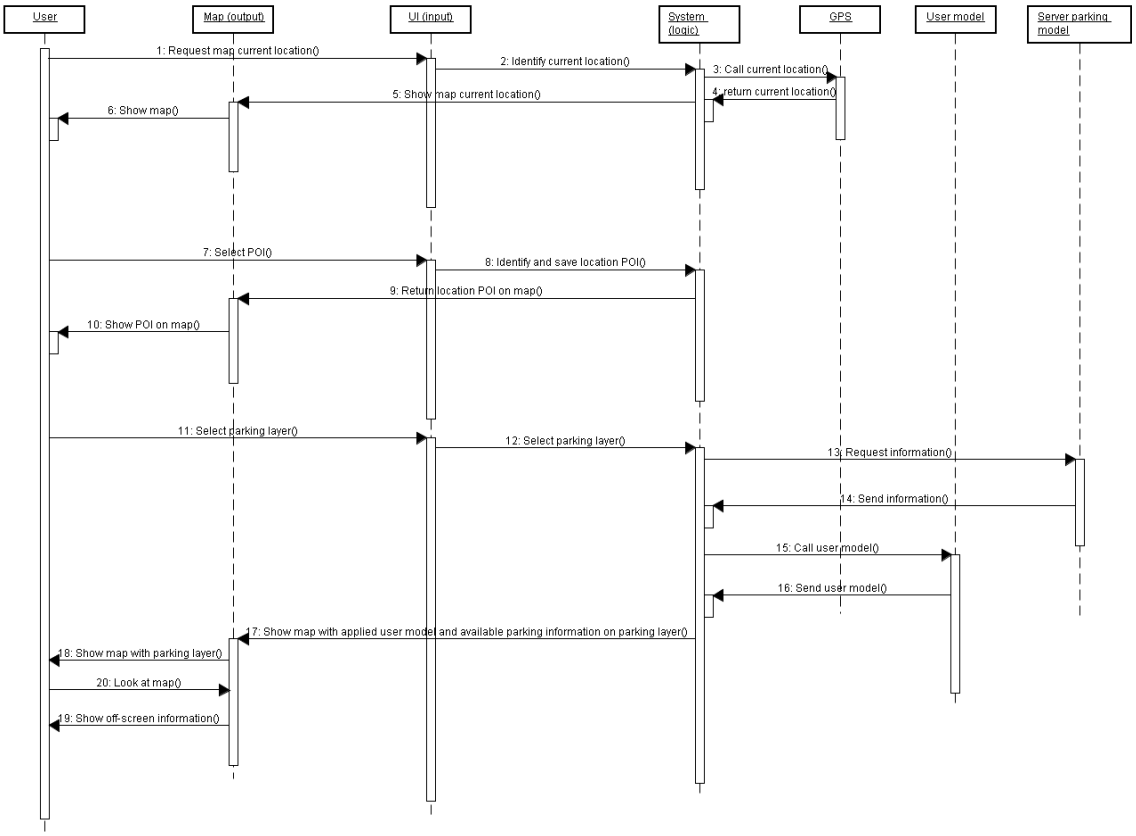


Figure 2.5: Sequence diagram: Get information about parking places.

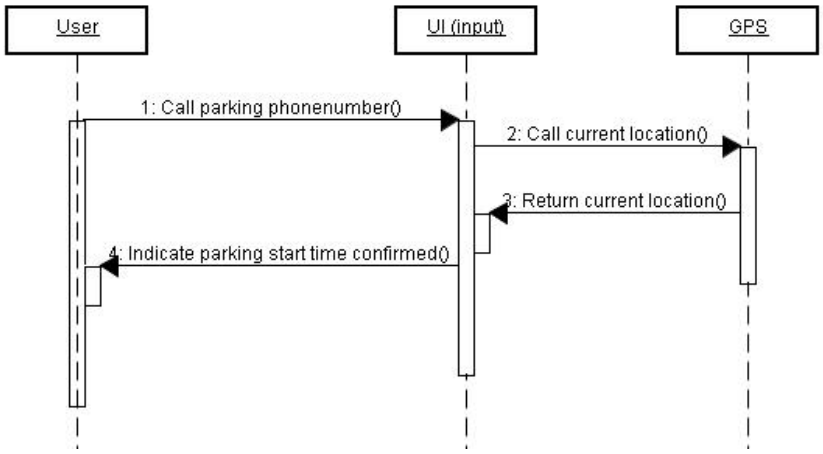


Figure 2.6: Sequence diagram: Buy parking ticket.

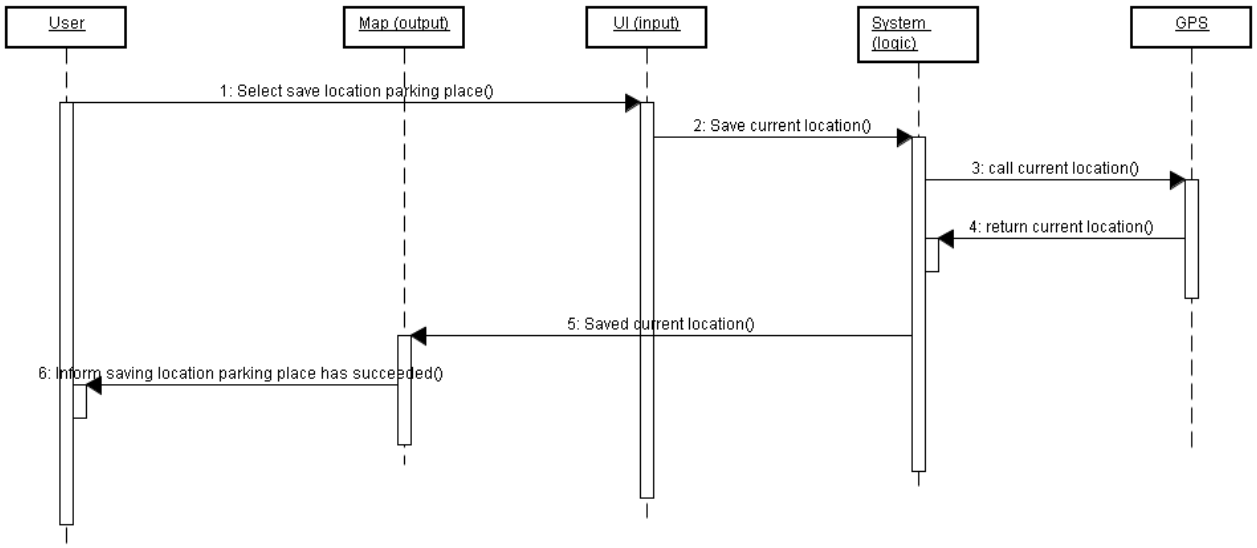


Figure 2.7: Sequence diagram: Store location car.

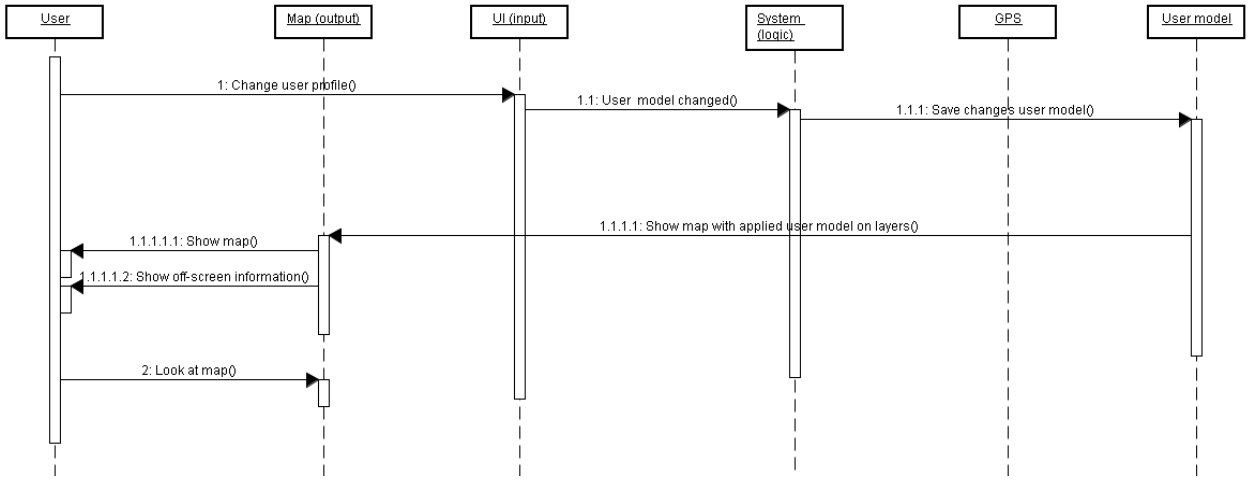


Figure 2.8: Sequence diagram: Change user profile.

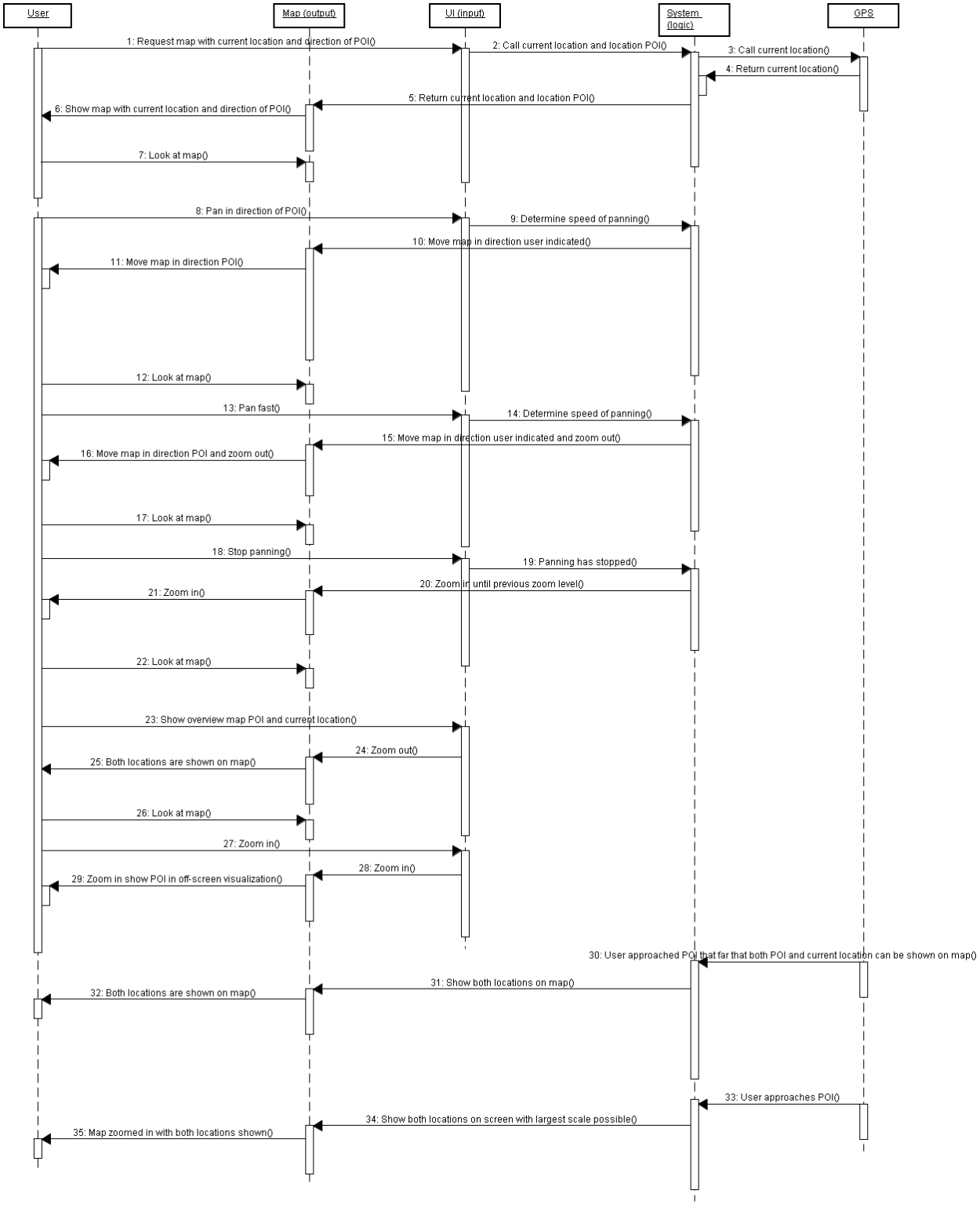


Figure 2.9: Sequence diagram: Walk to POI.

2.4 User interface design

There are several possibilities to go from the different models described in UML to user interface design. Alternatives are:

- user interface diagrams, UI add-ons for UML,
- wireframes. Wireframes are used to get a suggestion for the basic layout and placement for fundamental UI elements. They provide a visual reference upon which to structure each page (figure 2.4),
- prototypes, which can be drawn on paper, made in flash or HTML, or programmed in ready to use code,
- storyboards.

2.4.1 User interface diagrams

Because UML does not include diagrams for user interface design, alternatives have been proposed and are even included in some UML drawing tools. One example are the user interface diagrams presented in the figures below.

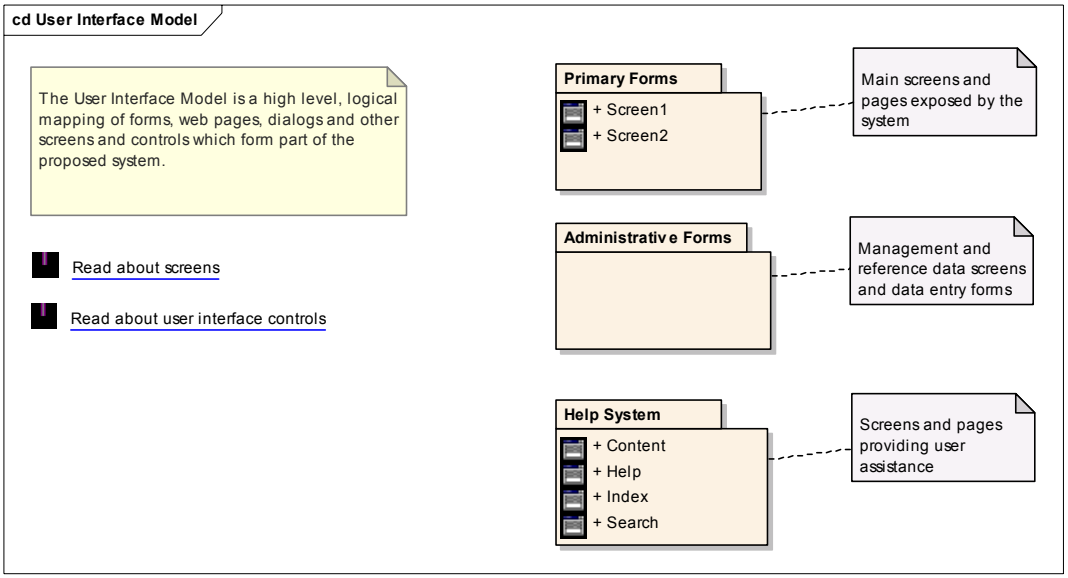


Figure 2.10: UML add-on for UI.

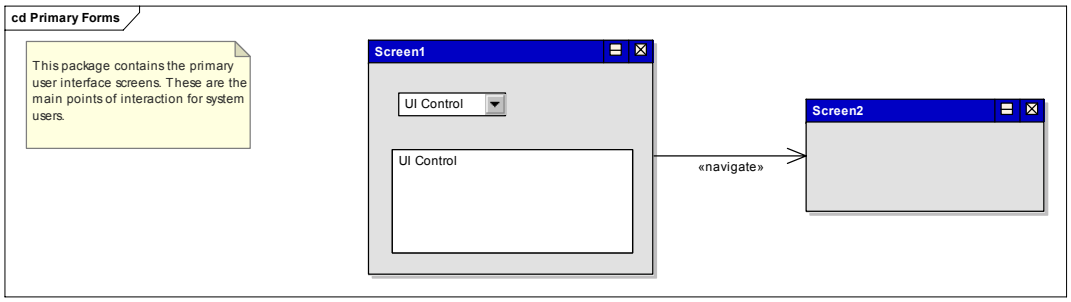


Figure 2.11: Primary forms.

2.4.2 Wireframes

A wireframe is a visualization to present the layout of a user interface or web site. Wireframes are effective because they:

- require minimal skills to create prototypes,
- allow for a quick, iterative testing of designs,
- focus on how site works and reads, not "look and feel" in early stages of site development.

When presented with site design prototypes (or mockups), users and designers tend to focus more on visual elements of a prototype rather than the proposed functionality, structure or content. A wireframe attempts to separate the look and feel from the way it works and reads by presenting a stripped down, simplified version devoid of all distractions.

There is al lot of discussion what a wireframe should or shouldn't include, but generally shown are:
- key elements & location: header, footer, navigation, content objects, branding elements,
- grouping: side bar, navigation bar, content area, etc.,
- labelling: page title, navigation links, headings to content objects
- place holders: dummy text and image place holders.

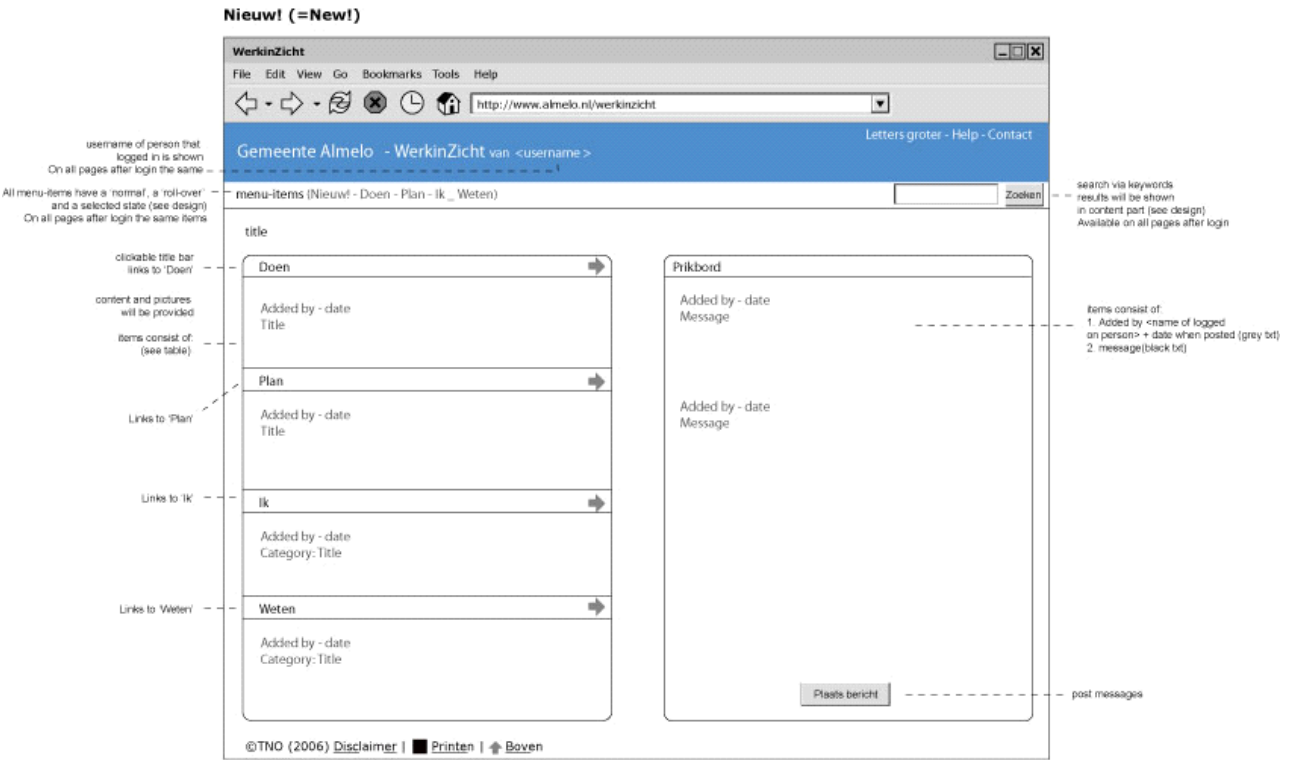


Figure 2.12: Wireframe for a communal website.

