Utveckling av en Web Map Service-klient –
Standarder och tillämpning

Development of a Web Map Service Client –
The Application of Standards

Tobias Lennartsson

Avdelningen för fastighetsvetenskap
Lunds Tekniska Högskola
Lunds Universitet

Department of Real Estate Science
Lund Institute of Technology
Lund University, Sweden

ISRN LUTVDG/TVLM 04/5100 SE

Avdelningen för
FASTIGHETSVETENSKAP
Lunds Tekniska Högskola
Lunds Universitet
Box 118
221 00 LUND

Department of
REAL ESTATE SCIENCE
Lund Institute of Technology
Lund University
P.O. Box 118
SE-221 00 LUND
SWEDEN

Utveckling av en Web Map Service-klient – Standarder och tillämpning

Development of a Web Map Service Client – The Application of Standards

Examensarbete omfattande 20 poäng utfört av:
Tobias Lennartsson
Civilingenjörsutbildningen inom Lantmäteri, 180 poäng
Lunds Teknisk Högskola

Handledare:
Lars Harrie, Avdelningen för Fastighetsvetenskap, Lunds Tekniska Högskola

Examinator:
Klas Ernald Borges, Avdelningen för Fastighetsvetenskap, Lunds Tekniska Högskola

Juli 2004

ISRN LUTVDG/TVLM 04/5100 SE

Sökord: Web Map Service, WMS-klient, SVG, Open GIS Consortium, öppen standard

Keywords: Web Map Service, WMS client, SVG, Open GIS Consortium, open standard

Language: Swedish

Förord

Examensarbetet har utförts på GIS-centrum vid Lunds universitet och avslutar mina studier inom lantmäteriutbildningen vid Lunds tekniska högskola.

Jag vill tacka Lars Harrie (avdelningen för fastighetsvetenskap vid LTH) vars handledning har inneburit värdefull hjälp. Jag vill också tacka min examinator Klas Ernald Borges (avdelningen för fastighetsvetenskap vid LTH) och Peter Segerstedt som opponerade på arbetet och gav många värdefulla synpunkter. Mikael Grimheden, Anders Pikkuniemi, och Lassi Lehto vill jag tacka för information om de WMS-servrar de administrerar.

Jag vill också tacka personalen på GIS-centrum där jag haft min arbetsplats. Jag riktar även ett tack till alla på SVG Developers-forumet och däribland speciellt Frank Steggink, Mathias Fatene och Holger Will för värdefull hjälp. Och slutligen ett varmt tack till Ann-Sofi för att du står ut med mig och ger mig stöd i både regn och solsken.

Lund 2004-08-16

Tobias Lennartsson

Sammanfattning

I första delen av denna rapport beskrivs ett antal olika typer av Internet-baserade karttjänster samt de standarder och tekniker som används för att skapa dessa karttjänster. Därefter beskrivs ett antal nya standarder för distribution av geografiska data via Internet. Rapporten behandlar även projektet GiMoDig vars systemarkitektur exemplifierar användandet av dessa nya standarder. I rapportens andra del beskrivs utvecklingen av en klientapplikation för visning av kartbilder från en *Web Map Service*-server.

Karttjänster har under de senaste åren blivit allt vanligare på Internet och de har också blivit en vanligare del av andra Internet-baserade tjänster. Det finns ett stort antal olika metoder för visning av kartor via Internet där de enklaste består av statiska avbildningar av papperskartor och de mest avancerade tillåter analys och presentation av geografiska data i vektorformat.

För kommunikation via Internet används ett stort antal standarder, några av dessa är speciellt intressanta för distribution av geografiska data och kartbilder. Dessa standarder innefattar bland annat kommunikationsprotokoll, märkspråk, grafikformat och programmerings- och skriptspråk.

Open GIS Consortium (OGC) arbetar med standardisering för att underlätta utbyte av geografisk information och har medlemmar från näringslivet, universitet och myndigheter. OGC har under de senaste åren standardiserat överföringsformat för geografiska data och ett flertal gränssnitt för distribution av geografisk information via Internet. Gränssnitten är utformade som webbtjänster som går att kombinera med varandra.

EU-projektet GiMoDig som pågått sedan 2002 har som mål att skapa ett system för sömlös distribution av kartdata från flera olika länder. Dessa kartdata ska levereras direkt från de olika ländernas nationella geografiska databaser och ska integreras och generaliseras i realtid. GiMoDig använder i sin systemarkitektur ett flertal standarder från OGC.

En webbaserad klientapplikation för visning av kartbilder från *Web Map Service*-servrar har skapats i examensarbetet. Klientens grafiska användargränssnitt är skapat i Scalable Vector Graphics-format (SVG) och JavaScript har använts för att möjliggöra interaktivitet. Klientprogrammet visar på enkelheten i att skapa en klientapplikation utifrån en öppen standard. Under utvecklingsarbetet har också tänkbara utvidgningar av standarden, för specialfallet vektorgrafik, diskuterats och testats.

Tester av klientens kompatibilitet gentemot ett antal olika serverimplementationer av WMS-standarden har utförts med nästan uteslutande goda resultat. Kartbilder visades på ett korrekt sätt i alla tester. Visning av attributdata misslyckades i ett fall men när attributdata efterfrågades i ett annat format lyckades förfrågningen. Slutsatsen är att det tack vare den öppna standarden med enkla medel går att skapa en WMS-klient som fungerar gentemot flera av varandra oberoende WMS-servrar.

Abstract

In the first part of this report a number of types of Internet based map services and the standards and techniques behind them are described. Thereafter a number of new standards for distribution of geospatial data via the Internet are described. The report also covers the system architecture of the GiMoDig project, which exemplifies the use of these new standards. The second part of the report describes the development of a client application for viewing of map images from Web Map Services (WMS).

During the latest years map services on the Internet have become more common and they have also become a more common part of other Internet based services. There are a great number of methods for viewing of maps via the Internet among which the simplest are static images of paper maps and the most advanced allow analysis and presentation of geographical vector data.

There are a great number of standards for communication via the Internet. Some of these are of special interest for distribution of geographical data and map images. These standards, among others, include communication protocols, mark-up languages, graphics formats and programming and scripting languages.

The Open GIS Consortium (OGC) develops standards within the area of exchanging geographical information and its members are within the business world and among universities and authorities. In the last few years the OGC has standardised transfer formats for geographical data and a number of interfaces for distribution of geographical information via the Internet. These interfaces are designed as combinable web services.

The goal of the European Union funded project GiMoDig, which started in 2002, is to establish a system for seamless distribution of cartographic data from several countries. These cartographic data are to be delivered directly from the geographical databases of the contributing countries and will be integrated and generalised in real-time. In its system architecture GiMoDig uses several of the OGC standards.

Within the project a web based client application for viewing of map images from WMSs has been developed. The graphical user interface of the application uses Scalable Vector Graphics (SVG) and to add interactivity JavaScript has been used. The client application shows the relative simplicity of creating a client application using an open standard. During the development of the application possible extensions of the standard to handle the use of vector graphics for cartographic visualisation have been discussed and tested.

The client application's compatibility has been tested by the means of communication with a number of server implementations of the WMS standard. The test results were almost exclusively successful. The map image was shown in a correct way in all the tests. In one case viewing of attribute data failed but when the information was requested in another format the request succeeded. The conclusion is that thanks to the open standard a WMS client capable of communicating with several independent WMS servers can easily be created.