2.5 Domain model

The models described above focus on user interaction, but often domain models are generated as well to describe important software components of the system. Generally, these models are designed using class diagrams, one of the most popular diagram types in UML. Figure 2.13 and Figure 2.14 show small examples of domain models for the Amsterdam parking use case, followed by a more detailed description of a number of attributes from the various objects.

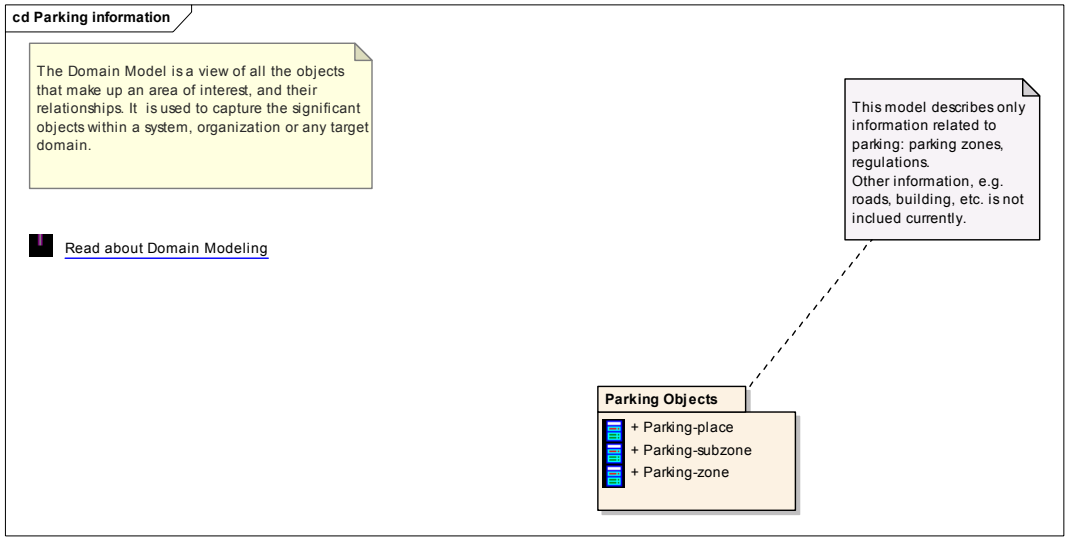


Figure 2.13: Domain model.

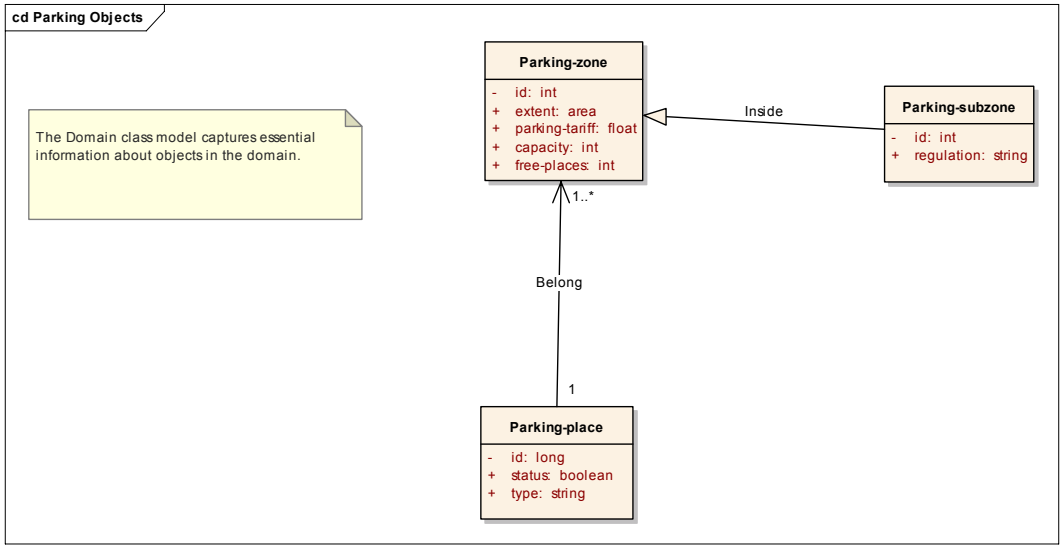


Figure 2.14: Parking Objects.

- **Parking-place**

This is assuming there is up-to-date (real time) information for each parking place. A parking place will have info on its status - free/taken, and the type, e.g. for disabled.

Attributes

Attribute	Notes
id long	An identifer for the parking place.
status boolean	The availibility of the place: yes/no showing free/taken parking place.
type string	Type of the parking place: e.g. regular, for disabled.

- **Parking-subzone**

This is a sub-area inside a zone having separate regulation according times, permissions, etc. The assumption is that the parking rates (tariffs) are the same as for the zone they belong to.

Attributes

Attribute	Notes
id int	Identifier for the subzone.
regulation string	Regulations giving allowed dates and times for parking, or permissions.

- **Parking-zone**

This describes parking zones as they are defined (spatially) in Amsterdam.

Attributes

Attribute	Notes
id int	Identifier for a parking zone.
extent area	This is the spatial extent of the zone, an area feature.
parking-tariff float	The parking tariff for the zone.
capacity int	Number of available parking places in the given parking zone.
free-places int	Number of available free parking places. This is calculated at any moment from the capacity, and the taken parking places in the given zone.

3 Usability of mobile maps

3.1 Introduction

In this chapter we focus on usability of mobile maps. Mobile maps get more common, but because of the usually cluttered small screen, it takes a lot of effort to use a mobile map. The map use task often distracts from another task instead of assisting to it. A mobile map is an example of an application for a fully mobile wirelessly connected device (Gorlenko and Merrick, 2003).

These devices present a number of usability problems that can be divided into technical, environmental, and social challenges. Technical challenges, such as battery life and network connectivity, are nearly solved and they will be completely solved in the near future. Other technical challenges of mobile devices, such as the screen size, we have to take into account when designing an application for a mobile device. The environmental and social challenges are much more challenging. The environmental challenges are very diverse, they include; changing temperature, changing light, changing level of noise, changing level of distraction, changing mobility of the user, competition for attention of other tasks, and the need to manipulate objects other than the mobile device. These environmental challenges are inherent to mobile interaction. Although the cognitive workload can be decreased by right adaptation to for example light and noise level. The last one is the social challenges category. The social challenges include privacy, acceptance and adoption issues, comfort, and personalization. When the technology proceeds and privacy can be better guaranteed people are more comfortable and accepting towards mobile devices. Personalization can also enhance the comfort and acceptance of mobile devices. Personalization can be done by the user, adaptability, or automatic, adaptivity. Adaptive applications can decrease the number of both the social challenges as the environmental challenges. Normally when designing a product the environment in which it will be used is known or at least it is known how it will be used, sitting, standing, or walking. A laptop for instance can be used in different environments, but the user will not be walking. With mobile devices the user can not only use it in different environments, but the user can be sitting, walking, or standing while using the device. The usability is very much influenced by these different activity modes of the user and the ever changing environment.

This chapter first gives an introduction to the usability engineering method we use for the applications we develop. Then we will discuss the state of the art. Next we discuss solutions regarding the technological challenge about the small screen size of mobile devices. With regard to the environmental and social challenges we look at adaptive user support and we end with two case studies.

3.2 Usability Engineering

With the use of usability engineering, which takes the user into account during the design, we would like to develop a usable mobile map that helps defining which design decisions help best decreasing the number of technological, environmental, and social challenges.

Because there are so many challenges for mobile maps ease of use is crucial. In our opinion Usability Engineering (UE) (Gould and Lewis, 1985) (Nielsen, 1994) (Figure 3.1) involves three general principles: 1) early and continuous focus on user and tasks, 2) empirical measurement, 3) and iterative design. With the early and continuous focus on user and tasks we adapt a user oriented design methodology. By taking the user as the center of the design it is also important to take into account the cognitive task load of the user (Neerincx and Lindenberg, 2005). There are several metrics to measure the cognitive workload of a user.

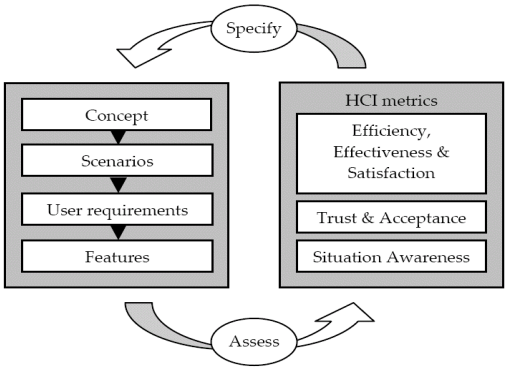


Figure 3.1: The usabality engineering method.

In UE several development cycles, with assessments and re-specifications, are worked through. A concept is used to develop a scenario with the users from which user requirements and features can be derived. The features are implemented and assessed by human-computer interaction (HCI) metrics. These metrics are closely related to the social challenges of (mobile) devices, regarding comfort and acceptance. But the technological challenges are also addressed in the metrics; a better design of the interface can enhance the effectiveness and efficiency of the user. The assessment of the metrics can be done in several different ways, by experts, and by users.

- Usability engineering for mobile devices

UE seems to be a good design approach for a usable mobile map, but has some shortcomings. In UE for example the user's direct manipulation with the device is investigated however this is not sufficient for mobile devices wherein the external context is of much importance also. Mobile interaction can take place anytime, anywhere. This brings along difficulties for the UE methodology (Gorlenko and Merrick, 2003). Understanding the dynamic use context is crucial for user-centered design of mobile applications (Vetere, Howard, Pedell, and Balbo, 2003). With mobile devices it is necessary to test them eventually while the user is mobile. This mobility makes the evaluation extra difficult. The usability can highly differ between a sunny day and a rainy day, between a noisy environment and a silent environment, between reliable connectivity and unreliable connectivity due to for example buildings. The choice between a lab experiment or a field experiment is therefore crucial for mobile devices (Kimber, Georgievski, and Sharda, 2005; Kjeldskov and Graham, 2003; Zhang and Adipat, 2005).

In the task analysis for example we have to identify all possible usages, but because mobile devices can be used anywhere, anytime, there are infinite usages. Furthermore the mobile device usage is a secondary task instead of a primary task and the primary task can have a strong influence on the use of the mobile device. These are two reasons that make prototyping difficult. Not only the device has to have a high degree of fidelity, but the primary task has to be simulated realistic too.

(Zhang and Adipat, 2005) give a summary of challenges in usability testing, these are the same challenges a mobile application has to deal with. Mobile context, connectivity, small screen size, different display resolutions, limited processing capability and power, and different data entry methods were already mentioned as application design challenges, but they are also usability testing challenges. An example is that a lower resolution could have disastrous effects on the usability of a mobile application. Zhang and Adipat (Zhang and Adipat, 2005) say that all the research that has been done on usability testing of mobile applications addresses at least one of five research questions:

1. Can proposed presentation methods help users easily search for/browse/understand specific information of their interest on mobile devices?
2. What are appropriate designs of menu and link structures that help users to navigate easily?
3. Can users easily carry out specific activities (e.g. query searching, filling form, making notes) of an application on mobile devices?
4. What kind of data entry methods can enable users to enter data easily and quickly?

5. How well can mobile applications be used, considering mobile context, mobility, and slow network connection?

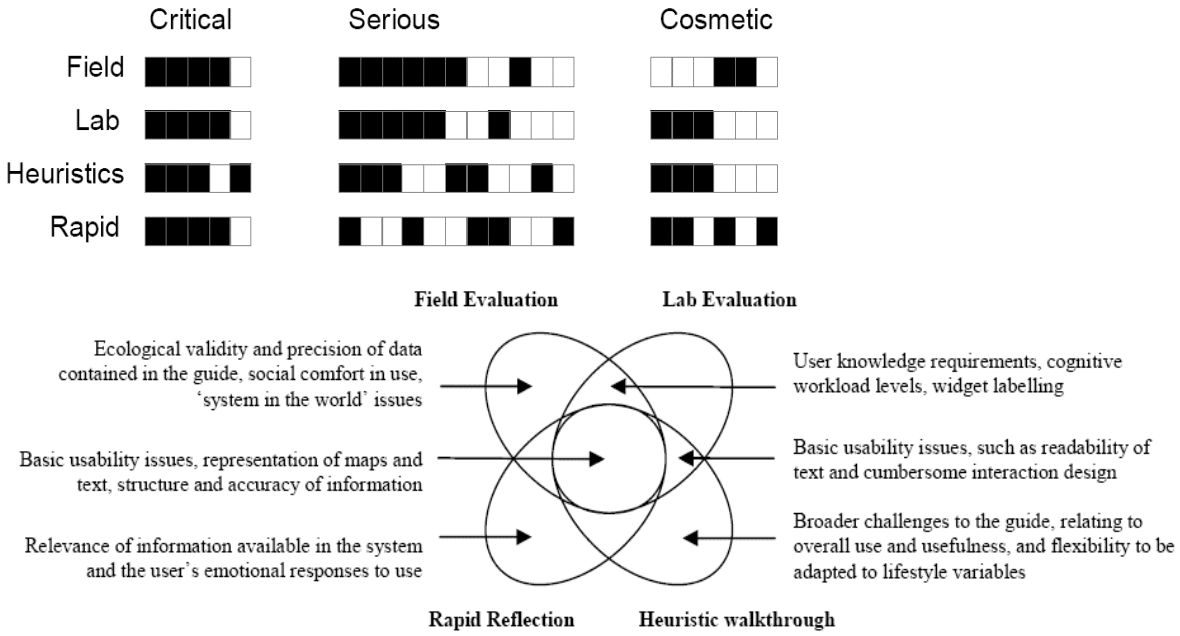


Figure 3.2: Found errors with different experimental methods and the derived framework from Kjeldskov et al (Kjeldskov, Graham, Pedell, Vetere, Howard, Balbo, and Davies, 2005).

These research questions are all interesting, in varying levels, for usable well scaled maps. The questions can be answered using different research methodologies for usability testing. But when to use which method, (Kjeldskov, Graham, Pedell, Vetere, Howard, Balbo, and Davies, 2005; Streefkerk, Esch-Bussemakers, and Neerincx, 2006; Zhang and Adipat, 2005) give different frameworks for usability testing of mobile devices. Important questions that have to be asked before starting with usability testing is if the test will be in the laboratory or in the field and if only experts are used or also prospective users. In (Kjeldskov and Graham, 2003) 102 mobile human-computer conference and journal papers were selected and categorized according to research purpose and research methodology. Applied research, research based on intuition, experience, deduction and induction of the researchers, was most used for engineering purpose. For evaluation lab experiments were most commonly used and field studies were used for different purposes like understanding, engineering, evaluation, and describing practice. A comparative study about different evaluation techniques is done in (Kjeldskov, Graham, Pedell, Vetere, Howard, Balbo, and Davies, 2005). Field evaluation, lab evaluation, rapid reflection, and heuristic walkthrough are compared with each other. The heuristic walkthrough that is being used is a walkthrough especially made for mobile devices (Vetere, Howard, Pedell, and Balbo, 2003). They conclude that every evaluation technique has its benefits in relation to uncovered usability problems, but there also is great overlap between the uncovered usability problems. Rapid reflection sessions appeared to summarize the key issues from both field and laboratory user studies requiring considerably less hours for analysis than video analysis. The reported problems were less specific and less complete, but the top-most severe problems were found in less than half the time required for video analysis (Figure 3.2). Field testing stays of course an indispensable method to evaluate applications for mobile devices.

(Zhang and Adipat, 2005) proposes a framework that mostly depends on the question if the experiment is done to improve the interface design or if the usability of the application in the real context is tested. In the first case one can do a lab experiment while in the other case a field experiment is more appropriate. After selecting the testing method a tool should be chosen. In field experiments this is the actual device, but in lab experiments this can be an emulator as well. While an emulator makes it easier to capture user behavior an actual mobile device allows testers to collect more realistic information. Actual mobile devices should be used whenever this is possible. Of course the experiment has to choose what to measure and how to do data collection.

		Focus group	Wizard of Oz	Game-based	Field
Stage	Analysis	+	-	-	-
	Design	+	+	+	+
	Implementation	-	-	+	+
Purpose	Formative	+	+	+	+
	Summative	-	+	+	+
Complexity	Low	+	-	-	-
	Medium	-	+	+	-
	High	-	-	+	+
Participants	Representatives	-	+	+	-
	End-users	+	+	+	+
Setting	Independent	+	-	-	-
	Natural	-	+	-	+
	Artificial	-	+	+	-
Duration	Short	+	+	+	+
	Longitudinal	-	-	-	+

Table 3.1: Framework of Streefkerk, Esch-Bussemakers, and Neerincx (2006).

Another framework is proposed in (Streefkerk, Esch-Bussemakers, and Neerincx, 2006). This framework uses the classification of (Kjeldskov and Graham, 2003) as a starting point. In contrast to (Zhang and Adipat, 2005) there are more constraints, with these constraints a chose is made for certain evaluation methods. Every experimenter can choose the evaluation method of his/her preference as long as it complies to the given constraints.

We conclude that there are many difficulties for usability testing in the mobile device field, but several frameworks exist which can be used for designing a mobile usable map application.

3.3 State of the art technology

Before creating usable well scaled mobile maps it is important to know what the available techniques are and which efforts have been done on hardware and software level to make mobile maps usable.

3.3.1 Global location

Most mobile maps currently used on mobile devices use the Global Positioning System (GPS) as global navigation satellite system (GNSS) to locate the position of the device. The accuracy of GPS is 4-20 meters. This accuracy can be improved by using fixed ground based reference stations to broadcast the difference between the positions indicated by the satellite systems and the known fixed positions. This is called Differential GPS (DGPS) and can improve the accuracy to 1-3 meters.

A big drawback of GPS is that it is an American military product and it is therefore possible that America decides to give selective availability to civilians. Until May 2000 the selective availability was always on, what meant that there was a standard deviation of 15-100 meters. Another drawback is that it takes time before the required amount of satellites is found and this can be especially difficult in an urban environment. A-GPS resolves the time issue by using an assistant server. Most GPS receivers, such as mobile phones, have limited processing power and are normally not in an ideal position fixing location. By communication with an assistance server the process is quicker and more efficient than regular GPS.

Two European projects to improve accuracy and to be less dependent on the United States of America are launched: the European Geostationary Navigation Overlay Service (EGNOS), and the Galileo positioning system. EGNOS augments the current GPS signal by using a satellite-based augmentation system which

improves the horizontal accuracy to 1-2 meters and the vertical accuracy to 3-4 meters. EGNOS is operational since July 2006 and in June there was a first test to guide visually impaired through a city (Madrid) while using a phone that used EGNOS to give directions (European Commission, 2003). Galileo is a GNSS just like GPS, but European and with a higher precision. Galileo will be commercially available from 2010 and can be as accurate as 1 meter globally and 10 centimeters locally.

3.3.2 Mobile Internet

Currently map applications such as, TomTom and Garmin, which are used in mobile devices, often have the maps preinstalled on memory cards. These do not take an enormous amount of information, but if we want to have dynamic adaptable maps we need a web-service based application. At the moment such applications are restricted because of the limited bandwidth, because speed is one of the main aspects for acceptance of Location Based Services. In the Netherlands GPRS can have a speed up to 115 kb/s, UMTS up to 384 kb/s, and High-Speed Downlink Packet Access (HSDPA) up to 1.8 Mb/s and in the future up to 3.2 Mb/s or even 7.2 Mb/s. HSDPA is a third generation mobile internet platform, but Samsung has announced that in 2010 there will be commercially available mobile internet at the speed of 100 Mb/s for users on the move and 1 Gb/s for users at standstill.