Innehåll

Förord III

Sammanfattning IV

Abstract V

Innehåll VI

1 Inledning 1
1.1 Bakgrund 1
1.2 Syfte 2
1.3 Metod 2
1.4 Rapportutformning 2

2 Karttjänster på Internet 4
2.1 Datorgrafik 4
2.2 Klassificering av karttjänster 5
2.3 Tjocka och tunna klienter 8

3 Standarder för Internet 10
3.1 Uniform Resource Identifier (URI) 10
3.2 HyperText Transfer Protocol (HTTP) 10
3.3 Common Gateway Interface (CGI) 11
3.4 HyperText Markup Language (HTML) 12
3.4.1 HTML-baserade karttjänster 12
3.5 eXstensible Markup Language (XML) 13
3.5.1 XML-dokument 13
3.5.2 DTD och XML-Schema 14
3.5.3 Namespace 16
3.5.4 eXstensible Stylesheet Language Transformations (XSLT) 17
3.6 Grafikformat 17
3.6.1 Rastergrafik 17
3.6.2 Vektorgrafik 18
3.7 Scalable Vector Graphics (SVG) 19
3.7.1 Exempel på SVG-dokument 19
3.7.2 Ändring av SVG-objekts utseende 21
3.7.3 SVG och filstorlek 22
3.7.4 Kartografiska tillämpningar 23

4 Programmeringstekniker 25
4.1 Document Object Model (DOM) 25
4.2 ECMAScript 26
4.2.1 ECMAScript och rasterkartor 27
4.2.2 ECMAScript, DOM och vektorkartor 27
4.3 Java 28
4.3.1 Java-Applet-program 29

5 Standarder från Open GIS Consortium 30
5.1 Geography Markup Language (GML) 30
5.2 Web Map Service (WMS) 31
5.2.1 GetCapabilities 32

5.2.2 GetMap 33
5.2.3 GetFeatureInfo 35
5.3 Web Feature Service (WFS) 36
5.4 OpenGIS Location Services (OpenLS) 37
5.5 Styled Layer Descriptor (SLD) 38

6 GiMoDig-projektet 39
6.1 Systemarkitektur 40
6.1.1 Förfrågan efter kartdata 40
6.1.2 Behandling av kartdata 41
6.2 Användande av andra tjänster 42

7 Utveckling av klientprogram 46
7.1 Översikt 46
7.2 Användargränssnitt 47
7.3 Använd teknik 52
7.4 GetCapabilities-anrop 53
7.5 GetMap-anrop 53
7.5.1 Kartdata i SVG-format 54
7.6 GetFeatureInfo-anrop 57

8 Test av WMS-klientens kompatibilitet 59
8.1 deegree 59
8.2 MapServer 59
8.3 Demis 60
8.4 Intergraph – GeoMedia WebMap Professional WMS Adaptor Kit 60
8.5 ESRI – OGC Interoperability Add-On och ArcIMS WMS Connector 60
8.6 GiMoDig 61
8.7 Utvärdering av kompatibilitetstest 61

9 Diskussion 62

10 Slutsatser 64

Referenser 66
Litteratur 66
Internet 68
Personlig korrespondens 70

Appendix A. Programstruktur

Appendix B. Programkod

Appendix C. Ordlista

Index

1 Inledning

1.1 Bakgrund

Karttjänster på Internet har under de senaste åren blivit allt vanligare och från att i stort sett bara ha funnits på renodlade kartwebbplatser finns nu kartor som en integrerad del av många andra tjänster. Många företag och andra organisationer visar till exempel var de finns med hjälp av karttjänster, budfirmor visar positionen för ett paket och kollektivtrafikföretag visar sina turer med hjälp av kartor.

Det finns ett flertal tekniker för visning av kartor på Internet, men få av dessa möjliggör interaktivitet och analys av de geografiska data som visas. Den enklaste typen av karttjänster består av en statisk kartbild utan möjlighet till varken zoomning eller panorering. Vissa karttjänster använder ett antal rasterbildar med hyperlänkar sinsemellan så att användaren genom att klicka i en kartbild kan panorera eller zooma till nästa kartbild. Mer avancerade karttjänster använder sig av program på webbservern som genom att tolka användarens mus- och tangentbordsinmatningar, från geografiska databaser i realtid, skapar kartbilder som visas i användarens webbläsare.

Det finns ett flertal aktörer som arbetar med standardisering av informationsflöden på Internet. World Wide Web Consortium (W3C) och Open GIS Consortium (OGC) är två av de viktigaste aktörerna när det gäller geografisk informationsbehandling via Internet. W3C är ett standardiseringsorgan för utveckling av standarder för kommunikation på Internet och OGC är ett standardiseringsorgan inom området geografiska informationssystem (GIS). OGC har bland annat standardiserat överföringsformat för geografiska data samt utvecklat standarder för hur geografiska data ska efterfrågas och hur svar på förfrågningar ska levereras. (OGC 2003)

Vid slutet av 2001 fastslog OGC en standard för efterfrågning och visning av kartor på Internet. Standarden kallas Web Map Service (WMS) och den anger hur kartor, attribut till dessa kartor samt metadata för karttjänsten ska efterfrågas och levereras. WMS-tjänster får leverera kartor i raster- eller vektorformat. Ett möjligt vektorformat är Scalable Vector Graphics-format (SVG). SVG är en standard för vektorgrafik från W3C som bland annat möjliggör identifiering av enskilda objekt i kartbilden. Detta innebär att till exempel enskilda punkt-, linje- eller ytobjekt i kartbilden kan identifieras och hanteras var för sig. (W3C 2003b).

Geospatial info-mobility service by real-time data-integration and generalisation (GiMoDig) är ett EU-projekt som bland annat har tagit fram en modell för distribution av geografiska data från nationella kartverk till mobila enheter. Modellen bygger på en lagerstruktur med fyra lager och ett klientlager där varje lager motsvarar en tjänst som behandlar och vidarebefordrar geografiska data. Kommunikationen mellan klientlagret och det närmast underliggande lagret sker enligt WMS-gränssnittet. Det understa lagret består av de ingående ländernas nationella geografiska databaser vilket innebär att de data som levereras till användarna alltid är uppdaterade.

1.2 Syfte

Syftet med examensarbetet är att skapa ett klientprogram för kommunikation med karttjänster enligt Web Map Service (WMS). Ett mål med skapandet av WMS-klienten är att undersöka Web Map Service-standardens GetFeatureInfo-funktionen och dess tillämpbarhet på kartbilder i SVG-format. Arbetet ska också undersöka öppna standarder som används i klientprogrammet och i GiMoDig-projektet. Även GiMoDig-projektets systemarkitektur ska beskrivas.

1.3 Metod

Den teoretiska delen av arbetet utgörs av en litteraturstudie om GiMoDig-projektets systemarkitektur samt beskrivningar av några av de standarder som används i examensarbetet. Litteraturen innefattar specifikationer av standarder, tekniska artiklar, böcker samt webbplatser och Internetbaserade diskussionsgrupper i ämnet.

Litteraturstudien ska också ligga till grund för den praktiska delen av arbetet. Det praktiska arbetet består av implementation av ett klientprogram för kommunikation med WMS-tjänster. Kommunikationen ska ske med hjälp av WMS-standardens *GetMap* för erhållande av kartbild samt *GetFeatureInfo* för erhållande av attributdata. Användandet av dessa två typer av förfrågningar ska undersöka möjligheterna att utföra enklare analyser av geografiska data i en WMS-klient. Klientprogrammet ska kunna hantera kartdata i GIF-, PNG- och SVG-format.

1.4 Rapportutformning

Rapportens första del behandlar GiMoDig-projektets systemarkitektur och de tekniska specifikationer klientprogrammet bygger på. Den andra delen beskriver klientprogrammet och utvecklingen av detta.

Teoridelen inleds med en allmän beskrivning av karttjänster på Internet och fortsätter därefter med beskrivningar av ett antal standarder för kommunikation via Internet som alla används, direkt eller indirekt, i GiMoDig-projektets systemarkitektur. Kapitel fyra beskriver olika programmeringstekniker som används på Internet och kapitel fem beskriver ett antal öppna standarder från Open GIS Consortium. Sist i teoridelen beskrivs GiMoDig-projektet och dess systemarkitektur.

Den praktiska delen består av en beskrivning av hur klientprogrammet kan användas och hur det är konstruerat samt en beskrivning av tester av klientapplikationens kompatibilitet gentemot ett antal WMS-servrar.