A drawback of using internet is that it needs power from the mobile. At the moment battery life is limited, but increasing. In the first quarter of 2004 a normal phone had about 3 hours talk time and 150 hours of standby time, nowadays there are some phones with 6 hours talk time and up to 400 hours of standby time. Next to battery life price is also of concern as we want usable maps to be affordable. At the moment unlimited access to the internet costs around 10 Euro a month in the Netherlands, which is a reasonable amount.

3.3 Challenges

We think that a good design of an application can solve some challenges for mobile devices. The size of the screen and the limitations of the input methods are two important challenges for mobile devices. With taking these two things into account while designing an interface they could be of less influence for the efficiency and effectiveness of the device. The environmental and social challenges such as changing light and noise levels, and personalization can be partially tackled by an adaptive or adaptable application. In this section we first focus on possible design solutions for technological challenges after which we focus on adaptivity to tackle environment and social challenges. By experimenting with the different design solutions and measuring the HCI metrics we hope to resolve some of the challenges.

3.3.1 Technological challenges

The usability of mobile maps depends heavily on the interaction with the map, the interaction with the mobile device, and the visualization of the information on the map. With the design of the interaction and the visualization of the information, technological challenges such as screen size, interaction limitations of the device (buttons, joystick, physically moving the phone, speech, or touchscreen), and wireless connection speed have to be taken into account. For service/client enabled maps for example the connection speed is very important to make the zooming and panning smooth. A slow connection can be partially resolved by prediction of what the user likes to see in the future and retrieving this information beforehand and by animating the transitions between different images (Klein and Bederson, 2005; Van Wijk and Nuij, 2003) to keep the user spatially and situational aware. In the following section we give an overview of techniques and manners of interaction and visualization for mobile maps that can enhance the usability.

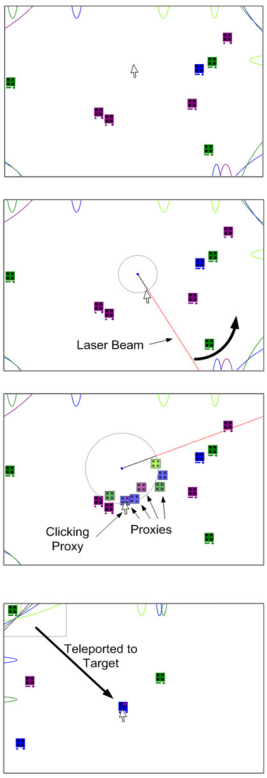
- **Panning**

On a mobile device there is not a lot of space to display maps. Therefore it is useful to have the ability to pan the map.
Several panning methods exist for devices without touch screen:

1. Panning with the buttons of the device.
2. Panning with the joystick (if the device has one).
3. Panning by physically moving the phone.

There also exist methods that can be only be used if there is a touch screen:

4. Panning by moving the stylus over the screen.
5. Panning by centralizing the point on the screen that is touched by the stylus.
6. Gestures
   In (Karlson, Bederson, and SanGiovanni, 2005) gesture based panning and zooming is used. Eight gestures were mapped to corresponding commands. In a study that was performed people indicated that using the gestures was satisfying to fun. The directional gestures were easy to use, but the other gestures were more difficult presumably because the semantic mapping was more abstract.



3.3a

3.3b

3.3c

3.3d

Figure 3.3: Hopping (figure from Irani, P., C. Gutwin & X.D. Yang (2006)).

7. Radial scroll
   Radial scroll (Smith, 2004) is an interface widget to support scrolling. Scrolling is different from panning the does not support diagonal moves. In radial scroll only vertical scroll is supported. The widget is presented by a central point with emerging radial lines. By making circles clockwise the document advances and by making circles counterclockwise the document reverses. When the user is moving near the center the scrolling is fast and further from the center the scrolling is slower.
8. Touch Edge
   In touch edge users push at the edge of the screen at the side they want to pan (Johnson, 1995).
9. Tap-and-Drag
   With tap-and-drag people tap on a touch screen and then drag the stylus/finger along the screen. Continuous pressure is required to drag (MacKay, Dearman, Inkpen, and Watters, 2005). In Johnson (1995) touch edge and tap-and-drag are compared. First participants were asked how they would think a touch-controlled interface should work. Half of them chose for tap-and-drag with moving background. A quarter of the participants chose touch edge with moving (moving in the direction of the off-screen information). And the rest chose something else. Users were faster, made fewer errors, and made fewer moves with the tap-and-drag interface than with the touch edge interface.
10. Touch-n-Go
    Touch-n-Go (Dearman, MacKay, Inkpen, and Watters, 2005) makes it possible to navigate in any direction at variable speed. The speed and direction are determined by the location of the user's touch

relative to the center of the screen, the further from the center the faster the panning. In (Dearman, MacKay, Inkpen, and Watters, 2005; MacKay, Dearman, Inkpen, and Watters, 2005) touch-n-go is compared in a user study with tap-and-drag and a scrollbar (with a scrollbar only vertical and horizontal panning is possible). The scrollbar was significantly slower than the tap-and-drag and touch-n-go technique in all three conditions (sitting, standing, walking). There was no significant difference in speed between tap-and-drag and touch-n-go, but the participants rated the touch-n-go interface highest for ease of use and preferred it also.

11. Hopping is a Halo+Proxy technique (Irani, Gutwin, and Yang, 2006). It consists of three components. Halo is the first component (Figure 3.3a), we will explain Halo later, but it is a technique to show off-screen information. The second component is the laser beam. This is a line which indicates to what point at the border of the screen the user is pointing. The beam is invoked by clicking the mouse on the background and dragging the cursor toward an edge. A circle is drawn with the mouse-down position as center and the travelled distance of the cursor as radius (Figure 3.3b). When the user moves the cursor in a radial fashion, the laser beam will intersect a halo. If a halo is intersected, a proxy is created of the off-screen information that is displayed by this halo. The proxy is placed near the circle (Figure 3.3c). Proxies fade away in five seconds. Users can release the mouse-button and select a proxy to be teleported to the off-screen object. Teleporting is the last component of Hopping. After clicking on a proxy the viewport is moved in a 400ms long animated transition to the location of the object (Figure 3.3d).

Speed dependent automatic zooming (SDAZ) can be used as an adaptation of each of the mentioned panning methods.SDAZ adjusts the zoom level to the panning speed, the faster the user pans the more the map is zoomed out (Jones, Jones, Marsden, Patel, and Cockburn, 2005). In (Jones, Jones, Marsden, Patel, and Cockburn, 2005) SDAZ is compared with traditional panning/scrolling/zooming on PDA for maps and documents. SDAZ was found to be significantly slower for documents and there was no significant speed difference for maps. It was less accurate in target acquisition in documents but more accurate in target acquisition in maps. SDAZ did require fewer actions than the conventional interface. Cockburn and Savage (Cockburn, Karlson, and Bederson, 2006) compare SDAZ also with a conventional interface for maps and documents, but in contrast with the results of Jones et al (2005), their results were all in favour of SDAZ. The SDAZ was faster on both tasks and the workload was rated significantly lower when SDAZ was used for both tasks. The most salient difference between Jones et al. (2005) and Cockburn and Savage (Cockburn, Karlson, and Bederson, 2006) is that the latter did perform their experiment on a 19inch screen while Jones et al (2005) did perform it on a screen of about 3.8 inch. The total display area of the small screen was less than 4% of the size of the other screen. This shows that SDAZ is not optimal for small displays. In (Gutwin and Fedak, 2004) it is also shown that there are big differences in speed between a normal screen and a small screen. This supports the idea that programs have to be adapted to small screens to be usable.

- **Zooming**

The method of zooming is important for the efficiency and effectiveness of the device. In desktop applications most methods of zooming involve clicking something. Most mobile devices do not have a touch screen so methods wherein the keypad is used are developed.

1. In zonezoom the screen is divided into nine zones that can be controlled by the numbers 1-9 on the keypad of the phone (Figure 3.4) (Robbins, Cutrell, Sarin, and Horvitz, 2004). Once zoomed in the map is again divided into 9 segments. To prevent loosing the overview there are several visual cues used. Border shading is used to disambiguate the relationship to the parent and in an overview the location is showed relative to the parent. They implemented it on a smart phone, but have not done user tests with it.

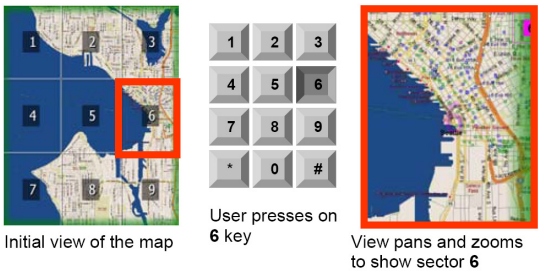


Figure 3.4: Zonezoom (Picture taken from (Robbins, Cutrell, Sarin, and Horvitz, 2004))

2. Focus+context is a method that can be both used for keypads and touch screens. Focus+context displays provide both overview and detail in the same window (Gutwin and Fedak, 2004). Examples of Focus+Context techniques are fisheye views (Sarkar and Brown, 1994), bifocal displays (Robertson and Mackinlay, 1993), and table lenses (Rao and Card, 1994). Fisheye views have proven to perform well for different tasks (Gutwin and Skopik, 2003; Schaffer, Zuo, Greenberg, Bartram, Dill, Dubs, and Roseman, 1996). In (Gutwin and Fedak, 2004) fisheye is compared with a two-level zoom (the image can be zoomed in twice) and a pan/zoom technique on a small screen. The techniques were compared in three different tasks, editing in Powerpoint, navigating through a webpage, and a monitoring task. In the editing and monitoring task the two-level zoom was fastest, in the navigation task the fisheye was fastest and the two-level zoom slowest. Most participants preferred the two-level zoom for all tasks.

All focus+context techniques distort part of the view to be able to show both detail and overview. This distortion can cause problems for targeting (Gutwin, 2002), because distances are distorted and it is easy to overshoot the target.

Other zooming methods require a touch screen:
1. Zooming in and out by clicking on the screen and using a button to indicate zooming in or zooming out.
2. Zooming by selecting a rectangle that is zoomed in.
3. Zooming in and out by dragging the stylus over the screen up- or downwards in vertical direction.

Not only the zooming should be smooth, but also the visualization of the changing level of detail of the map should be smooth. Difficulties with both smooth zooming/panning and automatic adaptation of level of detail are that the best solution is very user dependent. Different users like different speeds of automatic zooming/panning and different users prefer different levels of detail. The preferred speed and level of detail could even be dependent on the mental state of the user. When someone is tired it is possible that he/she might prefer a lower level of detail than normally. Adaptive user support, which we discuss hereafter, could be a solution.

- **Visualization**

Good visualization of the information is very important for mobile maps. Not everything can be shown on the small screen so choices have to be made on the level of detail, enhancement effects that are used, viewpoints, and visualization of on-screen and off-screen object.

- **Level of detail**

To show a map in the right level of detail is very important, because on a small screen not much information can be shown. The challenge is to get as much information on the screen without cluttering it. The preferred level of detail is very dependent on the user preferences and the current state of the user. When the user is tired less information can be processed and the level of detail should be lower to keep the map usable than normally.

Nested level of detail: Nested level of detail means that a higher level of detail is nested in a lower one (Meng, 2005). This can be used for instance when the user is outside a city centre (e.g. the airport) and wants to go to the city centre. There is an overview of the region and a detailed description of the connection from airport to city centre.

Level of detail dependent on location: Bozkurt et al. propose (Bozkurt, Groth, Hansson, Harrie, Ringberg, Stigmar, and Torpel, 2005) that the preferred level of detail is dependent on the amount of information in the map. In a city more detail is required than in a rural area. The outcome of the experiment indicated that the amount of information in reality should be considered while determining the level of detail in mobile maps.

Single window details on demand: Because of the limited screen space of mobile devices not all information, such as the legend, should be shown all the time. Preferably the legend is not necessary at all and all the symbols are self-explanatory (Sarjakoski and Nivala, 2005). But users could need the information of the legend at a certain moment, if the interface uses window detail on demand he/she can access the legend by demanding for it.

- **Enhancement effects**

It is important that target information can be perceived fast, while map reading is most of the time not the primary task of the user. Keates (Keates, 1996) discusses detection, discrimination, identification, recognition and interpretation as fundamental processes involved in map use.
The pop-out concept is about detection of a target, it draws attention to an object. Colour, brightness, movement, direction of illumination, distinct curvature, and tilt are identified to induce a pop-out effect (Julesz, 1984; Treisman, 1986). Size has no strong pop-out effect (Baldassi and Burr, 2004). Lee, Forlizzi, and Hudson (Lee, Forlizzi, and Hudson, 2006) performed a study to order different pop-out effects. They distinguished different categories of targets and distracters: semantic text, semantic numbers, complex symbols, and simple symbols. Regarding colour (colour and black and white) and regarding size (large, medium, and small). In an experiment participants had to find a target among distracters. The experiment showed that semantic text was perceived fastest, than the semantic numbers, followed by the simple and complex symbols. The pop-out effects disappear when multiple semantic or symbolic symbols were presented at the same time. Colour was an effective feature to establish a pop-out effect, size was not.

One has to keep in mind when designing a map that pop-out effects have to be used sparingly otherwise they lose their function. Another thing to keep in mind is that colours have different meanings in different cultures. Red for example means danger in many countries, but in China it means joy (Zipf, 2002).

The Gestalt principles (Wertheimer, 1923): proximity, similarity, closure, simplicity, continuity, connectedness, figure-ground, familiarity/experience, good shape, and common fate can enhance the map reading process also. Other methods are selective filtering (don't show everything), and representation (centering, variable scale).
By showing symbols on the map one should take into account the hierarchy Barkowsky and Freksa (Barkowsky and Freksa, 1997) propose for the importance of depictions. This hierarchy shows that distance and shape can be shown relatively while localization, connectedness, and orientation should be precise (Figure 3.5).

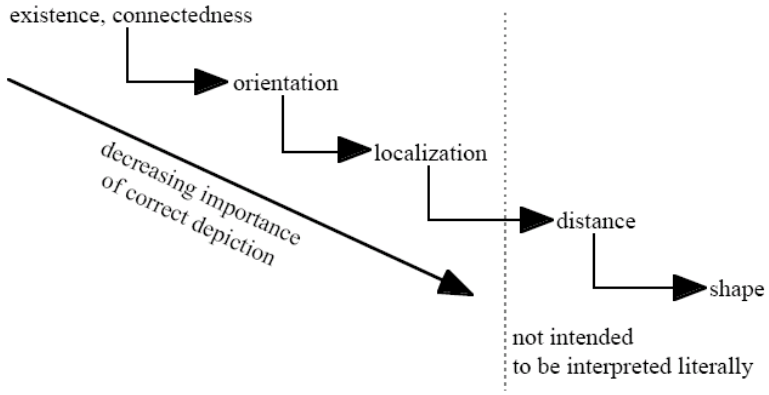


Figure 3.5: The hierarchy proposed by Barkowsky and Freksa (Barkowsky and Freksa, 1997

Pop-out effects, the Gestalt principles, and the hierarchy of Barkowsky and Freksa (Barkowsky and Freksa, 1997) all have one thing in common. They want to reduce the cognitive workload of the user.

- **2D and 3D visualization**

Some usability tests do show that 3D maps can improve task performance in comparison with 2D maps, because spatial relations are better understood. Rakkolainen and Vainio (2001) did an experiment to compare 2D and 3D maps and their results show that search and visualization of location-based information on a life-like 3D map is more intuitive than on a symbolic 2D map. Laakso, Gjesdal, and Sulebak (2003) also did an experiment to compare 2D maps with 3D maps. They also found that 3D had advantages over 2D, but that these advantages would be limited to a minimum for experienced 2D map users. A general downside of 3D representation is that the overview diminishes and that it is less legible on a small screen than a 2D representation. The display can show the map in north-up or forward-up configuration. The preferred configuration highly depends on the task and the individual user (Darken and Cevik, 1999) (Figure 3.6).

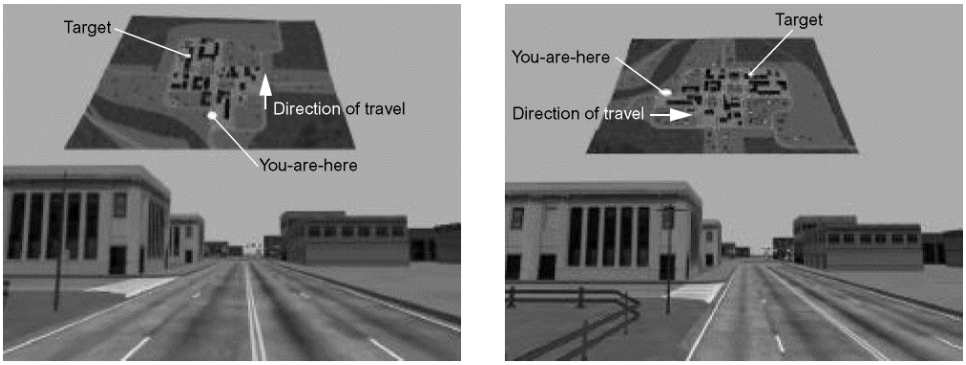


Figure 3.6: On the left forward up map presentation, on the right north up (picture from (Darken and Cevik, 1999)).

- **Visualization of off-screen information**

Because the screen of a mobile device is very small it is inevitable that some information is outside the screen. This information can be shown with off-screen visualization techniques. Below we describe three different visualization techniques which are compared to each other.

CityLights line: This is a line wherein the direction of the off-screen information is visualized by line position (Mackinlay, Good, Zellweger, Stefik, and Baudisch, 2003). The distance to the off-screen information can be encoded by line thickness, line colour, or a label (Figure 3.7).

Arrows: Arrows encode the direction of the off-screen information by the orientation of the arrow and the distance can be encoded in several ways; length, size, colour, shape, or label (Figure 3.7).

Halo: With halos part of a circle/ellipse, an arc is shown on the screen. The orientation is encoded by the position of the arc, while the distance is encoded by how much you see from an arc. The further the off-screen information, the larger the circle/ellipse, the more is visible of the arc on the screen (Baudisch and Rosenholtz, 2003) (Figure 3.7).

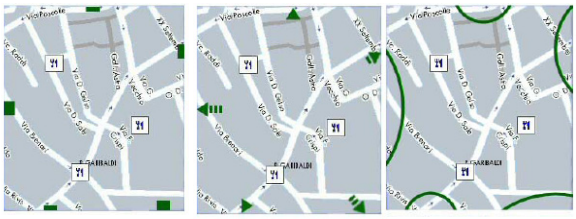


Figure 3.7 :CityLights, Arrows, and Halo (picture from Burigat, S., Chittaro, L., & Gabrielli, S. (2005))

Halo and Arrows, with labels, were compared with each other on four different map tasks by Baudisch and Rosenholtz (Baudisch and Rosenholtz, 2003). Users were 16-33% faster when using the Halo technique without making more errors. Most users did prefer the Halo technique over the Arrows. In another experiment (Burigat, Chittaro, and Gabrielli, 2005) Halo was also compared with Arrows, but then with scaled and stretched arrows. The larger a scaled arrow or the longer a stretched arrow the closer is the information. In this experiment users were slowest when using the Halo technique. They did prefer the scaled arrows on three of the four tasks. Possibly the Halo technique was cluttering the screen too much to be useful. Arrows do clutter also, but less than Halos. A solution for the cluttering of halos is to take several halos together when they represent objects that are near each other, or reduce the amount of off-screen information that is visualized.

Different users could prefer different interaction and visualization styles. An adaptive or adaptable map could be a solution.