I arbetet nämns begreppet "kartdata" vid ett flertal tillfällen. Läsaren bör beakta skillnaden mellan geografiska data och kartdata. Med kartdata menas en visualisering (vanligen i form av en kartbild) av ett visst set geografiska data. För kortfattad förklaring av ett antal tekniska termer och förkortningar hänvisas till ordlistan som återfinns i Appendix C. Ordlista.

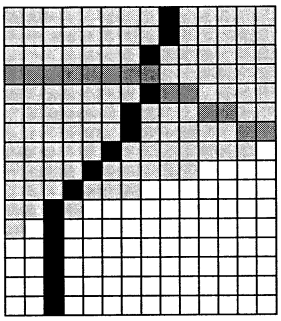
Teoridel

2 Karttjänster på Internet

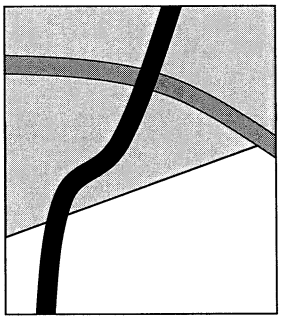
Det finns ett stort och växande antal tjänster som via Internet tillhandahåller kartinformation av olika slag. De enklaste tjänsterna är vanliga webbsidor som innehåller statiska kartbilder i rasterformat och de mest avancerade kan innehålla Java-applikationer för avancerad visualisering och analys av kartdata.

2.1 Datorgrafik

Kartbilder och grafiska framställningar i allmänhet kan av datorer representeras på två olika sätt: rastergrafik och vektorgrafik. I en rasterbild representeras föremål med hjälp av celler i en rektangulär matris (Figur 2.1). Varje cell, eller bildelement, kan ha olika värden vilka bestämmer bildelementets färg. Ett geografiskt objekt kan representeras av noll till flera celler beroende på matrisens storlek eller bildens upplösning. Med detta menas att om upplösningen inte är tillräckligt stor kommer de minsta eller minst betydelsefulla objekten att generaliseras bort. (Eklundh 2000)

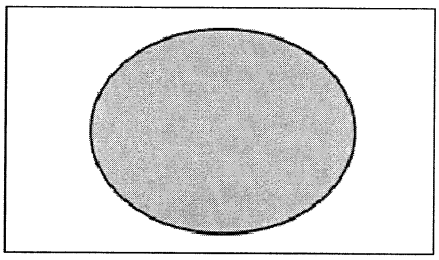


Figur 2.1. Raster som innehåller 16x14=224 celler fördelade på fyra klasser:
109 odlad mark-celler (ljusgrå)
14 vattendragsceller (mörkgrå)
15 vägceller (svarta)
86 bebyggt område-celler (vita).

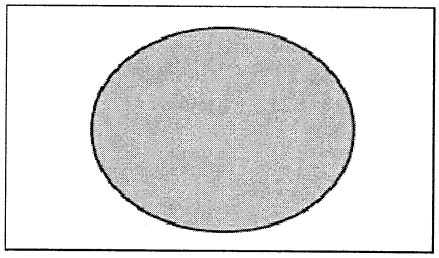


Figur 2.2. Visualisering av vektordata innehållande:
1 odlad mark-yta (ljusgrå)
1 vattendragslinje (mörkgrå)
1 väglinje (svart)
1 bebyggt område-yta (vit).

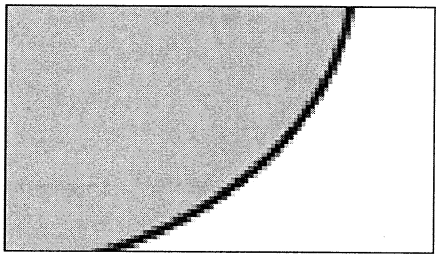
Vid användande av vektorgrafik representeras föremål som enskilda objekt i form av punkter, linjer och ytor vars läge och utsträckning anges med koordinater (Figur 2.2). Objekten renderas i realtid vilket innebär att datorn måste generera en grafisk bild (egentligen en rasterrepresentation) varje gång objekten ska visas. Eftersom bilder genereras i realtid utifrån koordinater medger vektorformat inzoomning utan förlust av upplösning (jämför Figur 2.4 och Figur 2.6 med Figur 2.3 och Figur 2.5). (Eklundh 2000)



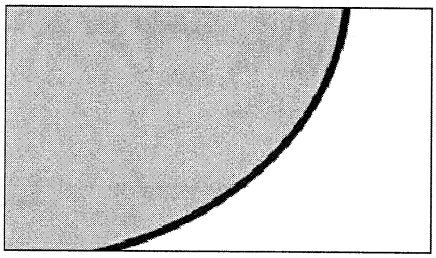
Figur 2.3. Rasterbild i GIF-format.



Figur 2.4. Vektorbild i SVG-format.



Figur 2.5. Nedre högra hörnet av Figur 2.3 förstorat.

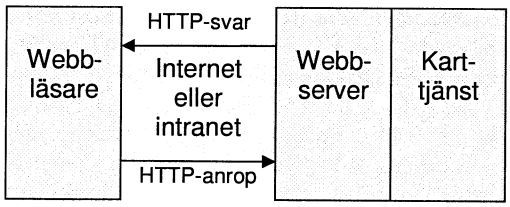


Figur 2.6. Nedre högra hörnet av Figur 2.4 förstorat.

För att visa vektorgrafik i webbläsare krävs oftast något insticksprogram som sköter renderingen. Eftersom rasterbilder inte behöver renderas är de enklare att visa och majoriteten av webbläsarna stödjer ett flertal rasterformat medan endast en webbläsare stödjer ett vektorformat med utbredd användning utan installation av insticksprogram (Mozilla 2003). Detta innebär att rasterformat, trots vektorgrafikens många fördelar, är vanligare i Internetbaserade karttjänster. (Köbben och Kraak 1999)

2.2 Klassificering av karttjänster

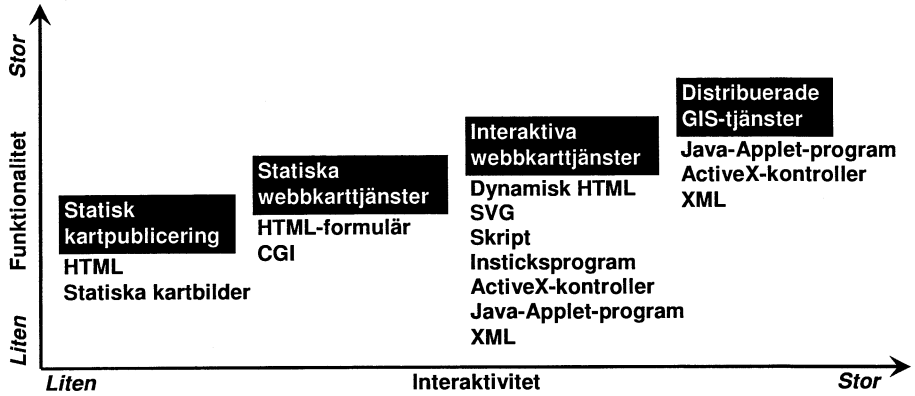
En webbserver (Figur 2.7) är ett program som kan användas för att göra filer och program på en dator tillgängliga från andra datorer. En webbläsare är ett program som kan användas för kommunikation med en webbserver och visning av filer tillgängliggjorda av en webbserver. En karttjänst kan förmedla kartdata till en användare av en webbläsare genom ett webbserverprogram.



Figur 2.7. Internetbaserad karttjänst.

En indelning kan göras i statiska och dynamiska karttjänster. Statiska kartor visar en representation av verkligheten vid ett tillfälle och dynamiska kartor kan till exempel visa ett händelseförlopp med hjälp av en animerad kartbild. Båda dessa grupper kan delas in i

kartor som möjliggör eller inte möjliggör interaktion med användaren. Interaktion i detta sammanhang kan till exempel innebära möjlighet att zooma, panorera, välja kartlager för visning samt klicka i kartbilden för att få fram attributdata till de geografiska objekt som visas. (Kraak och Brown 2000)



Figur 2.8. Olika tekniska lösningar möjliggör olika interaktivitet och funktionalitet. (omarbetad från Peng och Tsou 2003)

Statiska kartor utan interaktionsmöjlighet består vanligtvis av digitaliserade eller skannade kartdata som presenteras i rasterformat. Även vektorformat kan användas för kartor utan interaktion men vissa vektorformat medför automatiskt en viss möjlighet till interaktion eftersom programmen som används för visning i vissa fall har inbyggda funktioner för zoomning och panorering. (Kraak och Brown 2000)

Vektorformat lämpar sig väl även för mer avancerad interaktion eftersom användaren kan välja objekt i kartbilden för att till exempel redigera objektet eller efterfråga dess attributdata. I rasterbaserade tjänster måste tillverkaren av användargränssnittet själv definiera hur objekt i kartbilden ska tillgängliggöras.

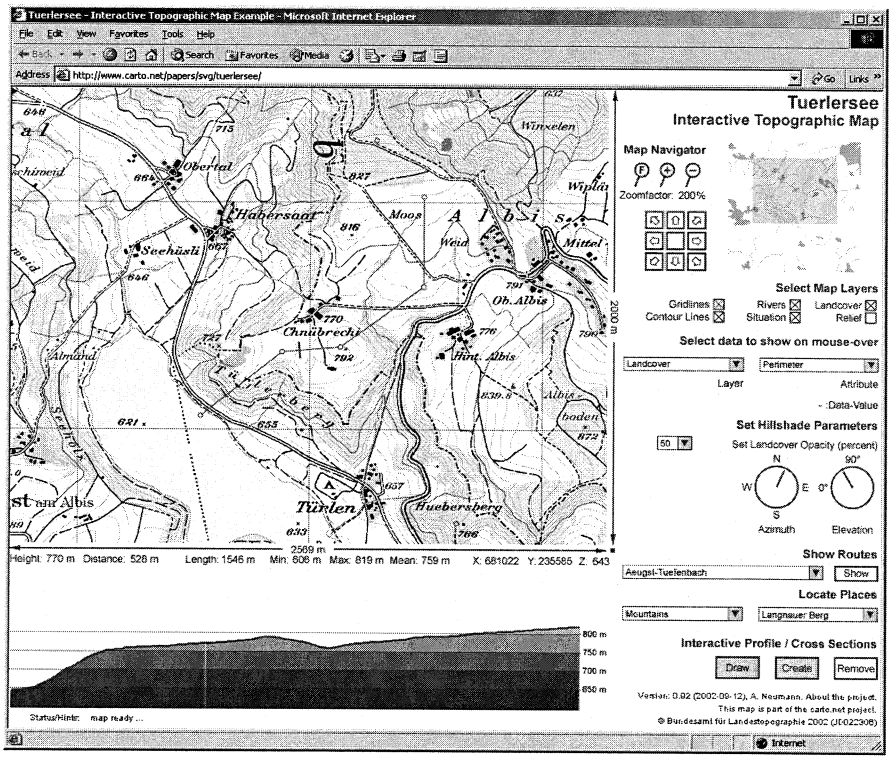
Vektordata är i allmänhet mindre utrymmeskrävande än rasterdata vilket innebär att överföringstiden för dessa tjänster kan vara kortare än för tjänster som skickar rasterdata. Tillgång till vektordata kan också innebära större möjligheter för analys hos klienten. Eftersom vektordata är skalbara kan det också vara lättare att använda denna typ av tjänster på små skärmar, till exempel i mobiltelefoner eller handdatorer. En annan fördel med vektordata är att det generellt sett är enklare att kombinera vektordata från flera källor samt att generalisera data i realtid (Lehto och Kilpeläinen 2000).

Små vektorbaserade karttjänster levererar ofta alla kartdata vid första kontakten med klienten, all zoomning och panorering sker därefter i de kardata som finns hos klienten. I rasterbaserade karttjänster sker nästan alltid överföring av kartdata efterhand som användaren zoomar eller panorerar. Karttjänster som sänder rasterdata till klienten har en fördel gentemot tjänster som sänder vektordata i att rasterdata inte innebär något större problem med avseende på upphovsrättsligt skydd. Tjänster som skickar vektordata till

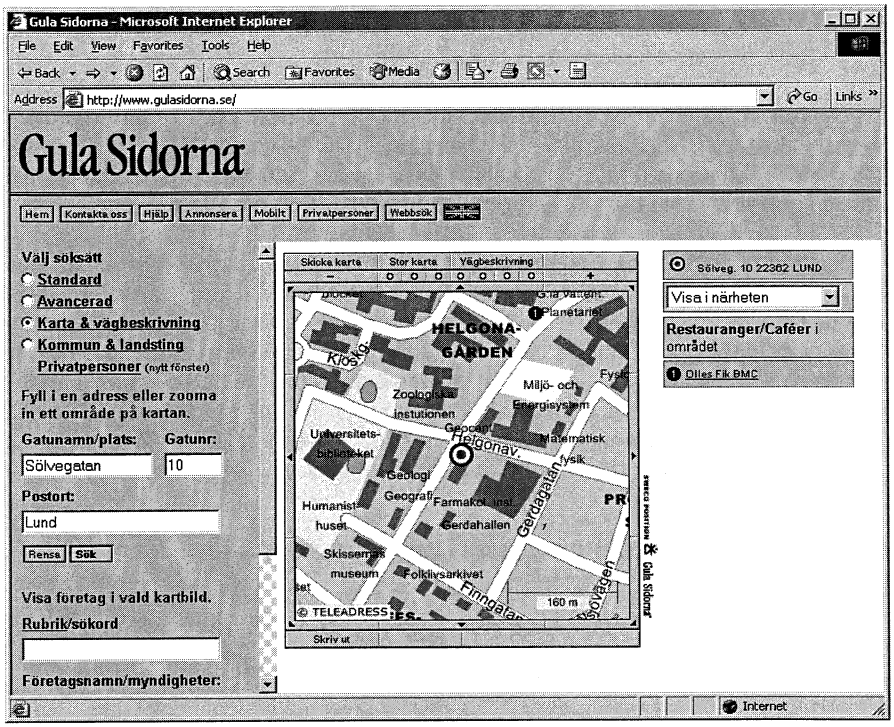
klienten skickar däremot koordinater för geografiska objekt och dessa koordinater kan lagras av klienten för användning utöver den avsedda.

Figur 2.9 visar ett exempel på en karttjänst med en bakgrundskarta i rasterformat och överlagrade skikt med vektorkartdata. Här skickas alla kartdata till klienten vid första kontakten med servern. I karttjänsten kan viss analys utföras. Till exempel kan användaren skapa en linje i kartbilden för vilken en höjdprofilkurva kan konstrueras.

Många karttjänster bygger på rasterkartbilder som framställts i förväg. För att in- och utzoomning ska möjliggöras måste en uppsättning kartbilder för varje önskat skalläge framställas i förväg. Detta eftersom rasterbilder ej är skalbara. När användaren väljer att zooma in visar tjänsten kartbilder ur den uppsättning som har en nivå större skala än de föregående. Det finns även karttjänster där ett serverprogram i realtid, från vektordata, skapar rasterbilder vilka sedan skickas till klienten. Även denna typ av tjänster kan ha steglös panorering och zoomning (Kraak och Brown 2000). I Figur 2.10 visas en rasterbaserad karttjänst som till exempel medger möjlighet att utföra adressökning och ruttplanering.