- **Route visualization and zooming**

We are focusing on showing maps on mobile devices, but in many cases these maps will be used for showing routes. For showing routes the same visualization and zooming techniques can be used and some more. In (Agrawala and Stolte, 2000) five different visualizations of routes are described

1. Route Highlight Maps: The route is highlighted on a normal route map. There is a constant scaling used which results in either a small map with too little detail, or a detailed map that is too big.

2. TripTiks: TripTiks resemble highlight maps, but the route is stretched over multiple pages. Each page has a constant scaling, but the scaling varies over the pages. TripTiks can show more detail when necessary, but because of the varying scale and the multiple pages it can be difficult to get an understanding of the overall route.
3. Overview/Detail Maps: Overview/Detail maps show multiple maps on one page with different scales to present a single route. A problem with these maps is that users have difficulties to get a cognitive model of the route.
4. 2D Nonlinear Distortion Maps: 2D nonlinear distortion maps provide focus-plus-context viewing. This means that users can choose a region on the map they like to focus on and then apply a nonlinear magnification technique, such as the fisheye technique. An advantage of these maps is that routes can be shown on one page, but the problem is that the edges between magnified and non-magnified parts of the map are heavily distorted.
5. Hand-Drawn Maps: Hand-Drawn maps often use a relative ordering of roads by length. Because of this relative ordering the map can fit on a small page and remain readable. Only some landmarks are drawn along the route, while roads are generally represented with incorrect intersections and shape. Although much is removed from the map the necessary information still remains, therefore hand-drawn maps are often a good combination of readability, clarity, completeness, and convenience.

Agrawala and Stolte (2000; 2001) propose a computer-based route map design which is based on hand-drawn maps, LineDrive. It generalizes the route with respect to shape, road length, and intersections. This generalization is supported by (Barkowsky and Freksa, 1997) hierarchy. The whole route can be shown on a small display, while still being readable. A disadvantage of the current system is that there is a minimal amount of landmarks, e.g. no cities along the route are mentioned. LineDrive maps can be drawn on the internet and the technique is used to implement a navigation system, MOVE (Lee, Forlizzi, and Hudson, 2006). They found that drivers depend on landmarks, paths (important streets), and nodes (intersections) when following directions. Lee, Forlizzi and Hudson (Lee, Forlizzi, and Hudson, 2006) found the following order of salience for the landmarks: neighbourhood, known street, orientation of destination street, number on street of destination, right or left side, and nearby landmark. These landmarks should be salient on the map in the above order for an easy interpretation of the map.

There are not only different visualizations possible for routes than for maps, but also different methods to zoom:

1. Zoom in context
   In (Lee, Forlizzi, and Hudson, 2006) zoom in context means that the road segment where the vehicle is on is enlarged to the maximum available size. The whole route remains visible; this has the advantage that the driver has an overview of the route, but the disadvantage is that the position of the vehicle is moved around the screen inconsistently.
2. Route scrolling
   In route scrolling (Lee, Forlizzi, and Hudson, 2006) the vehicle's location is always in the center of the screen, this resolves the problem of the moving target location from zoom in context. Now the vehicle is constantly in the middle of the screen, but not the whole route is perceived. Because the vehicle is always centered the screen is not used effectively.
3. Speed dependent automatic zooming
   This technique is already discussed above, but can be used also in route maps. Dependent on the speed of the user the route map is zoomed in or out. E.g. if the user is walking the map is zoomed in and when the user is driving on the freeway the map is zoomed out.

3.3.2 Environmental and social challenges

Environmental challenges such as changing light and noise conditions, and social challenges such as personalization, comfort, and acceptance, can be taken into account by an adaptive interface. The information given by the mobile device can be adapted according to the context. Zipf and Jöst (Zipf and Jöst, 2005) represent the context as having three different aspects that overlap each-other: User, situation, and knowledge (Figure 3.8). The user aspect contains the user model, the interaction history and the user situation, the situation contains user and general situation, and knowledge contains both the system knowledge as the user knowledge. Adaptation is important for user centered design, because by adaptation the device can accommodate the specific needs of a user and take into account the specific limitations of the user and of the device. By making the user model adaptable by the user, the adaptivity due to the current

context can be changed, and the user stays in control of the adaptivity of the map. In the following section we first give a short introduction into user modelling and then explain the different categories of user situation, knowledge and interaction history.

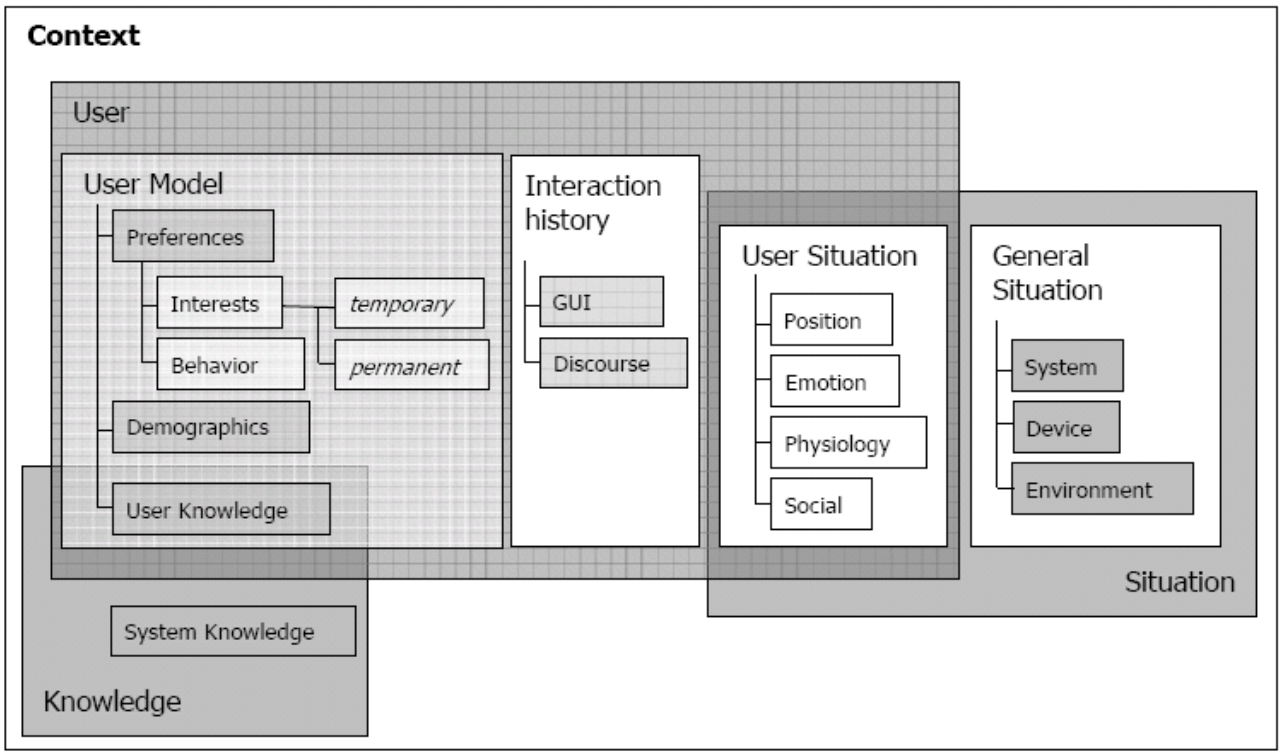


Figure 3.8: Context model of Zipf and Jöst (Zipf and Jöst, 2005).

- **User modelling**

Different users like different maps. But they do not only like different maps, they are also better in using different maps (Dillemuth, 2005) (Darken and Cevik, 1999) and are less susceptible for interruptions when the map matches their needs (Nivala and Sarjakoski, 2004). Therefore to be usable for everyone the map has to be adaptive to the user or adaptable by the user. The question is how to personalize aspects of a service for a specific user. It is for instance very user dependent what a user understands under near. Near is dependent on the user and his/her physical condition, but also on the weather, the task, knowledge about the region, steepness of the area, and structure of the region. The objects that are adapted can be adapted on several levels (Reichenbacher, 2005). High level adaptation could be for instance that the map automatically zooms out when the user pans fast, while low level adaptation could be the colour and the shape of the identifier of the users' location.

To be adaptive the device has to be context aware, but what is context? Schilit, Adams and Want (Schilit, Adams, and Want, 1994) identify three categories of context: 1) computing, 2) user, and 3) physical, these three categories can be matched with the technological, social, and environmental challenges of Gorlenko and Merrick (Gorlenko and Merrick, 2003). Chen and Kotz (Chen and Kotz, 2000) extend these with time and history, and in Sarjakoski and Nivala (Sarjakoski and Nivala, 2005) the user category and the physical category are further specialized for mobile maps. User is divided in purpose of use, social, cultural, and user, physical in physical surroundings, location and orientation. The user is the central point in the context this means that for good context modelling good user modelling is necessary. User modelling can be adaptive and adaptable by the user. We use the categorization of the user from Sarjakoski and Nivala (Sarjakoski and Nivala, 2005).

Purpose of use: What are the tasks at hand? Reichenbacher (Reichenbacher, 2005) identifies five typical actions/tasks for a user of a mobile map: 1) locating, 2) navigating, 3) searching, 4) identifying, and 5) event checking.

Social: This is very difficult to incorporate in a device, because knowledge about relationships of the user with other people is necessary. Incorporation on a lower level is possible, i.e. by vibrating instead of ringing to notify the user when he/she is talking to someone.

Cultural: Things like date and time formats are cultural dependent and when the device is used for the first time this can be indicated by the user.

User: The user model contains information about gender of the user, but also about preferences, skills and knowledge. How can the device get this information?

- **Several methods to create user models**

There are different methods to create user models. Below we explain a few methods and their pros and cons.

Standard user models: A few standard models are implemented in the program and the user belongs to one of the models. Standard user models can be very useful when not all necessary information about the user to make a decision is available (Fink and Kobsa, 2002). Standard user models need some information about the user to assign the user to a certain user model. This information can be obtained explicitly or implicitly.

Explicit acquisition of information: With explicit acquisition users have to fill out questionnaires or fill-in forms. Some advantages are that users have to make their preferences explicit, they know that information about them is gathered, and they can change their preferences quickly (Cremers, Lindenberg, and Neerincx, 2002). Some disadvantages are that people have to put effort in it, don't give accurate answers, and that information may need recurrent updating (Cremers, Lindenberg, and Neerincx, 2002). Implicit acquisition of information could be a solution.

Implicit acquisition of information: No effort is asked from the user when using implicit acquisition methods for information, it also is unobtrusive, and information can be collected for characteristics the users themselves are not aware of (Cremers, Lindenberg, and Neerincx, 2002). But there are also disadvantages for implicit acquisition. Users could dislike the idea that the device is gathering information about them, the system can take wrong decisions due to the uncertainty of some drawn conclusions, and acquiring enough information implicitly for a good user model takes time.

Get information from other users: This resembles the standard user models method, but is different in the way that the user is compared with several other users (Fink and Kobsa, 2002). An advantage over standard user models is that there are more users than standard user models. So the chance to identify a model that is relatively similar to the users' model is bigger. A disadvantage is the privacy, how to keep the information of the different users secure.

Mixed method: All these methods could be mixed also. Some questions, i.e. age, gender, could be asked preferably explicit, while other information is better acquired implicitly.

All the context information is used to show a map that is adapted to the current context. Ya, Wang, Reichenbacher (Yu, Wang, and Reichenbacher, 2003) structure the geo-database along three dimensions: geographic area, content categories, and level of details. Depending on the demographic parameters of users and their current tasks users need different subsets of the same geo-database. This adaptation can take place on four different levels according to Reichenbacher (Reichenbacher, 2003).

- **Situation, knowledge, and interaction history**

Following the representation of context of Zipf and Jöst(Zipf and Jöst, 2005) we give a non-exhaustive list of examples of situation, knowledge and interaction history awareness.

Location awareness: Location awareness is used to provide the user with information of where he/she is and what is in the neighbourhood. Most users would like to know there position on the map as said before GPS can be used here. Depending on preferences of the user the system can also provide Location Based Services such as information about interesting sites in the neighbourhood.

Environmental awareness: This is the ability to know in what interaction setting the user is. Is the user in a noisy crowd or sitting alone in a room? Depending on this the interaction style of the device can be automatically adapted by using for instance vibration or sound. Besides noise there are more environmental variations a device could take into account when adapting. Examples are temperature, lighting conditions, other tasks. But the environmental awareness is broader, the device could for instance know if a museum is closed and adapt its advice to this information.

Mobility awareness: Mobility awareness means that the device knows if the user is sitting, standing, laying, walking, or running. It can derive this from the movements and body posture of the user. If a person is sitting the user maybe prefers filling out some information, but when he/she is running this is no option.

Health awareness: By measuring heart rate, body temperature, and blood pressure the device can know what the health condition of the user is and can give perhaps a warning when the heart rate goes up fast. This could be especially useful for people who have indicated in their user model that they have for instance heart problems.

Activity awareness: The device knows what high level activities the user is involved in and acts accordingly. The user is for instance reading, or writing. The user is most of the time doing something else besides using the map (Reichenbacher, 2005) (Hampe and Paelke, 2005). When the user is for instance reading a route instruction it should not be the case that he/she gets information about an interesting site in the neighbourhood.

Device awareness: A map could be looked at by the user at home behind a desktop computer, but could also be used on a PDA or mobile phone. The screen sizes differ significantly among these devices and therefore the way the information is shown should differ. On a desktop screen a higher level of detail is possible before the screen is getting too cluttered. On a mobile phone on the other hand the screen is best kept as empty as possible, but with all the necessary information. Information about the situation of the device, such as battery level, could be useful too. The backlight could be for instance dimmed when the battery is low. Device awareness is an example of system knowledge.

History: With knowledge about historical actions of the user, the user model can be adapted. If a user for example has chosen on several occasions to have the museums in the surroundings shown the user model can be adapted. In the future the museums in the surroundings will be shown automatically.

Adaptable or adaptive maps could enhance the usability, but therefore they must be accurate. Otherwise the usability decreases. To be accurate the device should take the context as much as possible into account and adapt its context information regularly. The environmental and social challenges are therefore heavily linked to the usability of the application.

3.4 Case -studies

In this section we discuss two case-studies that were performed with maps on mobile devices. We link the usability engineering method we are going to use to the methods they used.

3.4.1 Tourist information and navigation support by using 3D maps displayed on mobile devices (Laakso, Gjesdal, and Sulebak, 2003)

The project in which this study was performed, TellMaris, has the aim to develop 3D maps with tourist information and GPS for mobile devices. In this study there was a strong focus on user requirements and feedback of potential users on the prototypes. The study was divided in three stages. In the first stage the intended user group was asked how they did perform the task, which the application was going to support, now and what functionalities they would like to have in the proposed application. With the answers it was possible to create a prototype which was tested in the second phase. The application was tested in a usability test and with focus groups. Both the participants of the usability test and the participants of the focus groups belonged to the group of intended users. The tasks that were performed in the usability test were typical tasks for the application and performed in a realistic field environment. There was one drawback of the design of the experiment. The 3D map was shown on a mobile device whereas the 2D map was of paper; therefore it was difficult to compare the two views. In the last phase another prototype will be evaluated with the use of usability tests and questionnaires. The 3D map was found fun but less usable than the 2D map, an explanation could be that participants were used to 2D maps. Furthermore results showed that location positioning, through for example GPS, is very important for map information on mobile devices.
The experimental set-up of this study is a sound example of usability engineering. All three general approaches are followed; there is an early and continuous focus on the user, empirical measurements are used, and it is an iterative design. In this experiment the technological challenge for visualization of the viewpoint was examined and several HCI metrics were used to measure the usability of the design. Users were included in both the design stage as during the usability testing.

3.4.2 Understanding users' strategies with mobile maps (Oulasvirta, Nivala, Tikka, Liikkanen, and Nurminen, 2005)

In this study 2D and 3D maps on mobile devices were qualitatively compared. In contrast with the previous study both the maps were shown on the mobile device during usability tests. This study was a preliminary

study to get an idea of what cognitive processes are involved in interacting with mobile devices. With the idea that a design of a map application can be improved when one knows which cognitive processes are involved in interacting with mobile maps. A usability test in a realistic field environment was performed. Participants had to do two kinds of tasks typical for the use of a map, orientation and navigation. The workload of the participants was frequently measured during the experiment. Participants were encouraged to think aloud during the experiment and afterwards they filled out a questionnaire, were interviewed, and debriefed. In the next stage a cognitive model of mapping was developed. They found several strategies of how humans use maps and differences between the strategies used for 2D and 3D maps.
In this case-study usability engineering was properly applied in an experiment about the influence of the viewpoint on the cognitive workload of a map user, this is a social challenge because the user is the bottleneck. Participants were asked how they worked with an application, what they liked, and what they liked to see improved. A cognitive model of interaction with mobile devices was made on which the design of a prototype can be based.
Both cases studies showed that usability engineering is a good approach to develop usable mobile maps.

3.5 Conclusion

In this chapter we focused on the usability of maps on mobile devices. We looked at different aspects of usability, from testing to current state of the art. Different methods of visualization, interaction, and adaptive user support to improve usability were discussed in the view of technical, environmental and social challenges. We conclude that there is a range of methods to improve the usability of mobile maps, but that there is not yet a consensus of which methods improve the usability best. And we looked if there are methods that improve the usability best, or if usability is very user dependent. In future experiments, using the usability engineering method, we would like to compare the different methods and resolve the question of which methods should be used to improve and test the usability of mobile maps.

3.6 References

1. Agrawala, M. and Stolte, C. (2000). A Design and Implementation for Effective Computer-Generated Route Maps. In *AAAI Symposium on Smart Graphics* (61-65).
2. Agrawala, M. and Stolte, C. (2001). Rendering effective route maps: improving usability through generalization. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (241-249).
3. Baldassi, S. and Burr, D. C. (2004). "Pop-out" of targets modulated in luminance or colour: the effect of intrinsic and extrinsic uncertainty. In *Vision Research* (1227-1233).
4. Barkowsky, T. and Freksa, C. (1997). Cognitive requirements on making and interpreting maps. In *Spatial information theory: A theoretical basis for GIS* (347-361).
5. Baudisch, P. and Rosenholtz, R. (2003). Halo: A Technique for Visualizing Off-Screen Locations. In *Proc.CHI 2003* (481-488).
6. Bozkurt, M., Groth, R., Hansson, B., Harrie, L., Ringberg, P., Stigmar, H., and Torpel, K. (2005). Towards Extending Web Map Services for Mobile Applications. In *ICA Workshop on generalisation and multiple representation*.
7. Burigat, S., Chittaro, L., and Gabrielli, S. (2005). Visualizing Locations of Off-Screen Objects on Mobile Devices: A Comparative Evaluation of Three Approaches. In *Mobile HCI2006*.
8. Chen, G. and Kotz, D. (2000). A Survey of Context-Aware Mobile Computing Research. *TR2000-381*. Dartmouth Computer Science.
9. Cockburn, A, Karlson, A., and Bederson, B. B. (2006). A Review of Focus and Context Interfaces.
10. Cremers, A. H. M., Lindenberg, J., and Neerincx, M. A. (2002). Apples or Oranges: a user-centred framework for cooperative user profile management. In *7th WWRF Meeting, Eindhoven, the Netherlands*.
11. Darken, R. P. and Cevik, H. (1999). Map Usage in Virtual Environments: Orientation Issues. In *Proceedings of IEEE VR* (133-140).
12. Dearman, D., MacKay, B., Inkpen, K. M., and Watters, C. (2005). Touch-n-Go: Supporting Screen Navigation on Handheld Computers. Technical Report CS-2005-08), Halifax, NS. Dalhousie University.
13. Dillemuth, J. (2005). Map Design Evaluation for Mobile Display. In *Cartography and Geographic Information Science* (285-301).