Figur 2.9. Exempel på karttjänst med viss interaktivitet. Nederst i figuren visas en höjdprofil som klientapplikationen har skapat på användarens begäran. Höjdprofilen löper längs den tunna linjen med början i sjön och slut nära byn Albis i kartans övre högra kvadrant. (carto:net 2004a)

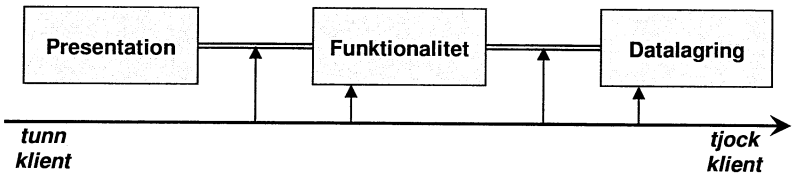


Figur 2.10. Exempel på karttjänst med viss interaktivitet. Här har en sökning efter adressen Sölvegatan 10 i Lund gjorts och sedan har en förfrågan efter restauranger och kaféer, i det område som den resulterande kartbilden visar, gjorts. (Gula Sidorna 2004)

2.3 Tjocka och tunna klienter

Ovan nämndes att tillgång till vektordata kan innebära större möjligheter till interaktivitet. Beroende på vad som levereras från karttjänsten krävs också olika mycket av klientprogrammet. Ett klientprogram som tar emot rasterdata kan vara en vanlig webbläsare men om vektordata mottas måste oftast något insticksprogram installeras för att hantera rendering av dessa vektordata till en kartbild. Ett klientprogram som handhar en stor del av arbetet med till exempel rendering av kartbild och analys av kartdata kallas en tjock klient, motsatsen är en tunn klient. Det finns olika grader i denna skala där de tunnaste klinterna endast kan utföra visualisering och de tjockaste klienterna utför presentation, rendering, analys och har del i hanteringen av datalagring (se Figur 2.11).

Karttjänsten i Figur 2.9 kan klassas som en tjock klient, eftersom presentation, rendering och all analys sker på klientsidan, och kan i Figur 2.11 representeras av den andra vertikala pilen från höger. Karttjänsten i Figur 2.10 representeras i Figur 2.11 av den första vertikala pilen från vänster vilket innebär att klientapplikationen här i stort sett bara utför presentation. (Peng och Tsou 2003)



Figur 2.11. De vertikala pilarna visar möjliga gränsdragningar mellan klientprogrammets och serverprogrammets uppgifter. (omarbetad från Peng och Tsou 2003)

3 Standarder för Internet

World Wide Web Consortium (W3C) har utformat många standarder som är centrala för utvecklingen av Internet. W3C bildades 1994 och har bland annat tagit fram de öppna standarderna HyperText Transfer Protocol (HTTP), eXstensible Markup Language (XML), HyperText Markup Language (HTML) och Scalable Vector Graphics (SVG).

Att standarderna är öppna innebär att de är offentliga och får implementeras av vem som helst. En fördel med öppna standarder är att datautbyte underlättas. Alla aktörer inom en bransch kan använda samma standardiserade överföringsformat vilket innebär att varje aktör endast måste utveckla en metod för konvertering av data och denna fungerar gentemot alla andra aktörer. Inom organisationen kan dessutom samma programvara som tidigare användas oavsett med vilken aktör samarbetet sker. (Harold och Means 2001)

Kapitlet inleds med beskrivningar av grundläggande standarder för möjliggörande av kommunikation via Internet och fortsätter sedan med ett antal standardformat för de data som skickas via Internet. Sist i kapitlet inriktas beskrivningen av standardformat mot grafikformat.

3.1 Uniform Resource Identifier (URI)

Uniform Resource Identifier (URI) är den standard som används för att ange adressen till en resurs på Internet. En resurs kan vara allt från en enkel webbsida till ett multimediaklipp eller ett inlägg i en diskussionsgrupp. URI är en övergripande term och det finns flera olika typer av URI:er (W3C 2001b). På svenska används ofta termen webbadress för URI.

Transmission Control Protocol (TCP) och Internet Protocol (IP) är två av protokollen som utgör TCP/IP som är en standard för hur kommunikation via bland annat Internet ska fungera. I standarden definieras bland annat hur datorer ska identifieras på Internet med hjälp av IP-adresser. En IP-adress består av fyra tal mellan 0 och 255 separerade med punkter (Hall 2000). Som exempel kan nämnas att 127.0.0.1 alltid är IP-adressen för den egna datorn och 130.235.128.122 är IP-adressen till Lunds Universitets webbserver.

För att göra det enklare att komma ihåg Internet-adresser finns ett översättningssystem där ett domännamn i textformat kan översättas till en IP-adress och vice versa. Den tjänst som sköter översättningen heter Domain Name System (alternativt Domain Name Service) (DNS) och fungerar i nätverk på ett sådant sätt att om tjänsten som får en förfrågning om översättning inte känner till namnet eller IP-adressen, skickas förfrågningen vidare till en annan DNS-server (Hall 2000). Till exempel översätts 127.0.0.1 till localhost och 130.235.128.122 till zeus.net.lu.se. Den adress som vanligen anges för att kontakta Lunds universitets webbserver är www.lu.se men flera DNS:er kan peka på en IP-adress och i det här fallet fungerar till exempel även DNS:en zeus.net.lu.se.

3.2 HyperText Transfer Protocol (HTTP)

W3C standardiserade 1999 HyperText Transfer Protocol (HTTP) som är ett av de protokoll som använder sig av URI för att beskriva var på Internet en tjänst eller dylikt

befinner sig och en adress angiven enligt HTTP-standard brukar kallas en URL (Uniform Resource Locator). HTTP är troligtvis det mest använda protokollet på Internet, några exempel på andra protokoll är File Transfer Protocol (FTP) som används för överföring av filer, Network News Transport Protocol (NNTP) som används för att distribuera meddelanden i diskussionsgrupper och Telnet som används för fjärrstyrning av datorer. (Musciano och Kennedy 2002)

```
http://www.opengis.org/specs
```
protokoll:värdnamn/begärd resurs

```
http://127.0.0.1:80/svg-exempel.svg
```
protokoll:värdnamn:portnummer/begärd resurs

```
../texter/text.txt
```
moderkatalog/katalog/begärd resurs

Figur 3.1. Exempel på olika webbadresser (URI).

Det första exemplet i Figur 3.1 använder sig av HTTP för att komma åt resursen "specs" hos värden "www.opengis.org". Resursen är en webbsida med information om OGC:s specifikationer. Det andra exemplet resulterar i visning av filen "svg-exempel.svg" från en webbserver på den egna datorn via port 80 som är standardport för kommunikation mellan webbläsare och webbservrar. I det första exemplet anges namnet för värden och i det andra anges IP-adressen. Det tredje exemplet visar en relativ URL till filen text.txt i katalogen "texter" som finns i den aktuella katalogens moderkatalog.

3.3 Common Gateway Interface (CGI)

Common Gateway Interface (CGI) är en standard för hur en klient kan förmedla parametervärden till ett program på en server. Standarden har utarbetats av National Center for Supercomputing Applications (NCSA) och vidareutveckling utförs från och med 1997 av Golux Enterprises. Webbservern som tar emot anropet, som oftast görs enligt http-standarden, vidarebefordrar parametrarna till något program som använder dem för att till exempel generera en webbsida eller en fil som webbservern kan inlemma i en webbsida. (NCSA 1998)

Det finns två olika metoder för att överföra parametervärden, HTTP-GET och HTTP-POST. När POST-metoden används skickas parametrarna i HTTP-meddelandet till serverprogrammet utan att användaren ser dem. När GET-metoden används skrivs parametrarna sist i webbadressen och skiljs från resursnamnet med hjälp av ett frågetecken (?). Avgränsare mellan CGI-parametrar är och-tecken (&) och mellan parameternamn och värde används likhetstecken (=). (NCSA 1998)

När användaren av "Kartsidan" i Figur 3.2 klickar i kartan görs följande HTTP-GET-anrop: `http://www.map.zz/kartsida.cgi?zoom=out&map.x=256&map.y=428` under förutsättning att användaren klickat på knappen för utzoomning och sedan klickat på positionen (x,y)=(256, 428) i kartbilden (koordinaterna är i enheten pixlar med origo i bildens övre vänstra hörn). Ett program på webbservern på den påhittade webbadressen tar emot koordinat- och zoom-parametrarna och skapar en ny kartbild som skickas till

webbservern som i sin tur infogar kartbilden i ett nytt HTML-dokument som skickas till användarens webbläsare. För att programmet på servern ska ta emot parametrarna ges HTML-dokumentet filändelsen *cgi* men när dokumentet har behandlats av serverprogrammet blir resultatet ett vanligt HTML-dokument som kan läsas av alla webbläsare som uppfyller HTML-standarden. (NCSA 1998)

3.4 HyperText Markup Language (HTML)

HyperText Markup Language (HTML) är en standard för uppmärkning av information för presentationsändamål. Hyperlänkning innebär att ett objekt (till exempel ett textavsnitt eller en bild) ges ett länkattribut vars värde är adressen till någon resurs med anknytning till objektet i fråga. När användaren aktiverar länken genom att till exempel klicka på objektet laddas det dokument som länkattributets adress pekar på. Ett HTML-dokument består av en textfil där innehållet strukturerats med märkord enligt HTML-standard.

Den första standardiserade versionen av HTML offentliggjordes 1995 och kom att heta HTML 2.0. HTML har haft en central roll i utvecklingen av Internet eftersom standarden legat till grund för mycket av informationsspridningen via Internet samt på grund av sitt relativa plattformsoberoende. (Musciano och Kennedy 2002)

3.4.1 HTML-baserade karttjänster

I HTML-dokument kan bilder infogas genom att en URL till bilden i fråga skrivs i HTML-koden. Det märkord som används för att infoga en bild är *img* (*image*). *img*-elementet har ett attribut som heter *src* (*source*) vars värde är webbadressen till bilden som ska visas.

Elementet *map* i HTML-standarden används för att definiera en samling hyperlänkar i form av polygoner. En sådan karta med hyperlänkar kan anges som attribut för ett *img*-element för att dela upp bilden i klickbara områden. De olika områdena kan till exempel vara nio lika stora rutor där varje ruta är länkad till en bild som visar rutans område förstorat. En motsvarande länkkarta kan appliceras på den nya bilden så att klickning i denna medför ytterligare förstoring. Ett serverprogram skapar en ny webbsida och infogar den nya kartbilden varje gång användaren gör något val (Musciano och Kennedy 2002). På Malmö stads hemsida (Malmö 2004) finns ett exempel på användning av denna metod.

Vidare finns element för mer detaljerad avläsning av användarens inmatningar. *Form*-elementet används för att avgränsa ett formulär för användarens inmatningar av olika typer av uppgifter. Inom ett formulär placeras *input*-element vilka är inmatningsfält av olika slag. *Input*-elementets *type*-attribut avgör vilken typ av inmatning som ska användas och ett möjligt värde på detta attribut är *image*. När *type*-attributet är *image* ska webbadressen till önskad bild anges som ett annat attribut till *input*-elementet. När användaren klickar i en kartbild som används som inmatningsfält skickar webbläsaren klickningens koordinater i bilden till serverprogrammet vilket sedan kan generera en ny kartbild som infogas i en ny webbsida som skickas till klienten (Musciano och Kennedy 2002). I Figur 3.2 visas HTML-kod som använder denna metod och på amerikanska

folkräkningsbyråns hemsida (U.S. Census Bureau 2004) finns en implementation av samma metod.

```
<html>
      <head>
            <title>Internetkartan</title>
      </head>
      <body>
            <h1><img src=" http://www.map.zz/bild.gif">Kartsidan</h1>
            Klicka i kartan för att zooma
            <form action="http://www.map.zz/kartsida.cgi"
            method="GET">
                  <input type="image" name="map"
                  src="http://www.kartsite.se/map_1.gif"><br>
                  <input type="radio" name="zoom" value="in"
                  checked>Zoom In<br>
                  <input type="radio" name="zoom"
                  value="out">ZoomOut<br>
            </form>
      </body>
</html>
```

Figur 3.2. En HTML-sida med en bild (bild.jpg), texten "Kartsidan" i rubrikformat, texten "Klicka i kartan för att zooma" och ett formulär. Formuläret består av en klickbar kartbild och två knappar som tillåter användaren att välja om klickning i kartan ska innebära in- eller utzoomning.

3.5 eXstensible Markup Language (XML)

eXstensible Markup Language (XML) har utvecklats av W3C sedan 1996 (W3C 2004) och det bygger på språket *Standard Generalized Markup Language* (SGML). XML är en standard som berättar hur data kan märkas upp och organiseras i en hierarkisk struktur. XML är dessutom ett metaspråk som möjliggör definitioner av andra språk. Några av de språk som skapats med hjälp av XML är *Mathematical Markup Language* (MathML) skapat av W3C för matematiska uttryck, Scalable Vector Graphics (SVG) skapat av W3C för vektorgraf och *Geography Markup Language* (GML) skapat av OGC för uppmärkning av geografiska data. Dessa efterföljande standarder kallas tillämpningar av XML. Ytterligare ett exempel på en XML-tillämpning är *XML Linking Language* (XLink) som används i XML-dokument för att skapa länkar mellan olika resurser med hjälp av URI:er.

Eftersom XML ska kunna användas för uppmärkning för mer än presentationsändamål skiljer man i XML, i motsats till till exempel HTML, på innehåll, struktur och presentation. För att tydliggöra detta har avsnittet getts i stort sett samma indelning; avsnitt 3.5.1 XML-dokument behandlar innehåll, avsnitt 3.5.2 DTD och XML-Schema behandlar struktur och avsnitt 3.5.4 eXstensible Stylesheet Language Transformations (XSLT) behandlar bland annat presentation. Hela avsnitt 3.5 bygger i huvudsak på Harolds och Means (2001) bok om XML.

3.5.1 XML-dokument

XML-filer lagras i textformat och är därför läsbara i en vanlig texteditor och de kan dessutom vara självförklarande. Innehållet märks och beskrivs av så kallade taggar som

inleder och avslutar varje element. Element kan innehålla andra element och en hierarkisk struktur kan på så sätt skapas. Taggar kan förutom ett obligatoriskt namn också innehålla attribut. (W3C 2004)

För att vara *välformulerat* enligt XML-specifikationen måste ett dokument bland annat uppfylla följande syntaktiska regler:

- Varje dokument måste ha en XML-deklaration.
- Varje dokument måste ha ett och endast ett rotelement.
- Varje element måste ha en sluttagg.
- Ett nytt element får inte påbörjas före det att föregående element på samma hierarkiska nivå avslutats.
- Varje attributvärde måste avgränsas med citationstecken.

```
<?xml version="1.0"?>

<personal arbetsgivare="volvo">
      <person>
            <namn>
                  Pelle
            </namn>
      </person>
      <person>
            <namn>
                  Lena
            </namn>
            <efternamn>
                  Nilsson
            </efternamn>
      </person>
</personal>
```

Figur 3.3. Exempel på XML-notation som kan användas för ett enkelt personalregister.

För att utveckla en ny tillämpning av XML krävs endast att regler skapas för vilka element som får ingå i ett dokument av den önskade tillämpningen samt vilka attribut elementen får ha och i vilken ordning elementen får förekomma. I Figur 3.3 ovan har data märkts upp enligt en tänkt standard för personalregister. De data som lagrats i exemplet är orden Pelle, Lena och Nilsson. Taggarna <personal>, <person>, <namn> och <efternamn> med tillhörande sluttaggar berättar vad orden har för innebörd och hur de är relaterade till varandra. Pelle, vars efternamn vi inte känner, och Lena Nilsson är två personer som har samma arbetsgivare. Personal-elementet har attributet "arbetsgivare" vilket i det här fallet har värdet "volvo".