14. European Commission (2003). The Galilei Project: Galileo design consolidation.
15. Fink, J. and Kobsa, A. (2002). User Modeling for Personalized City Tours. In *Artificial Intelligence Review* (33-74).
16. Gorlenko, L. and Merrick, R. (2003). No wires attached: Usability challenges in the connected mobile world. In *IBM Systems Journal* (639-651).
17. Gould, J. D. and Lewis, C. (1985). Designing for usability: key principles and what designers think. In *Communications of the ACM* (300-311).
18. Gutwin, C. (2002). Improving focus targeting in interactive fisheye views. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves* (267-274).
19. Gutwin, C. and Fedak, C. (2004). Interacting with big interfaces on small screens: a comparison of fisheye, zoom, and panning techniques. In *Proceedings of the 2004 conference on Graphics interface* (145-152).
20. Gutwin, C. and Skopik, A. (2003). Fisheyes are good for large steering tasks. In *Proceedings of the conference on Human factors in computing systems* (201-208).
21. Hampe, M. and Paelke, V. (2005). Adaptive maps for mobile applications. In *Mobile maps 2005 interactivity and usability of map-based mobile services*.
22. Irani, P., Gutwin, C., and Yang, X. D. (2006). Improving selection of off-screen targets with hopping. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (299-308).
23. Johnson, J. A. (1995). A comparison of user interfaces for panning on a touch-controlled display. In *Proc.CHI 1995* (218-225).
24. Jones, S., Jones, M., Marsden, G., Patel, D., and Cockburn, A. (2005). An evaluation of integrated zooming and scrolling on small screens. In *International Journal of Human-Computer Studies* (271-303).
25. Julesz, B (1984). A brief outline of the texton theory of human vision. In *Trends in Neuroscience* (41-45).
26. Karlson, A. K., Bederson, B. B., and SanGiovanni, J. (2005). AppLens and LaunchTile: Two Designs for One-Handed Thumb Use on Small Devices. In *Proc.CHI 2005*.
27. Keates, J. S. (1996). Map understanding. Second Editon. In Edingburg: Longman.
28. Kimber, J., Georgievski, M., and Sharda, N. (2005). Developing Usability Testing Systems and Procedures for Mobile Tourism Services. In *Annual Conference on Information Technology in the Hospitality Industry*.
29. Kjeldskov, J. and Graham, C. (2003). A review of mobile HCI research methods. In *Proceedings of Mobile HCI 2003* (317-335).
30. Kjeldskov, J., Graham, C., Pedell, S., Vetere, F., Howard, S., Balbo, S., and Davies, J. (2005). Evaluating the usability of a mobile guide: the influence of location, participants and resources. In *Behaviour & Information Technology* (51-65).
31. Klein, C. and Bederson, B. B. (2005). Benefits of Animated Scrolling. In *Proc.CHI 2005*.
32. Laakso, K., Gjesdal, O., and Sulebak, J. R. (2003). Tourist information and navigation support by using 3D maps displayed on mobile devices. In *Proceedings of HCI in Mobile Guides (Udine, Italy: in conjunction with Mobile HCI 2003)*.
33. Lee, J., Forlizzi, J., and Hudson, S. E. (2006). Iterative Design of MOVE: A Situationally Appropriate Vehicle Navigation System. In *International Journal of Human-Computer Studies (in press)*.
34. MacKay, B., Dearman, D., Inkpen, K., and Watters, C. (2005). Walk'n scroll: a comparison of software-based navigation techniques for different levels of mobility. In *Proceedings of the 7th international conference on Human computer interaction with mobile devices & services* (183-190).
35. Mackinlay, J., Good, L., Zellweger, P., Stefik, M., and Baudisch, P. (2003). City Lights: Contextual Views in Minimal Space. In *Proc.CHI 2003* (838-839).
36. Meng, L. (2005). Ego centres of mobile users and egocentric map design. In Zipf, A. E. D. T. and Reichenbacher, T. E. D. T., (89-105). Springer.
37. Neerincx, M. A. and Lindenberg, J. (2005). Situated cognitive engineering for complex task environments. In.
38. Nielsen, J. (1994). Usability Engineering. In Morgan Kaufmann.
39. Nivala, A. M. and Sarjakoski, L. T. (2004). Preventing Interruptions in Mobile Map Reading Process by Personalisation. In *Proceedings of The 3rd Workshop on'HCI in Mobile Guides* (13-16).

40. Oulasvirta, A., Nivala, A. M., Tikka, V., Liikkanen, L., and Nurminen, A. (2005). Understanding usersÆ strategies with mobile maps. In *Mobile Maps 2005-Interactivity and Usability of Map-based Mobile Services, a workshop*.
41. Rakkolainen, I. and Vainio, T. (2001). A 3D City Info for mobile users. In *Computers & Graphics* (619-625).
42. Rao, R. and Card, S. K. (1994). The table lens: merging graphical and symbolic representations in an interactive focus+ context visualization for tabular information. In *Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence* (318-322).
43. Reichenbacher, T. (2003). Adaptive Methods for Mobile Cartography. In *The 21th International Cartographic Conference* (1311-1321).
44. Reichenbacher, T. (2005). Adaptive Egocentric Maps for Mobile Users. In Meng, L., Zipf, A., and Reichenbacher, T., (141-158). Springer.
45. Robbins, D. C., Cutrell, E., Sarin, R., and Horvitz, E. (2004). ZoneZoom: map navigation for smartphones with recursive view segmentation. In *Proceedings of the working conference on Advanced visual interfaces* (231-234).
46. Robertson, G. G. and Mackinlay, J. D. (1993). The document lens. In *Proceedings of the 6th annual ACM symposium on User interface software and technology* (101-108).
47. Sarjakoski, L. T. and Nivala, A. M. (2005). Adaptation to Context - A Way to Improve the Usability of Mobile Maps. In Meng, L., Zipf, A., and Reichenbacher, T., (107-123). Springer.
48. Sarkar, M. and Brown, M. H. (1994). Graphical fisheye views. In *Communications of the ACM* (73-83).
49. Schaffer, D., Zuo, Z., Greenberg, S., Bartram, L., Dill, J., Dubs, S., and Roseman, M. (1996). Navigating hierarchically clustered networks through fisheye and full-zoom methods. In *ACM Transactions on Computer-Human Interaction (TOCHI)* (162-188).
50. Schilit, B. N., Adams, N., and Want, R. (1994). Context-aware computing applications. In *Proceedings IEEE Workshop Mobile Computing Systems and Applications 1994* (85-90).
51. Smith, G. M. (2004). The radial scroll tool: scrolling support for stylus-or touch-based document navigation. In *Proceedings of the 17th annual ACM symposium on User interface software and technology* (53-56).
52. Streefkerk, J. W., Esch-Bussemakers, M. P., and Neerincx, M. A. (2006). Designing personal attentive user interfaces in the mobile public safety domain. In *Computers in Human Behavior* (749-770).
53. Treisman, A. (1986). Features and objects in visual processing. In *Scientific American* (114-125).
54. Van Wijk, J. J. and Nuij, W. A. A. (2003). Smooth and efficient zooming and panning. In *Proc.INFOVIS 2003* (15-22).
55. Vetere, F., Howard, S., Pedell, S., and Balbo, S. (2003). Walking through mobile use: novel heuristics and their application. In *Proceedings of OzCHI* (24-32).
56. Wertheimer (1923). Untersuchungen zur Lehre von der Gestalt. II. In *Psychologische Forschung* (301-350).
57. Yu, Z., Wang, Y., and Reichenbacher, T. (2003). User Modelling for Geo-Database Adaptation. In *The 21th International Cartographic Conference* (426-434).
58. Zhang, D. and Adipat, B. (2005). Challenges, Methodologies, and Issues in the Usability Testing of Mobile Applications. In *International Journal of Human-Computer Interaction* (293-308).
59. Zipf, A. (2002). User-Adaptive Maps for Location-Based Services (LBS) for Tourism. In Woeber, K, Frew, A, and Hitz, M., *Proceedings of ENTER* Springer Computer Science.
60. Zipf, A. and Jöst, M. (2005). Implementing Adaptive Mobile GI Services based on Ontologies-Examples for pedestrian navigation support. In *CEUS-Computers, Environment and Urban Systems-An International Journal.Special Issue on LBS and UbiGIS*.

4 Server side technology

The tGAP (topological Generalized Area Partitioning) structure is a collection of data structures that enable generalisation of spatial data. These structures store results of a generalisation process, and allow selection of features to be shown for any required level of detail (LoD), performing in that way an on-fly map generalisation by feature selection in tGAP. The generalization process reduces the level of detail by merging unimportant features to more important features (see Figure 4.1). Data forming a partition of space is considered. Results of the generalisation process are stored in the tGAP structure. It stores the geometry only for features belonging to the highest level of detail. As we work with a topological model for storing spatial data, geometry is indeed stored only for edges forming boundaries of area features. References for the relation between a face, i.e. an area feature, and its boundary edges (typical references of a topological model) are stored in the tGAP structure. Specific references stored from the tGAP structure are those between highest LoD features, and features created during the generalisation. For area features created from merging, tGAP structure stores references to the merged features. Each feature is associated with an importance range, which is used for selecting the right features for a required LoD. A relation is established between importance values and (LoD, which is compatible and translated to) scale of a map[1].

Merging of less important faces to more important faces is based on importance values associated to each face. Figure 4.1 illustrates the generalization process for the map partition shown in 'Step 0'. The other maps in Figure 4.1 show the result of the generalization in steps, and are labelled according to that. The result of each step is a (map) partition. A map[2] is a collection of faces, and each face is constructed by the set of edges that form its boundary. The collection of faces that should be visible at a certain scale determines the collection of edges that should be visible, namely edges that are in the boundary of at least one of these faces. There is a last issue in the generalization process: boundary edges get simplified as the level of detail decreases. This can be also seen in Figure 4.1.

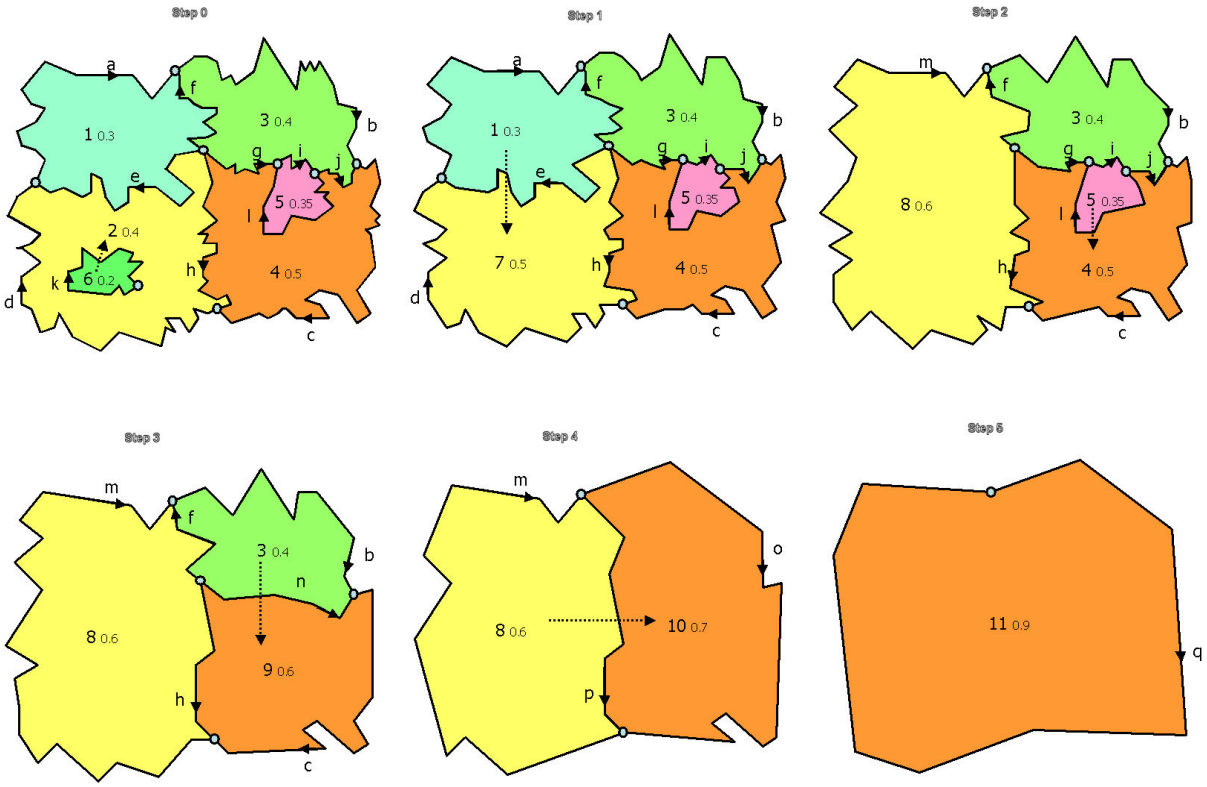


Figure 4.1: Merging of faces based on importance values. Different colours show different classes. Faces are numbered, and edges are labelled with letters. The subscript to a face number is its importance value.

[1] Considering this relation, we use the terms LoD and map scale interchangeably.
[2] For simplicity we will use the term 'map' throughout the text, meaning a map partition.

To capture the generalization process we need to keep track of the merging of faces in each step, how this is reflected to the boundary edges, and the simplification of edges. The data structures forming the tGAP take care of these three issues. The tGAP structure consists of a face tree holding the hierarchy of faces formed by merging, an edge forest that holds the corresponding relations between boundary edges, and BLG (Binary Line Generalization) trees, one tree for each edge that holds information about edge simplification.

The coming sections explain the tGAP structure and its implementation. Generalization process shown in Figure 4.1 is used for illustration. Section 4.1 describes how the tGAP is filled from the merging; Section 4.2 explains how tGAP is used to select the right features for a given importance level (to be translated to map scale). The implementation of tGAP structure in Oracle Spatial is explained in Section 4.3. Section 4.4 contains ideas about progressive transfer and visualisation on the client side.

4.1 Creating the tGAP structure

The tGAP structure consists of a face tree, an edge forest, and BLG trees. Building of each structure is treated separately in the coming sections.

- **Face tree**

Generalization is performed in steps. Each step merges two existing faces to a new face. The merged faces are replaced by the new face, which continues further in the merging process. The new face and the merged faces have a parent-child relation. The process ends when only one face is left. The hierarchy of faces created by this process is a binary tree. Figure 4.2 shows this hierarchy – the face tree created by the generalisation process of Figure 4.1. Leaf nodes in the tree are the original faces, i.e. faces at the highest LoD. The root of the tree is the complete area of the map, union of all the original faces. Faces created during the generalization process form the other nodes of the tree.

Figure 4.1 illustrates the 6 steps for the generalization of the map shown in Step 0. A dashed arrow shows the least important face and its most compatible neighbour (where the arrow is headed). In the original map, Step 0, face 6 has the lowest importance value, 0.2, and face 2 is its only neighbour, therefore the most compatible face. In step 1 the two faces are merged into a new face, labelled 7. The two faces labelled 6 and 2 cease existing at the importance level 0.2, whereas face 7 starts existing at this importance level. The importance value of face 7 is calculated, 0.5, and the face is considered for the next step in the merging process. The process continues until all is merged to one face, labelled 11. Figure 4.2 shows the hierarchy of faces created from this generalization process. A node in the tree is a face. Each node is associated with the importance range for which the node exists – the values below the node. In the right side of the tree are shown the steps performed to create the tree, each step is associated with its importance value. Nodes in the tree are levelled with the step in which they stop existing.

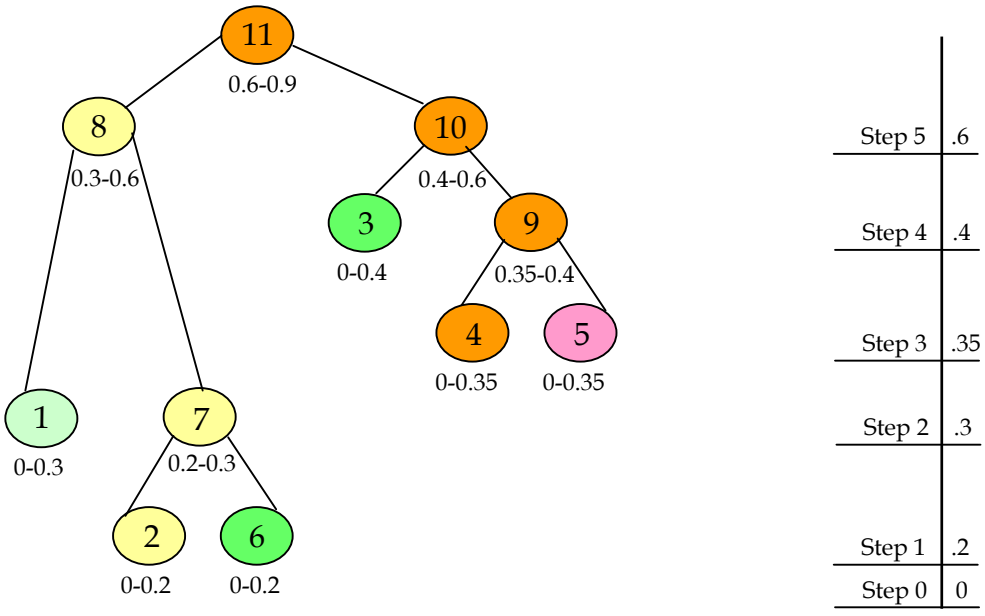


Figure 4.2: The face tree. Nodes in the tree are faces, and lines depict the merging of two faces into the parent face. Ranges associated to each node show the importance level at which the face is visible.

Merging is (currently) performed one by one: the least important face is merged to the most compatible neighbour. Merging is thus based on an importance value for each face, and compatibility between pairs of faces. The importance value of a face is calculated via a function e.g. as the product of the face area with the weight of the face class. The compatibility is calculated by another function, e.g. as the product of the length of the common boundary between two neighbour faces with a similarity value between their classes.
The merging process starts with the original faces. The least important face is selected first, then its most compatible neighbour. The two faces are merged to a new face, which gets the class of the more important face, i.e. the compatible neighbour. The importance value of the new face is calculated. The two merged faces are discarded from the next step of the merging process, and the new face is added. The next step continues in the same way: first selecting the current least important face, then its most compatible neighbour, merging these two faces to a new one that takes the class of the compatible neighbour (for an overview of the algorithm, see Appendix 3). Each step performs a change in the map, while between two consecutive steps the map is unchanged. Therefore, the number of steps is equal to the number of changes a map undergoes. (For the current generalisation this is one less than the number of faces at the highest LoD.)

- **Edge forest**

Area features can be stored in two different ways: explicitly storing their geometry, or storing the geometry of boundary edges together with references to faces of which they form the boundary. The second way of storing is not redundant, and it is known as topological storage (or topological model). Different topological models exist, e.g. the left-right topology, or winged edge topology. We store faces using the left-right topology without edge references. This model stores the edge geometry (as a directed arc with start and end node), together with references to the left and right face of the edge. Each face is then constructed from the list of edges that refer to it as a left or right face. That determines the types of face changes that effect edges. Edges that constitute the boundary of the two merged faces at a certain step undergo three kinds of changes. An edge disappears if it is part of the common boundary of the two merged faces. The other edges may continue existing, but the left or right face of each edge is changed. Two edges may join to form only one edge. An edge takes the importance value from the importance of the step in which it changed. The three cases of edge changes are illustrated with examples in the coming paragraphs.
Edge 'k' is common boundary between the merged faces 2 and 6 (see Figure 4.1), and it disappears at step 1, importance level equal to 0.2. An example of an edge that changes its left or right face is edge 'h' in step 1; its left face changes from 2 in step 0, to face 7 in step 1. Figure 4.3 shows the changes occurred to the edges in step 1. Edge 'k' disappears, edge 'd' changes its right face from 2 to 7, edges 'e' and 'h' change their left face from 2 to 7.

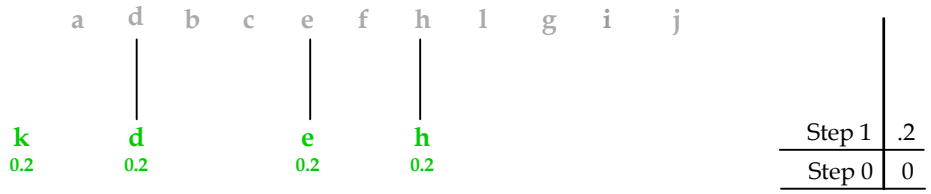


Figure 4.3: Changed edges in Step 1. The green colour shows edges to which changes have occurred.

When a face disappears, like face 5 in Step 3 (see Figure 4.1), its boundary edge 'i' joins the connected edges 'g' and 'j'. Figure 4.4 shows changes that occurred to edges in Step 3, importance equal to 0.35. The three edges 'g', 'i', 'j' joined to edge 'n', and disappear at that step. Edge 'l' disappears, as it is the common boundary between the merged faces, 5 and 4. Edge 'c' changes its left face from 4 to the new created face 9.

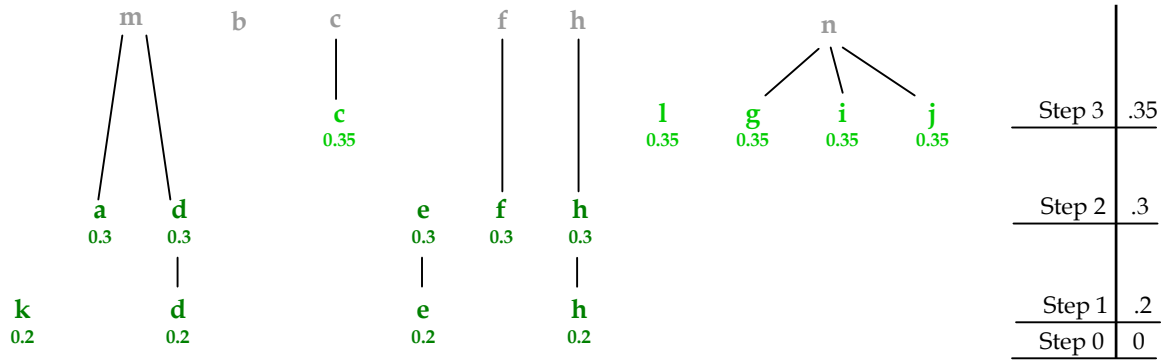


Figure 4.4: Changes edges in Step 3. The light green shows changes happening in this step, the dark green are changes occurred in previous steps.

Edges that disappear at a step remain isolated from the growing hierarchy of edges. This hierarchy does not have a single root; therefore it is not a tree. We call it a forest. The complete edge forest for the merging process of Figure 4.1 is shown in Figure 4.5. Ranges attached to an edge form the importance range at which the edge (associated with the left-right face information) exists. In the right sides there are the steps at which changes occur, each step associated with its importance level. Nodes in the forest are aligned with the step at which the edge information is changed.

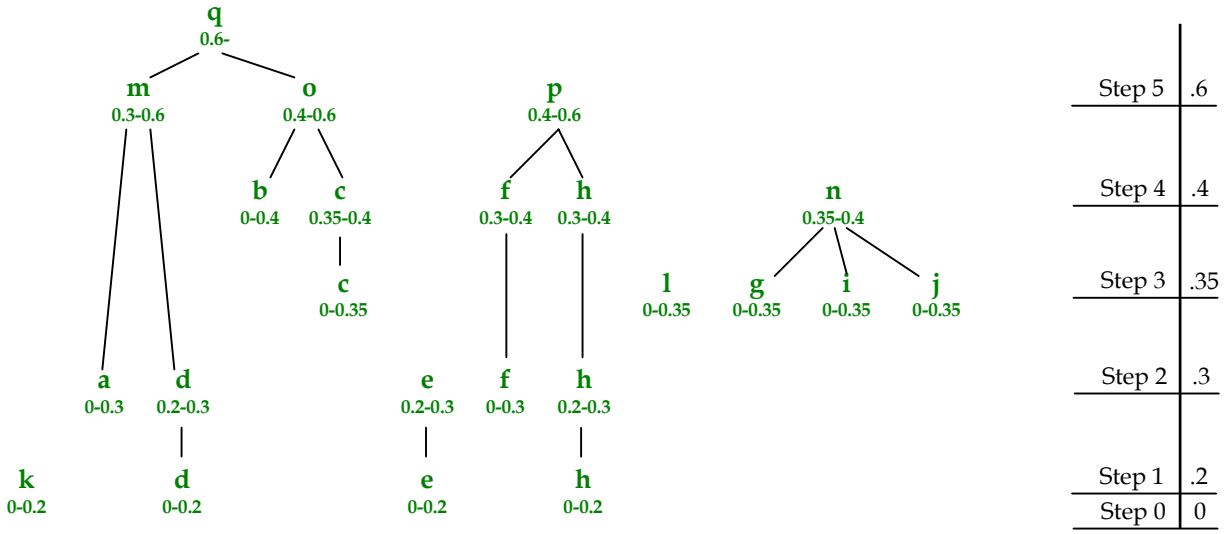


Figure 4.5: The edge forest. Ranges associated to a node show the importance values at which the edge information is unchanged.

- **The BLG trees**

There is a BLG (Binary Line Generalisation) tree for each edge. The BLG tree stores the results of the Douglas-Peucker algorithm for line simplification. Douglas-Peuker algorithm is one of the oldest and most popular algorithms used for line simplification. It uses the closeness of a vertex to a line segment to decide if a vertex will be included or not in the simplified version of the line for a given tolerance.

The algorithm starts with the roughest approximation of an edge being the straight line connecting the two end nodes. For each inner vertex, the distance to this straight line is calculated. The furthest vertex from the straight line is included in the list of vertices forming the next approximation of the edge. This new approximation consists then of two line segments (see Figure 4.6). For each new line segment, distances of all inner vertices to the line segment are calculated. Again, the furthest vertices to each line segment are included for the next edge approximation. This process continues until all vertices have a distance assigned. This distance is considered as a tolerance value for the vertex, and it is used to decide if the vertex will be shown for a certain LoD.

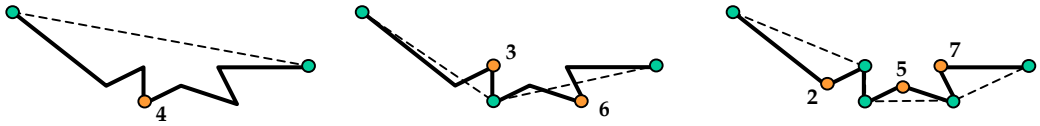


Figure 4.6: Three steps of Douglas-Peuker algorithm for edge simplification. The edge is drawn in thick black line; edge approximation on each step is drawn in thin dashed line. Vertices in green are part of the approximation; in orange are vertices selected for inclusion in the next approximation.

Figure 4.6 illustrates the application of Douglas-Peuker algorithm to an edge. The first step (left image) is the roughest approximation of the edge as the line segment connecting the start and end nodes of the edge, shown in green circles. The furthest vertex from this line segment is the 4th vertex (shown in orange). It is selected to be part of the approximation for the next step. Edge approximation in the second step is made of the line segment $\langle v_1, v_4 \rangle$ between 1st and 4th vertex, and the line segment $\langle v_4, v_8 \rangle$ between 4th and 8th vertex. For each line segment, the furthest vertex is calculated: vertex 3 is the furthest from $\langle v_1, v_4 \rangle$ line segment, and vertex 6 is the furthest from $\langle v_4, v_8 \rangle$ line segment. The 3rd and 6th vertex are added to the edge approximation for the next step, which is made of four line-segments $\langle v_1, v_3 \rangle$, $\langle v_3, v_4 \rangle$, $\langle v_4, v_6 \rangle$, and $\langle v_6, v_8 \rangle$. The third step calculates the distance of 2nd vertex from line $\langle v_1, v_3 \rangle$, distance of 5th vertex from $\langle v_4, v_6 \rangle$, and distance of 7th vertex from $\langle v_6, v_8 \rangle$. Addition of these vertices to the third approximation produces the original edge. The algorithm finishes after the third step.

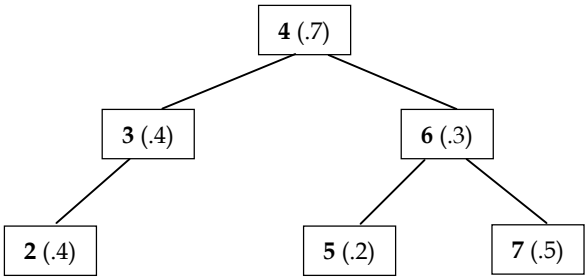


Figure 4.7: The BLG tree resulting from Douglas-Peuker edge simplification of Figure 4.6. Vertex number is shown in bold face, and tolerance for each vertex is given in brackets.

Results of the Douglas-Peuker algorithm are stored in a tree. Nodes of the tree are inner vertices of the edge, associated with the tolerance (i.e. the calculated distance). The root of the tree if the furthest vertex from the straight line connecting end nodes of the edge. Each step of the Douglas-Peuker algorithm adds to every leaf node v_i of the current tree (created from previous step) at most two nodes, added in the tree as children of v_i. The new nodes are the furthest vertices to the two line segments parting from v_i. They are one at the left and the other at the right of v_i, and are put accordingly in the tree to the left and right of node v_i. The tree formed in that way is a binary tree. Figure 4.7 gives the BLG tree for the edge simplification shown in Figure 4.6, and Figure 4.8 shows Douglas-Peuker algorithm for edges together with their BLG trees storing the results of the algorithm.

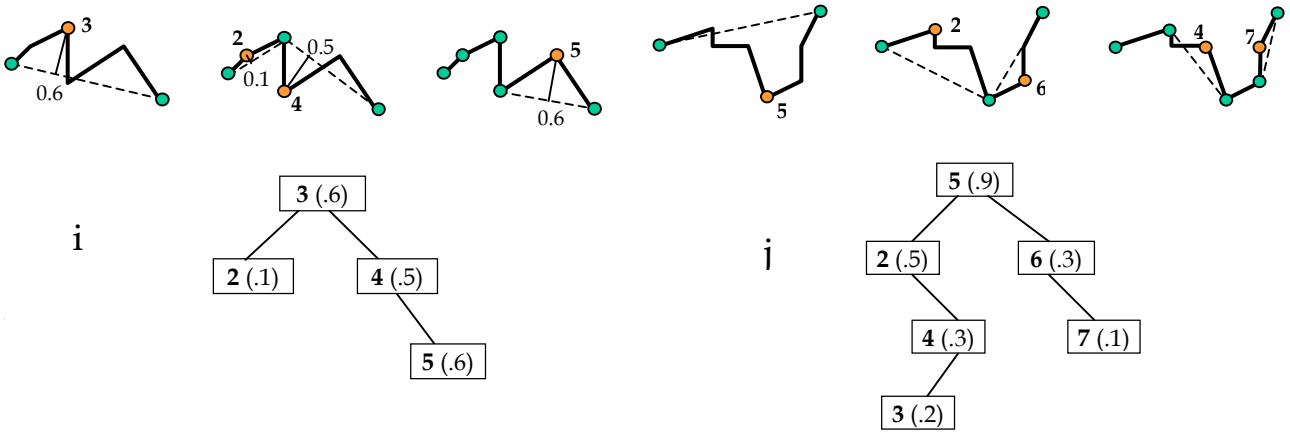


Figure 4.8: Douglas-Peuker simplification for edges 'i' and 'j', and their BLG trees.

There are cases when existing edges are to be joined to form only one edge boundary of a face at a lower LoD. For example, edges 'g', 'i', and 'j' are joined to edge 'n' in step 3 of the generalisation process (see Figure 4.1 and Figure 4.4). The geometry of edge 'n' is the union of geometries of the 'g', 'i', and 'j' edges. We use the BLG trees of the composing edges to form the geometry of the new edge 'n'. For each part of 'n' – 'g', 'i', and 'j' – we use the simplification performed already, i.e. we use the BLG trees of 'g', 'i', and 'j'. The common node between 'g' and 'i', and the common node between 'i' and 'j', are inner vertices for the new edge 'n', but they have no tolerance value assigned. Joining of BLG trees is done in pairs, and a tolerance value is calculated for the common node. Figure 4.9 (left) shows the joined BLG tree for edges 'i' and 'j'. A tolerance value 1.4 is associated to the common node, named 'ij'.

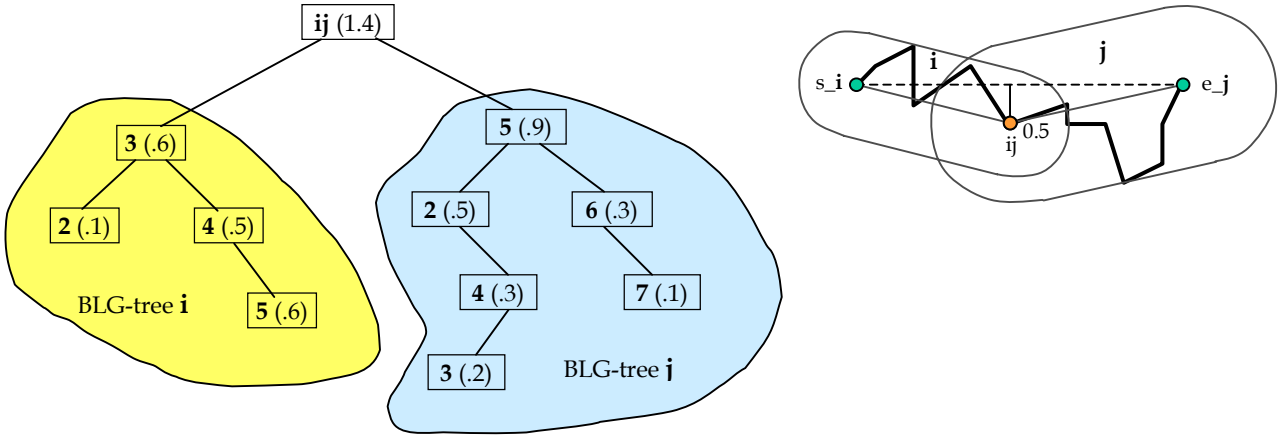


Figure 4.9: Joining of two BLG trees (left), and the tolerance calculation for the common node (right).

Figure 4.9 (right) illustrates the calculation of the tolerance for the common node. The tolerance value for the common node 'ij' is not calculated in the same way as for inner vertices of an edge (by Douglas-Peuker algorithm). It is estimated from the top tolerance of edges 'i' and 'j', and the distance of node 'ij' to the straight line connecting the end nodes of the joined line. The formula for the calculation is: $tol_{ij} = \max\{tol_{root(i)}, tol_{root(j)}\} + dist(ij, \langle s\_\mathbf{i}, e\_\mathbf{j}\rangle) = \max\{0.6, 0.9\} + 0.5 = 1.4$
When there are more than two edges to be joined, as it is the case of edge 'n' composed of three edges 'g', 'i', and 'j', the full joining is done in steps. For the example of edge 'n', BLG trees of 'i' and 'j' are joined first, then the joined BLG of 'i' and 'j' is joined with the BLG tree of 'g'. The tolerance of the common node is again estimated as previously. The tolerance value for the common node is bigger than the tolerance of all inner vertices, which means a common node is the first vertex selected for joined edge approximation, and during visualization will appear before any other vertex.

4.2 Using tGAP structure

Once the tGAP structure is built, it can be used to select features that should be shown for a certain scale. Once a map scale is given, it is translated to importance value, which is used to select features. A face will be shown if the given importance value is in the importance range of the face. Figure 4.10 gives faces to be shown for an importance value equal to 0.38.

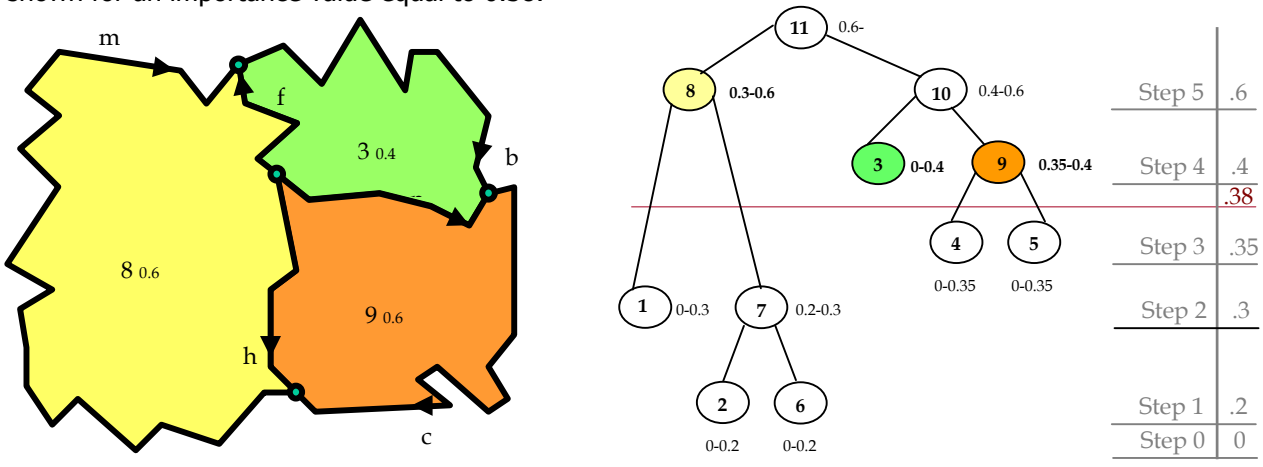


Figure 4.10: Faces to be shown for the importance level 0.38.

The importance value 0.38 is in the importance range [0.35, 0.4) formed between steps 3 and 4. The map created from step 3 is unchanged for values in this range. Faces to be shown are the leaf nodes of the (sub)tree created by cutting all nodes with importance values lower than 0.38. These are faces 3, 8, and 9; their importance ranges include the value 0.38. They form a partition of space, being leaf nodes of the binary (sub)tree.

Edges to be shown at the importance value 0.38 are leaf nodes in the forest remained after cutting nodes with importance less than 0.38. Those are the edges that include the importance value 0.38 in their importance range. They are the boundaries of faces to be shown for that importance, namely, faces 3, 8, and 9. Figure 4.11 shows the edges to be displayed at the importance value 0.38.

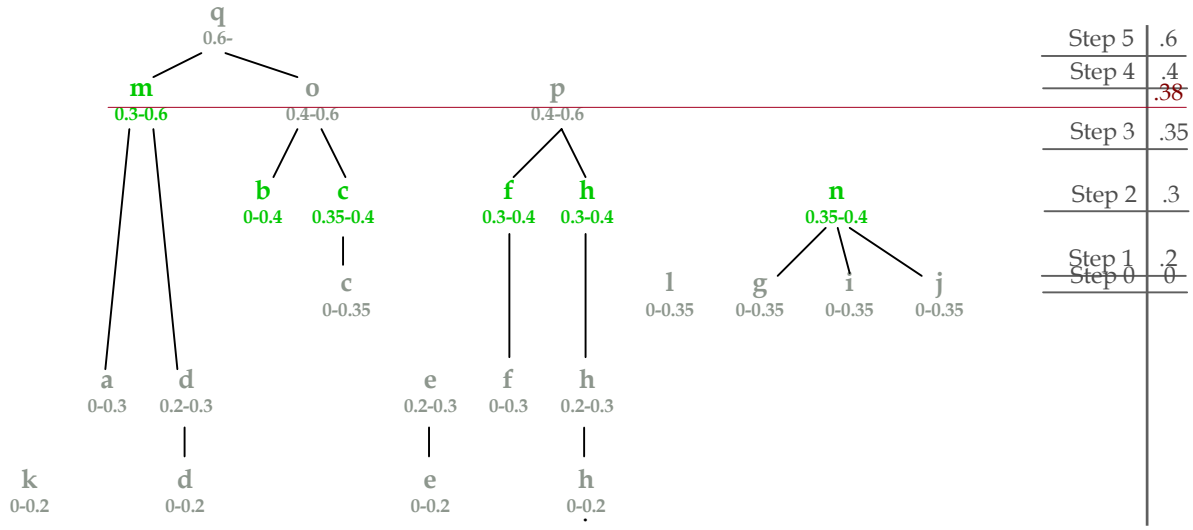


Figure 4.11: Edges to be shown for the importance level 0.38. Selected edges are drawn in orange.