3.5.2 DTD och XML-Schema

För att ange tillåtet innehåll i en XML-fil av en viss XML-tillämpning kan en *Document Type Definition* (DTD) skapas. En DTD anger bland annat vilka element som får eller måste förekomma, i vilken ordning de ska stå och vilka attribut de får ha. Tillämpningar av XML som har öppna specifikationer har ofta en DTD som finns tillgänglig på någon

webbserver. När till exempel ett SVG-dokument skapas anges i filens början sökvägen till den DTD som W3C har skapat för SVG-formatet (se Figur 3.4).

```
URL till den DTD som gäller för SVG version 1.0:
http://www.w3.org/TR/SVG/DTD/svg10.dtd
```

Figur 3.4. (W3C 1999).

```
<!DOCTYPE personal [

<!ELEMENT personal (person)>
<!ELEMENT person (namn, efternamn)>
<!ELEMENT namn (#PCDATA)>
<!ELEMENT efternamn (#PCDATA)>

<!ATTLIST personal arbetsgivare CDATA "saab">
]>
```

Figur 3.5. DTD för validering (kontroll av giltighet) av det enkla personalregistret i Figur 3.3.

I Figur 3.5 slås fast att en XML-fil av typen "personal" får innehålla ett personal-element, som i sin tur kan innehålla person-element, som i sin tur kan innehålla namn- och efternamn-element. Personal-element kan dessutom ha attributet "arbetsgivare" vilket ges standardvärdet "saab".

Användandet av DTD kan underlätta felsökning i XML-dokument eftersom program kan skapas för att jämföra innehållet i ett XML-dokument med de regler som finns i en DTD. Programmet kan också undersöka om syntaxen är välformulerad enligt XML-specifikationen. En XML-fil som till exempel tillämpar den DTD som finns för SVG korrekt kan kallas *giltig* enligt SVG.

Ett alternativ till DTD är XML Schema som är en standard definierad av W3C och den är i motsats till DTD en XML-tillämpning. Eftersom XML Schema är en XML-tillämpning är denna metod mer flexibel än användandet av DTD. XML-scheman kan till exempel transformeras och deras innehåll är åtkomligt genom XML DOM (se avsnitt 4.1 Document Object Model (DOM) nedan) (W3C 2001a). Ett exempel på XML-Schema ges i Figur 3.6.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

      <xsd:element name="personal" type="person"/>

      <xsd:complexType name="person">
            <xsd:sequence>
                  <xsd:element name="namn" type="xsd:string"/>
                  <xsd:element name="efternamn" type="xsd:string"/>
            </xsd:sequence>
            <xsd:attribute name="arbetsgivare" type="xsd:string"/>
      </xsd:complexType>

</xsd:schema>
```

Figur 3.6. XML Schema för validering (kontroll av giltighet) av det enkla personalregistret ovan.

3.5.3 Namespace

Det går att lagra information för flera XML-tillämpningar i samma dokument genom att använda så kallade *namespace*. *Namespace* används för att särskilja taggar och attribut som har samma namn i olika tillämpningar. Om data för till exempel tillämpningarna GML och MathML ska lagras i samma dokument bestäms ett prefix för någon av tillämpningarna eller för båda. GML-taggar och attribut kan ges prefixet "GML" och MathML-dito kan ges prefixet "Math". Deklarationen av ett prefix görs i elementet inom vilket prefixet ska användas eller högre upp i elementhierarkin med hjälp av en *Uniform Resource Identifier* (URI). Denna URI behöver inte nödvändigtvis leda till någon webbsida men den måste vara unik i förhållande till andra namespace-deklarationer i samma dokument. Även om den URI som används för att definiera ett namespace inte måste vara en giltig Internet-adress kan det vara bra om den pekar på en webbsida som innehåller till exempel tillämpningens specifikation. Detta kan underlätta validering av XML-dokument och det kan underlätta för användare som vill hitta information om tillämpningen.