A relation can be established between map scale and tolerance values of edge vertices. Once a scale is chosen for a map, it has to be translated to a tolerance value. For each edge to be shown at that scale, this tolerance value is used to select nodes from its BLG tree. The selected vertices, together with the start and end node of the edge, form the edge approximation that is to be displayed. The Douglas-Peuker algorithm is non-monotonic for the Euclidean distance used for tolerance calculation. That is to say, going down the tree does not guarantee decreasing tolerance values. For example, the BLG tree of edge 'j' – Figure 4.8, right – has decreasing tolerance values, whereas the BLG trees of edges 'g' and 'i' – Figure 4.7 and Figure 4.8 left, respectively – do not have decreasing tolerance values. For a given tolerance value, a BLG tree is descended to select vertices that will form the edge geometry for that value. For example, the geometry of edge 'g' (see Figure 4.6, and Figure 4.7 for its BLG tree) for a tolerance value equal to 0.32 is made of vertices v_2, v_3, and v_4, together with its start and end node. Figure 4.12 shows how the BLG tree of edge 'g' is descended to select the right vertices for the given tolerance 0.32. The tolerance of the root is bigger than the given tolerance, therefore v_4 is selected. Its right child, v_6, has a tolerance smaller than 0.32. The node is not selected; descending stops at that node. Its left child instead has a bigger tolerance than 0.32. Node v_3 is selected; descending goes further in this direction. The only child of v_3, node v_4, has a tolerance bigger than 0.32 and it is therefore selected.

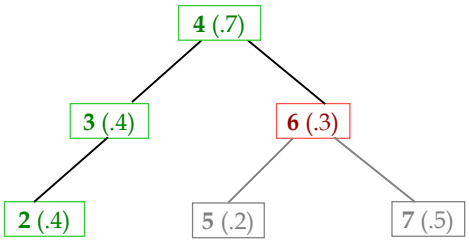


Figure 4.12: Descending a BLG tree to select vertices to be shown at tolerance value 0.32. Nodes in green are selected; the node in red is where the selection stops; its children (in grey) are not selected either.

4.3 Implementation of tGAP structure in Oracle Spatial

The tGAP structure is implemented as a collection of tables in Oracle Spatial (see Figure 4.13). Information about faces is stored in a `tGAP_face` table: face identifier, minimum bounding box, area size, importance range (`imp_low` and `imp_high` columns), parent id that allows to build the face tree, and a class attribute that is used from generalization. Information about edges is split into tables `tGAP_blg`, `tGAP_node`, and `tGAP_edge`. Geometry of edges is stored in the first two tables, which store BLG trees and start and end nodes, respectively. To remove redundancy, a BLG tree in the `tGAP_blg` table stores only the inner vertices of its edge. The start and end node of an edge are stored in the `tGAP_node` table, and retrieved via references `start_node_id` and `end_node_id` in the `tGAP_blg` table. Other columns in the BLG table store the BLG tree in `tree_source`, `child1_id` and `child2_id` store references to BLG trees for the joined BLG trees, and `top_tolerance` stores the root tolerance. When `tree_source` is filled, then `child1_id` and `child2_id` are empty, and vice versa. The `tGAP_edge` stores information of the edge forest: edge identifier, left and right face references, importance ranges that change with face references, and a reference to the corresponding BLG tree in `tGAP_blg` table.

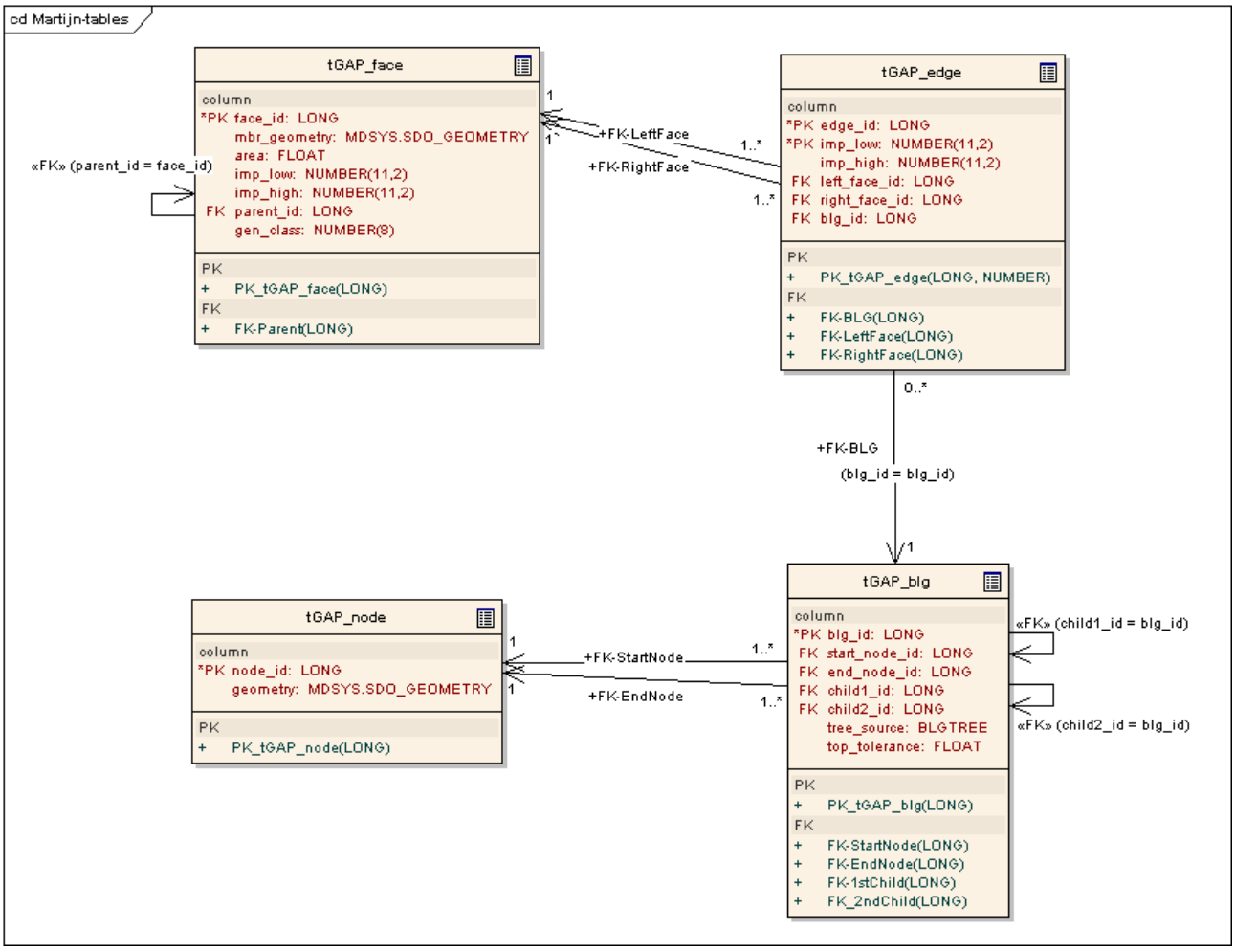


Figure 4.13: Diagram of tables and relationships that store the tGAP structure in Oracle Spatial.

A type `BLGTREE` is created for storing a BLG tree as a list of BLG nodes (in PL/SQL code):

```
create type BLGnode as object (
      x_coord number,
      y_coord number,
      left_node number,
      right_node number,
      tolerance float
);
create type BLGTREE as object varray(524288) of BLGnode;
```

The structure doesn't store explicitly the vertex position in the original (i.e. highest LoD) edge. Vertex positions are calculated following left and right references in the tree, when building edge geometry for a given tolerance.
Tables storing information about our example generalization (Figure 4.1) are given below: information about the face tree (Figure 4.2) is stored in `tGAP_face` table, and information about the edge forest (Figure 4.5) is stored in `tGAP_edge` table.

`tGAP_face` table

id	mbr_geometry	area	imp_low	imp_high	parent_id
1			0	0.3	8
2			0	0.2	7
3			0	0.4	10
4			0	0.35	9
5			0	0.35	9
6			0	0.2	7
7			0.2	0.3	8
8			0.3	0.6	10
9			0.35	0.4	10
10			0.4	0.6	11
11			0.6	0.9	

`tGAP_edge` table

id	imp_low	imp_high	blg_id	left_face_id	right_face_id
a	0.00	0.30	*a*	0	1
b	0.00	0.40	*b*	0	3
c	0.00	0.35	*c*	0	4
d	0.00	0.20	*d*	0	2
e	0.00	0.20	*e*	2	1
f	0.00	0.30	*f*	1	3
g	0.00	0.35	*g*	3	4
h	0.00	0.20	*h*	4	2
i	0.00	0.35	*i*	3	5
j	0.00	0.35	*j*	3	4
k	0.00	0.20	*k*	2	6
l	0.00	0.35	*l*	4	5
d	0.20	0.30	*d*	0	7
e	0.20	0.30	*e*	7	1
h	0.20	0.30	*h*	4	7
m	0.30	0.60	*m*	0	8
f	0.30	0.40	*f*	8	3
h	0.30	0.40	*h*	4	8
c	0.35	0.40	*c*	0	9
n	0.35	0.40	*N*	3	9
o	0.40	0.60	*O*	0	10
p	0.40	0.60	*P*	10	8
q	0.60		*Q*	0	11

A 3D functional index is built to insure fast access to features in a given spatial extent and a given importance range. It is based on a spatial index on bounding boxes of faces (or edges), plus importance values.

4.4 Progressive transfer and visualisation

Given a certain spatial extent (i.e. search rectangle) and a certain scale from the client, the server selects data from tGAP tables based on the spatial extent and a calculated importance range from (current and previous) scale. Then it starts sending these data progressively. The server sends edges ordered by their importance values (`imp_high` attribute alone, or in combination with `imp_low?`). Edge information sent by the server is their geometry, together with left and right face references. The topology of faces is to be built at the client side. A full importance range can be split into several intervals. The server collects all edges falling in an interval, which form boundaries of a partition of the given spatial extent. A complete (partition) collection is signalled to the client, which starts building topology for this collection edges. Faces are shown in the client screen. Other edges coming from the server start appearing in the screen. When a signal for

(another) complete edge collection is coming, the client starts building topology for the new faces, and visualises them in the screen.

Edge geometry can be calculated in the server side and send to the client. This requires re-sending edge geometry any time a more detailed shape is needed. Done differently, full information about edge can be sent to the client only once. Client has functionality to build the right geometry (edge detail) for any tolerance/scale. The following paragraph goes in more detail about ways to perform these.

Edge geometry can be created in the server from the BLG tree and a tolerance (calculated from scale) given from client. Edge geometry is then sent to the client. When more detail is needed for a received edge, a complete new (edge) geometry should be created and sent for the required detail. Another possibility is to send, instead of edge geometry, the BLG tree of the edge, together with its start and end node. In this case, the BLG structure, including functionality to create edge geometry, is required in the client side. A similar approach for progressive transfer of data is followed in the GiMoDig project. A third possibility is to rewrite a BLG tree in the server side as a sequence of vertices, and send this sequence to the client. For each vertex the sequence contains information on vertex position (in the highest LoD edge), vertex-coordinates, and vertex tolerance. The sequence can be ordered on tolerance values, which allows fast selection of vertices to be shown for an edge at a given scale/tolerance. For example, the order of vertices for the BLG tree of Figure 4.7 is $\langle(v_4, 0.7); (v_3, 0.4); (v_2, 0.4); (v_6, 0.3); (v_7, 0.5); (v_5, 0.2)\rangle$ – tolerances are not monotonically decreasing, whereas edge 'j', Figure 4.8 right, has decreasing tolerances: $\langle(v_5, 0.9); (v_2, 0.5); (v_6, 0.3); (v_4, 0.3); (v_3, 0.2); (v_7, 0.1)\rangle$. Selected vertices for a given tolerance will be ordered by the client according to the vertex position for visualisation on the screen.

4.5 References

1. Meijers, Martijn (2006). *Implementation and testing of variable scale topological data structures*. Master's Thesis TU Delft, 2006, 114 p.
2. Oosterom, Peter van(1990). *Reactive Data Structures for Geographic Information Systems*. PhD-thesis Department of Computer Science, Leiden University, December 1990.
3. Oosterom, Peter van (2005). Variable-scale topological data structures suitable for progressive data transfer: the GAP-face tree and GAP-edge forest *Cartography and Geographic Information Science, 32*, p.p. 331-346.
4. GiMoDig – "Geospatial Info-Mobility Service by Real-Time Data-Integration and Generalisation" (http://gimodig.fgi.fi/)

5 Client side technology

In the RGI 233 project ESRI-NL will build two prototypes. The first client prototype will be based on a simulated/desktop environment; a second prototype will run on mobile devices.
ESRI first builds a prototype client in the ArcMap environment using the ArcObjects model as an interface when a source web service (tGap Structure) is available (2007). Starting point will be the ArcObjects "CustomLayer" class. A custom layer allows you to support the drawing of new data formats and to customize how existing data formats display. Custom layers can be used to display dynamic data as well. In the ArcMap environment and with no requirements to provide functionality such as complex editing and data analysis, a custom layer object is an excellent solution.

Next we will look at the ArcGIS Server interface for mobile mapping on these kinds of services and the already built prototype code. ArcGIS Server has a special Application Development Framework (Mobile ADF) for working on vector servers online from a (small) mobile client having an uncertain mobile connection. Scenarios for building a Smartphone and a Pocket PC server client using the Mobile ADF are available (for an example, see Appendix 4).

Although it is hard to find a firm reason for having vector information on the client - we see that most mobile implementations do not need vectors on the client and most of our (ESRI Inc.) research goes into keeping functionality at the server if not needed at the client - the mobile ADF is the exception. The mobile ADF is designed for more complex applications on a mobile client, mostly editing. ESRI NL is nevertheless interested in the result, not only for mobile implementations but also for clients such as ArcGIS Explorer that really need a lot of vector information from a server. These applications use the Earth or Globe as an interface to the data. ArcGIS Explorer is the ESRI equivalent of Google Earth or Microsoft Virtual Earth. However ArcGIS Explorer is more open to different data sources and functionality from your own web services.

The Mobile ADF does not yet use a generalization concept like the tGap structure for retrieving data from the server. It can be that the client application will not be connecting directly to the tGap web service but uses a kind of local service having a local data structure to read the data from.
The tGap structure can be further developed to be used for smart feature retrieval in a Globe view. We are thinking of combination of a 3 dimensional tGap (qGap as a bTree compared to a qTree?) to be used to index data on the server for streaming retrieval starting with generalized data and getting less generalized data for the features closer to the center viewpoint. Streaming of objects from the server may also work from the center of the view outwards rather than from the upper-left to the lower-right corner. While panning, the system may anticipate on what is needed next, based on the panning direction. Furthermore, the user-perception of smooth zooming may be enhanced by having more detailed objects in the display appear and disappear by fading in and -out using transparency.

ESRI NL built a sample application (shown at the *GIS Conferentie*, see figure 5.1) that uses a ArcGIS Server Map Service (with mobile capabilities) for retrieving data and editing the data that is stored in a database on the server. This sample application will be the starting point for a sample application using the tGap web service.

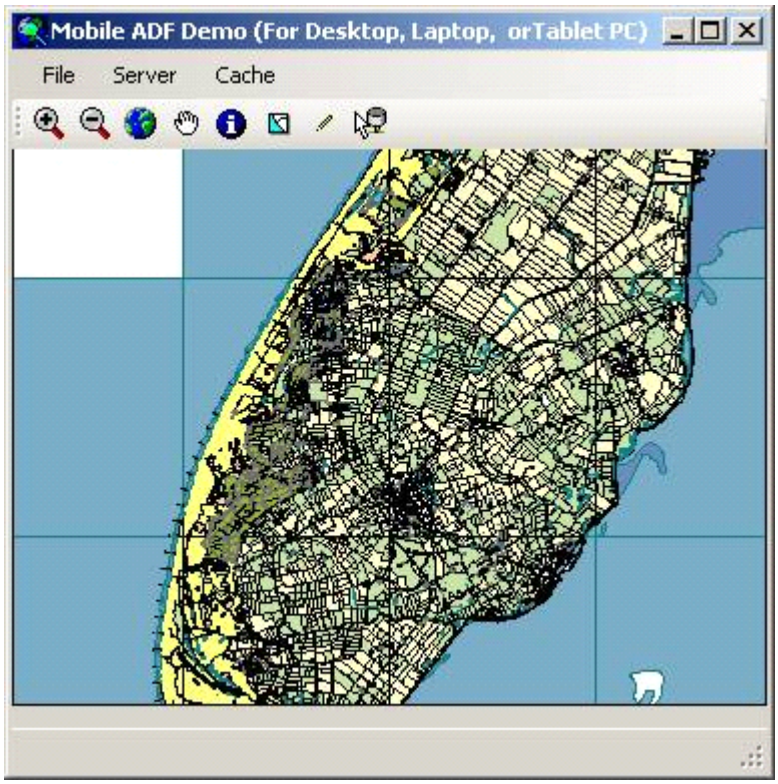


Figure 5.1.: Sample application.

Appendix 1 Amsterdam use case

Use case ´Stadstoezicht Amsterdam for project `Usable and well scaled mobile maps for consumers´

Introduction
The municipality of Amsterdam (in particular ´Dienst Geo Vastgoed´ and ´Dienst Stadstoezicht´) participates in the RGI 233 project to make user-friendly scalable geographic maps for various applications for citizens (end-users). This project is organized by TU Delft together with a number of representatives from the market sector. The project is co-financed via the Bsik/RGI organization. The project runs from 2006 up to and including 2008. ´Stadstoezicht´ and ANWB will develop separately a use case to test the developed concept. Below this use case is described and insight is given in the range of duties of ´Stadstoezicht´.

Short explanation of the range of duties of ´Stadstoezicht´

The ´Dienst Stadstoezicht´ (DTS) is in charge of supervising public space as follows:
a) executing parking management services,
b) supervising public space,
c) supervising living environment.

Parking management services

In the field of parking management services DST is de largest and most specialised service provider in the Netherlands. The following three main duties are distinguished:
1. enforcing the fiscal regime, also called ´paid parking´;
2. executing parking licence policy;
3. enforcing Regulations Traffic Rules and Traffic Signs, also called ´wrong parking´.

Supervising public space

The availability of suervisors in public space makes that the citizens of Amsterdam and visitor feel more safe. The supervisors present decrease petty crime and vandalism by signalling and reporting shortcomings and incorrect use of public space. Beside they participate in neighbourhoods project, support traffic supervision and act as host for tourist (information supply).

Supervising living environment

The activities regarding the living environment are amongst other things: supervising and enforcing rules for handing in litter at the right time and in the right way, taking away wrecked (moped) bikes and fighting inconvenience caused by dogs.

Use case

For its enforcement duties, especially regarding Parking, but in the future also for other enforcement duties, ´Stadstoezicht´ makes use of an automated system. The information of the various tariff areas, parking times, streets and addresses is managed in a spatial database. The obtained parking right (ticket machine, phone parking or license) is stored in a database. By means of hand terminals connected with this database the traffic warden checks if the parking person has a valid (digital) parking right. If not a a fine is calculated and written out (additional collection of taxes or notice of order).
Within the framework of the further development of the services of the ´Dienst Stadstoezicht´ the following case is relevant for this project.

´Stadstoezicht´ would like to offer the public the possibility to inform a person who wants to park about the parking regulations, tariffs and times in different streets and districts on the spot.

Elaboration case Parking Regulations

In Amsterdam parking rules have become more and more finely-woven during the past years and therefore require a precise recording. ´Stadstoezicht´ has recorded these rules in a spatial, electronic database. In this database all license areas, all tariff areas with block time for incidental parking including special regimes, like fiscal blue zones in shopping streets, have been recorded.

At the moment parking citizens are informed via user-friendly folders. For each district a folder has been produced including an overview map with all tariff and license areas and special regimes.

As far as ´Stadstoezicht´ is concerned the case ´should focuse on a solution which shows the citizen with the help of mobile services a map of Amsterdam on his PDA, mobile phone or smartphone. By means of GPS or by entering street name and house number he/she should have the possibility to locate his/her position on the displayed electronic map. Making use of a zoom in function it should be possible that at the moment individual streets can be recognized to ´point out ´ a street and to make the current parking regime for the street (part) in question visible. Besides it desirable that when zooming in at the chosen street of street part also the house numbers and the located ticket machines become visible.

The parking regime describes the following parts:
- license area (as a code);
- block time with corresponding tariff codes and tariffs (e.g. during the day low tariff but with a max. of one hour parking duration and in the evening normal tariff with out parking duration limitations);
- special regulations, like a certain kind of license is not allowed in this area.

Furthermore it should be possible to show car parks in the vicinity including tariffs and pay possibilities.

Appendix 2 ANWB use case

RGI 233 Use case ANWB functional requirements

Introduction
Some of the most grateful users of maps are tourists. Maps help tourists with questions such as: Where am I? What is this building/object I am facing? and How do I get to the railway station? Tourists are probably also the most critical users because in most cases they do not know the infrastructure. They therefore very much depend on maps for finding their way. A clear and well-detailed map with a good overview is essential to them.
In use case 2 - Mobile tourist information - we are facing the challenge to fulfil the needs of this demanding tourist with a device with one of the smallest screens possible: the mobile phone. There are some other disadvantages that make the device virtually unsuitable for this purpose: limited bandwidth, limited processor power, limited controlling possibilities and so on. Why would the tourist use his mobile to view a map? Because it is readily available, it is small and it is interactive.
The challenge will be to overcome all the disadvantages of the mobile device and offer a map solution that resembles the experience of using a print map: promptly available, clear and distinctive information, seamless panning and zooming and providing a good overview to name a few.

The final objective of use case 2 is the development of a combined street finder and route planner. This document describes the functional requirements of the street finder. The route planner will be defined and developed in a later stadium of the project.

Requirements
General description
With the street finder maps can be presented on base of address input by the user. The address is pinpointed on the map. The map can be zoomed, panned, rotated and tilted (3D). On certain map scales icons that represent tourist attractions and other points of interest (POI) are shown on the map. By selecting these icons more information can be obtained.

Address input
Address entry files available are:
- Street
- Number
- Postal code
- City

Not all fields have to be filled out. The following entry combinations (minimum) are accepted by the system:
- Postal code
- City
- Street-Postal code
- Street-City

In case the input concerns a larger area, a map with the geographical centre of this area is shown. The system should accept partial entries (e.g. Amsterd). In case the input matches more than 1 candidate (based on all entered data available), it should present the candidates to the user in order for him to select the required candidate.

Map coverage
Netherlands.

Hardware
To be decided.

Browser
The application should be browser based.

Minimal screen size
Available for screens with at least a resolution of 128 x 128 pixels, 4.096 colours.

Performance
After the entered address has been validated, the map should be fully visible within 5 seconds, based on the bandwidth specification provided by the telecom operator.

Map orientation

Zooming
Zooming should be possible in the direction left, right, up, down.
Zooming should be fluent and seamless, map should stay visible during zooming.
Minimum zoom level shows a scale of 1:2500 (screen size 4 * 4 cm).
Maximum zoom level shows the Netherlands.

Panning
Panning should be possible in the direction left, right, up, down.
Panning should be fluent and seamless, map should stay visible during panning.

Rotation
It should be possible to rotate the map clockwise and counter clockwise.
Rotating should be fluent and seamless, map should stay visible during rotating

Tilting
It should be possible to get a 3D map view comparable to the 3D view provided by most current navigation systems. Zooming, panning and rotating functionality should also be available in 3D mode

Presentation of street names / road numbers
Text should never be up side down.

Map controls
The map is controlled by using the cursor buttons and the central button on the keyboard of the mobile device. By default the cursor buttons control the panning of the map. With the central button the function of the cursor button can be changed. When clicking the button an overlapping (transparent) screen appears through which the desired button functionality can be selected:

- Panning;
- zooming (left and right button are idle);
- tilting (up and down button) and rotating (left and right button);
- cursor.

Tourist information in the map
On a zoom level of 1:10.000 or lower, icons are visible that represent points of interest. Different icons represent different types of points of interest. In the "cursor-mode", when the cursor is on top of an icon, a pop-up shows more information on the POI.

Appendix 3 Creating the tGAP face tree

```
Create Faces & Edges list from original data      /* SQL statements to tables of the highest LOD data */
Set n = number of original faces


/* Assign importance values to faces */
for i = 1 to n
        imp_low(i) = 0
        imp_high(i) = Importance(i)     /* e.g. the function may be Area(i)*ClassWeight(i) */
end for

Order faces on their importance in a list Faces
/* Every step decreases by 1 the no of faces; the cycle has n-1 steps */
do until one face is left
        Get the list of neighbours of Faces(1)    /* from the list of edges */
        Set best-comp = 0       /* compatibility value */
        Set best-nbhd = 0       /* neighbour face index */


        /* Calculate the compatibility with Faces(1)
        for each neighbor j
                comp = Compatibility(1,j)       /* e.g. as Length(Boundary(1,j)) * ClassSimilarity(1,j) */
                if comp > best-comp
                Set best-comp = comp
                        Set best-nbhd = j
                end if
        end for

        Merge face i with face best-nbhd to face ++n            /* Merge builds the edge forest */
        /* Change importance high value for face best-nbhd */
        imp_high(best-nbhd) = imp_high(i)

        /* Set parent-child relation */
        pid(best-nbhd) = n
        pid(i) = n

        Class(n) = Class(best-nbhd)
        /* Assign importance values to face n */
        imp_low(n) = imp_high(1)
        imp_high(n) = Importance(n)   /* e.g. (Area(1) + Area(best-nbhd)) * ClassWeight(n) */
        Remove faces 1 & best-nbhd from Faces list
        Add face n according to importance order to the Faces list
end do
```

This algorithm works on the assumption that new faces have higher importance (as calculated by the function) than their merged components. Words in italic indicate functions/procedures.

Appendix 4 Mobile ADF developer scenario for Smartphone

This walkthrough is for developers who wish to create and deploy a simple ArcGIS Mobile application for the Smartphone environment. It demonstrates how to create the application using the ArcGIS Mobile Map Control, the ArcGIS Mobile API and parts of ADO.NET.
You can find this sample in:
<ArcGIS_install_location>\DeveloperKit\Samples\Server\NET\Mobile_Applications\Walkthrough_SP05CSharp .zip

Project Description
The application will display map data that has been extracted from an ArcGIS Map Service using a Map Control, provide basic navigation tools such as zoom and pan, and include an identify function to show feature attributes. Once you complete this walkthrough, you can then extend the application you build with additional functionality.

Concepts
There are some important concepts you should understand before following this example. As a prerequisite, please read through the Mobile SDK conceptual documentation found under the heading *Developing Mobile Applications using the Mobile ADF* to gain an understanding of the mobile framework and architecture. You should also have a good understanding of Visual Studio .NET 2005 and how to create a Smartphone application.

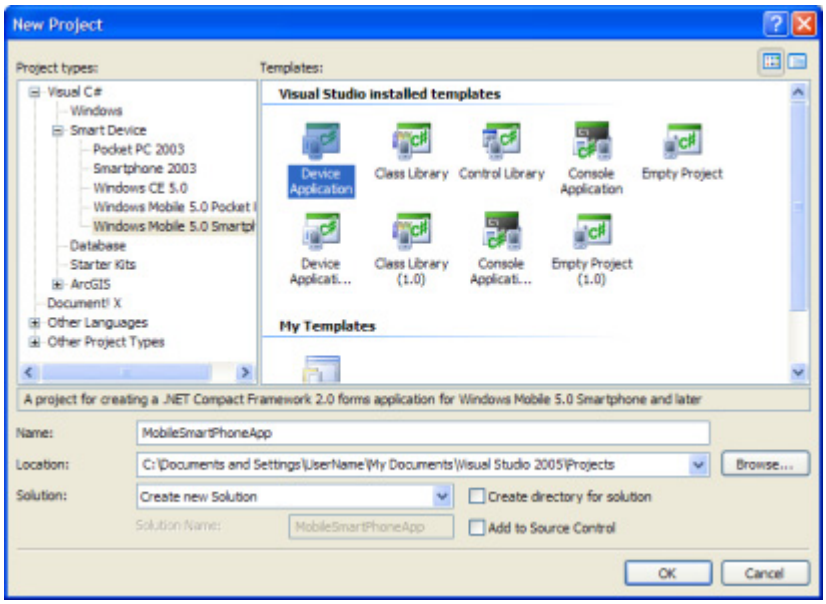
ArcGIS Mobile Components
The ArcGIS Server Mobile SDK provides serveral Visual Studio components to help you develop mobile applications. The primary components that you will work with in this scenario are the Map Cache and the Map. The Map Cache component requests data stored in a folder called the *Storage Path.* Th map cache you will use has previously been extracted from a map web service that was published with mobile capabilities. You do not need to create a mobile web service to complete this walkthrough. The Map component will display the contents of the map cache that has been created for you without having to connect to a server. Additional components are used to navigate the map itself and identify attribute information for given features. For more information on the components used in this walkthrough please refer to the ArcGIS Server Developer Help.

Requirements
In order to walk through this scenario you will need the following installed on your machine:
- Visual Studio .NET 2005 with C#
- Windows Mobile 5.0 Smartphone SDK (download from Microsoft)
- The ArcGIS Mobile SDK

This example is written in C# and will be deployed to the Windows Mobile 5.0 Smartphone emulator installed with Visual Studio. You should also reference the ArcGIS Mobile ADF object model diagram while proceeding through this example. You can find the OMD in ArcGIS Developer Help.

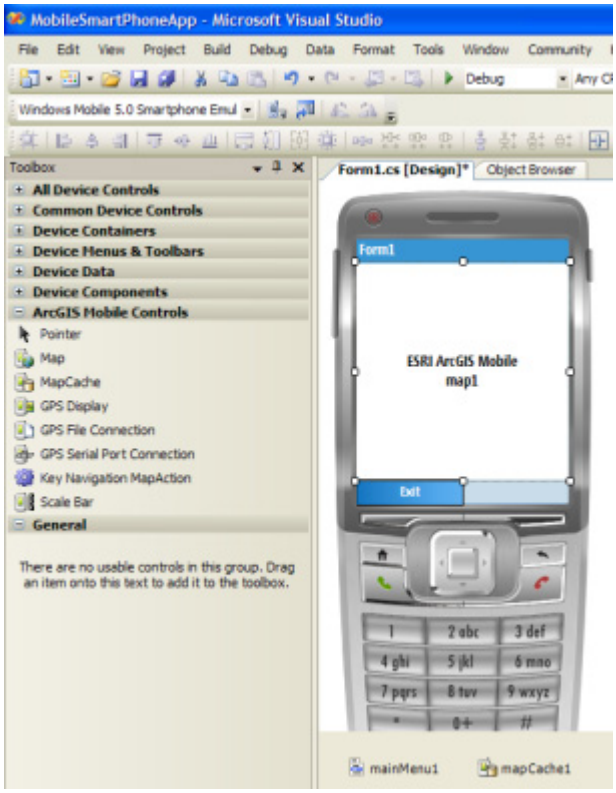
Implementation
In this example you will create a simple application within the Windows Mobile 5.0 Smartphone emulator that allows you to open a mobile map cache and display the layers in a map control. You will then add additional controls that will let you navigate the map and identify features within the layers. The first step is to create the new project.

Creating a new project
1. Start Visual Studio .NET 2005.
2. On the **File** menu, point to **New**, then click **Project** .
3. In the **New Project** dialog box, under **Project Types**, click **Visual C#,** expand **Smart Device** then click **Windows Mobile 5.0 Smartphone** .
4. In the **Templates** pane click **Device Application** .
5. Enter an appropriate name for the project.
6. Click **OK**. This will create a new project for you.



Using the Map Control at design time

The ArcGIS Mobile controls are added to the Visual Studio toolbox when you install the ArcGIS Mobile ADF. They are located in the ArcGIS Mobile Controls tab within the toolbox. To use the controls within your application you must drag and drop them from the Toolbox to your windows form.



Within the following section, you will add a **Map** and **MapCache** component to the application and configure them via properties.

1. Drag the **Map** component from the toolbox onto the form.
2. Resize the **Map** to fill the desired area as indicated by the screen shot above.

When the Map component is added to the form, a MapCache component will be added to your project automatically. Your form should now contain a Map, called Map1, and a MapCache component called

mapCache1. Additionally two new references have been added to your project, one to the Mobile ADF windows assembly and the other to the .NET System.Web.Services assembly.
You now need to configure the components via their properties.

1. Right click the **Map** Control and choose **Properties** on the context menu to display its properties within the **Properties** window.
2. Within the **Properties** window for Map1, set the **MapCache** property to **mapCache1** .
3. Now display the properties for the mapcache. Click the component **mapCache1** in the **Component** pane within the application.
4. Within the **Properties** window, select the **StoragePath** property and type **\Temp\Redlands.** Note that you will need to copy the Redlands map cache from the sample data folder to your temp folder.

You now need to add code to create the map cache on disk and retrieve data from the server while the application is running.

1. Create a **form load** event.
2. Switch to code view and add a using statement to include the ArcGIS Mobile assembly:

```
[C#]
using ESRI.ArcGIS.Mobile;
```

3. Add the following code to the **form1_load** event:

```
[C#]
 private void Form1_Load(object sender, EventArgs e)
 {
 if (!mapCache1.IsValid)
  {
   MessageBox.Show("Map Cache is not valid!");
   return;
  }
 try
  {
   mapCache1.Open();
  }
 catch
  {
  MessageBox.Show("Cannot open map cache");
  }
 }
```

This code will open the mapcache, retrieve data for the current map extent and draw it to the display.

Adding Map Navigation

You can add map navigation features such as zoom and pan via the Key Navigation MapAction. This provides a set of ready made navigation functions driven by the Smartphone keys and cursor.

Add the Key Navigation MapAction to the project.

1. Double click the **Key Navigation MapAction** from the Visual Studio Toolbox to your form. The map action will be added as a component to your project.
2. Set the **Map** property for the map action to **Map1**.

Application exit

Unlike PocketPC or Windows applications, Smartphone application forms do not present an Ok or X in the top right corner to close the form. You will need to add code to properly close your application. For this walkthrough you will close the application by pressing soft key 1.

1. Return to the designer view for your form.
2. Click the **menu** area above **Soft Key 1** in the designer view to create a new menu entry (MenuItem1).
3. Type **Exit** for the text property.
4. Double click the menu item to create a code stub.
5. Enter the following code in menuItem1_Clickevent.

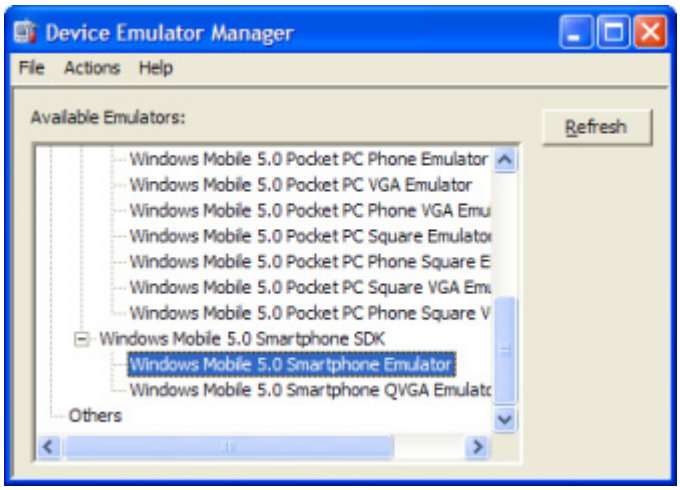
```
[C#]
private void menuItem1_Click(object sender, EventArgs e)
{
 this.Close();
}
```

Compile the application. On the **Build** menu, choose **Build Solution.** This will display the **Repair Native Dll's** dialog prompting you to add the Mobile ADF unmanaged library to the solution. Click **OK** to add the dll. This dialog is displayed if the unmanaged dll is missing or has a broken link and you build or run your application.

Configuring the Smartphone Emulator

You will use the Windows Mobile 5 Smartphone emulator to test the application but before you can it must be started and then virtually cradled.

1. From the Visual Studio **Tools** menu, choose **Device Emulation Manager**.
2. In the **Device Emulator Manager** dialog, locate the **Windows Mobile 5 Smartphone Emulator**, right click and select **Connect**. This will start the device emulator.
3. Right click again on the **Windows Mobile 5 Smartphone Emulator** entry and select **Cradle**. This will cradle the emulator via ActiveSync. Select a **Guest** partnership if asked for the type of ActiveSync relationship. Hint: If ActiveSync does not automatically connect to the device you can try this manually in ActiveSync by selecting **File** -> **Connection Settings**, then click the **Connect** button and follow the wizard.



NOTE: Before running the application, you need to copy the map cache data from the samples' data directory to the emulators' \Temp folder. From the Tools menu in ActiveSync, click Explore Device and navigate to \Temp and paste the Redlands folder from <ArcGIS_Server_Install_Location>\DeveloperKit\SamplesNET\Server\data\mobile\MapCaches\Redlands.

You may now compile and run the application. Select the Windows Mobile 5 Smartphone emulator as the target device. At this stage the form should display layers from the map service in the map control.

Run the application
The Smartphone emulator does not have a file explorer to locate and run the executable when deployed, for this exercise you can run the application via the debugger.
1. From the Visual Studio **Debug** menu, choose **Start Debugging**.
2. If asked, deploy the application to the emulator. The application should start after a while.
3. Close the application via **Soft Key 1**

Application Hint: Rather than using the debugger to run the application you can setup a shortcut from the excutable to the Windows start menu on the device. Explore the device using Activesync to setup the shortcut.

Experiment with the application. Zoom and pan the map via the Key Navigation keys.

Deployment

The application may be deployed to a Smartphone device using either of the methods described in the Deploying mobile adf applications topic.

Appendix 5 Project team

TU Delft – Section GIS technology:
- Arta Dilo
- Elfriede M. Fendel
- Martijn Meijers
- Peter van Oosterom
- Theo Tijssen

ITC:
- Corné van Elzakker
- Jantien Stoter

TNO Defense, Security and Safety:
- Guido te Brake
- Rosemarijn Looije
- Mark Neerincx

1Spatial:
- David Allen
- John Swinton

Municipality of Amsterdam:
- Fred Harms
- Ad van der Meer
- Jeroen de Vries

ANWB:
- Iwan Banens
- Wilmar Visschers
- Leendert Jan Zonneveld (until January 1, 2007)

ESRI:
- John van Smaalen
- Jeroen van Winden

Corresponding address

Delft University of Technology
OTB Research Institute for Housing,
Urban and Mobility Studies
Jaffalaan 9, 2628 BX Delft
PO Box 5030, 2600 GA Delft
The Netherlands
tel. +31 (0)15 278 30 05
fax +31 (0)15 278 44 22
e-mail: mailbox@otb.tudelft.nl
www.otb.tudelft.nl