```
<svg xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
      <image width="315" height="48"
      xlink:href="http://www.w3.org/Icons/w3c_main"/>
</svg>
```

Figur 3.7. SVG-dokument med användande av namespace.

På första raden i Figur 3.7 definieras i SVG-elementet standard-namespace och ett namespace för element och attribut med prefixet "xlink". Det namespace som anges som standard (xmlns=...) är SVG och eftersom det anges som standard-namespace tillhör alla element och attribut som inte ges något prefix detta namespace. På andra raden anges ett namespace för xlink (xmlns:xlink=...) som bland annat används för att ange sökvägar i XML. *Image*-elementet på rad fyra används i SVG för att infoga rasterbilder. Som attribut anges här i vilken storlek bilden ska presenteras i en SVG-läsare samt bildens sökväg i xlink-format.

3.5.4 eXstensible Stylesheet Language Transformations (XSLT)

eXstensible Stylesheet Language Transformations (XSLT) är en tillämpning av XML som används för att beskriva hur XML-dokument av ett slag kan transformeras till ett annat slag. XSLT härstammar från eXstensible Stylesheet Language (XSL) som delats in i XSLT och XSL Formatting Objects (XSL-FO) och därför används i många sammanhang förkortningen XSL.

Eftersom XSLT-dokument eller XSLT-stilmallar, är XML-dokument utför de inte någon transformation, de innehåller endast instruktioner för hur en XSLT-processor ska utföra en transformation och därigenom skapa ett nytt dokument utifrån ett XML-dokument. Resultatdokumentet kan till exempel vara ett nytt XML-dokument för en annan tillämpning eller för samma tillämpning, ett HTML-dokument eller ett rent textdokument. XSLT-dokument kan innehålla instruktioner för vilken text, andra tecken eller data från ursprungsdokumentet som ska skrivas i resultatdokumentet när XSLT-processorn påträffar ett visst element i ursprungsdokumentet.

XSLT lämpar sig väl för presentation av data från XML-dokument eftersom många webbläsare har inbyggda XSLT-processorer. Genom att ange sökvägen till önskat XSL-dokument i en XML-fil kan data från XML-filen enkelt presenteras i till exempel HTML-format med hjälp av den inbyggda XSLT-processor som finns i många webbläsare. Det finns även fristående program för transformering av XML-filer enligt XSL-dokument. Ett exempel på sådana är det Java-baserade Xalan från Apache (Apache 2004).

3.6 Grafikformat

Det finns två huvudtyper av grafikformat: Raster- och vektorgrafik. Nedan beskrivs ett antal standarder av respektive typ. Avsnittet om rastergrafik bygger i huvudsak på Gonzalez och Woods (2002) bok om digital bildbehandling.

3.6.1 Rastergrafik

Det finns ett flertal format för lagring av rastergrafik och de flesta innebär någon form av komprimering av lagrade data. En indelning i förstörande och ickeförstörande komprimering kan göras. Förstörande komprimering innebär att de data som komprimeras inte exakt kan återskapas vilket kan innebära att värdet i vissa celler inte återges korrekt. När beräkningar baserade på cellvärden utförs kan därför förstörande komprimering ge upphov till fel varför lagring av geografiska data bör ske med ickeförstörande eller ingen komprimering. Bland bildformaten nedan kan JPEG och JPEG2000 använda förstörande och ickeförstörande komprimering. Övriga format innebär ickeförstörande komprimering.

Graphic Interchange Format GIF

Vid lagring i GIF-format används maximalt en byte (åtta bitar) per pixel vilket innebär $2^8 = 256$ färger. Formatet kan använda en färg för genomskinlighet och flera bilder kan lagras i samma fil för skapande av animationer. Eftersom formatet använder ickeförstörande komprimering lämpar det sig väl för lagring av geografiska data i rasterformat. GIF är belagt med patent och användande av filformatet kräver därför i vissa fall licens från innehavaren av patentet (Unisys 2004).

Portable Network Graphics (PNG)
PNG-formatet kan liksom GIF hantera genomskinlighet men kan dessutom hantera olika grader av genomskinlighet. PNG använder upp till sex byte (48 bitar) för att representera en pixel vilket innebär 2^{48} = 281 474 976 710 656 färger. I likhet med GIF lämpar sig PNG väl för lagring av geografiska data i rasterformat.

Joint Photographic Experts Group (JPEG)
JPEG är det vardagliga namnet på standarden ISO/IEC IS 10918-1 eller ITU-T Recommendation T.81, JPEG är organisationen som utarbetat standarden. Vid JPEG-komprimering används multipler av cosinuskurvor för att approximera bilden som ska lagras. Bildkvaliteten är proportionell mot antalet använda multipler. Komprimeringen kan utföras ickeförstörande men få program implementerar ickeförstörande JPEG-lagring. Varje pixel representeras av maximalt tre byte (24 bitar) vilket innebär 2^{24} = 16 777 216 färger. Det stora antalet färger som kan användas och möjligheten till relativt små filstorlekar genom hög komprimering innebär att formatet lämpar sig väl för lagring av till exempel fotografier.

Joint Photographic Experts Group 2000 (JPEG2000)
Joint Photographic Experts Group har utarbetat en vidareutveckling av JPEG som kallas JPEG2000 och heter ISO/IEC 15444. Formatet använder multipler av delar av cosinusvågor för approximation av bilden. Formatet använder i likhet med JPEG tre byte per pixel och kan alltså hantera 16 777 216 färger. Även JPEG2000 lämpar sig väl för lagring av fotografier.

3.6.2 Vektorgrafik

Gemensamt för de två formaten för lagring av vektorgrafik som tas upp här är att de medger lagring av vektordata, ickespatiella attribut och presentationsattribut i samma fil.

Web Computer Graphics Metafile (WebCGM)
Computer Graphics Metafile (CGM) är en specifikation från International Organization for Standardization (ISO) från 1987. WebCGM är en tillämpning av CGM med inriktning mot användning på Internet. Tillämpningen har utarbetats av W3C i samarbete med CGM Open Consortium. WebCGM-filer lagras i binärformat vilket innebär att filerna blir relativt små men ej läsbara för människor. Formatet innehåller inget stöd för styrning av presentationsstilar. (W3C)

Scalable Vector Graphics (SVG)
Scalable Vector Graphics (SVG) är en XML-tillämpning som är speciellt lämpad för presentation av vektordata på Internet. Se mer därom i avsnitt 3.7 Scalable Vector Graphics (SVG).

Shock Wave Flash (SWF)

Shock Wave Flash är ett vektorgrafikformat speciellt utvecklat för användning på webben. Företaget Macromedia har utvecklat formatet och har offentliggjort specifikationen. Formatet har utbredd användning främst inom reklam och spel på Internet samt formgivning av webbsidor. (OpenSWF.org 2004)

3.7 Scalable Vector Graphics (SVG)

Detta avsnitt bygger huvudsakligen på World Wide Web Consortiums (2003b) specifikation av Scalable Vector Graphics (SVG) version 1.1.

XML lämpar sig som ovan sagt för att strukturera information inom de mest skilda områden och ett av dessa områden är vektorgrafik. World Wide Web Consortium offentliggjorde i februari 1999 det första utkastet till standarden SVG (W3C 2003c). Det fanns redan ett antal vektorgrafikformat men inget som byggde på en öppen standard och som var plattformsoberoende och anpassat för användande på Internet. Eftersom SVG är en tillämpning av XML följer SVG alla de syntaktiska regler som gäller för XML. SVG-specifikationerna och även de dokumenttypsdefinitioner som gäller för de olika versionerna av SVG visar vilka element som får användas i ett SVG-dokument och vilka attribut elementen får ha.

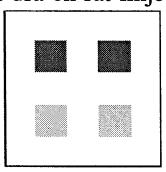
SVG kan användas för att skapa typiska vektorgrafiska element som punkter, linjer och ytor. Linjer kan skapas i form av räta linjer, polylinjer, elliptiska bågar samt kvadratiska och kubiska bezierkurvor. Ytor går att framställa som trianglar, rektanglar, polygoner samt cirklar och ellipser. SVG hanterar även text och det går också att infoga rasterbilder i SVG-dokument.

3.7.1 Exempel på SVG-dokument

I Figur 3.9 visas ett SVG-dokument, nedan följer en beskrivning i punktform av detta dokument och i Figur 3.8 visas resultatet av en rendering av dokumentet.

- På översta raden deklareras att dokumentet är ett XML-dokument
- Andra raden innehåller information om vilken tillämpning av XML det rör sig om.
- På tredje raden anges sökvägen till aktuell dokumenttypsdefinition.
- Därefter deklareras själva SVG-dokumentet genom rotelementet *svg*. Inom rotelementet anges också SVG-elementets bredd och höjd i enheten pixlar samt namespace för SVG.
- Det första elementet (*desc*) inom SVG-delen av dokumentet markerar en kommentar som kan användas för att beskriva innehållet i SVG-dokumentet. *desc*-elementet markerar en kommentar enligt SVG-syntax och elementet ignoreras därför vid rendering men den kan tolkas av både SVG-läsare och rena XML-läsare.
- Det andra elementet (*g*) är ett gruppelement som i detta fall innehåller eller grupperar två rektangelelement. Gruppelementet har ett stilattribut som innebär att de båda rektanglarna ges färgen *DimGray*. Gruppelementet har även ett *id*-attribut.
- De båda rektangelelementen har attribut som anger deras position (*x* respektive *y*) och utbredning (*width* respektive *height*) i antal pixlar.
- Efter gruppen med de två mörkgrå rektanglarna följer en grupp med två rektanglar i färgen *LightGrey*.

- Raden som börjar med "<!-- " är en kommentar enligt XML-syntax och den hoppas därför över av såväl XML- som SVG-läsare.
- Det sista i bilden som renderas är en rektangel utan fyllningsfärg men med svart (*black*) kantlinje som ram kring hela SVG-bilden. Rektangeln skapas med hjälp av ett *path*-element vilket ofta används i kartografiska tillämpningar bland annat eftersom syntaxen tar lite utrymme i anspråk. *Path*-elementets *d*-attribut anger dess geometriska läge och utsträckning. *M*-attribut (*move to*) inom *d*-attributet anger att en tänkt penna ska flyttas till positionen som anges med det efterföljande koordinatparet, *l*-attribut (*line to*) anger att den tänkta pennan ska dra en rät linje till efterföljande position och eftersom *l* här skrivs som gemen ska efterföljande koordinatpar tolkas som relativt föregående koordinatpar. *Z*-attributet anger att pennan ska dra en rät linje tillbaka till den position som angavs av det första koordinatparet. SVG-dokument renderas uppifrån och ner vilket innebär att det element som anges först i dokumentet ritas ut först och därmed underst. Efterföljande element ritas ut över de föregående vilket innebär att om *path*-elementets *fill*-attribut inte hade satts till *none* hade de fyra mindre kvadraterna inte varit synliga.



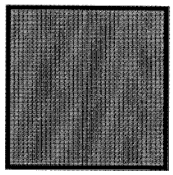
Figur 3.8. Rendering av SVG-exemplet i Figur 3.9.

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg width="100px" height="100px" version="1.1"
    xmlns="http://www.w3.org/2000/svg">
    <desc>Två grupper med två rektanglar (kvadrater) i varje
    </desc>
    <g id="group1" fill="DimGray">
        <rect x="20" y="20" width="20" height="20"/>
        <rect x="60" y="20" width="20" height="20"/>
    </g>
    <g id="group2" fill="LightGrey">
        <rect x="20" y="60" width="20" height="20"/>
        <rect x="60" y="60" width="20" height="20"/>
    </g>
    <!--Markera ytterkanten med en kvadrat-->
    <path d="M1 1 l97 0 l0 97 l-97 0 Z" fill="none" stroke="Black"
    stroke-width="1"/>
</svg>
```

Figur 3.9. Exempel på SVG-syntax.

3.7.2 Ändring av SVG-objekts utseende

Det finns flera olika sätt att ändra utseendet på elementen i ett SVG-dokument. Exempel på stilattribut som ett element kan utrustas med är färg, kantlinjetjocklek för ytor, typsnitt för text, och mönster för streckade linjer. Stilinställningar kan anges som separata attribut eller i en stilmall enligt *Cascading Style Sheets* (CSS). Stilmallsinformationen kan placeras i en extern fil, i ett *style*-element i SVG-dokumentet eller i ett *style*-attribut till ett element. I Figur 3.11, Figur 3.12, Figur 3.13 och Figur 3.14 visas olika sätt att ange stilparametrar i SVG-dokument och i Figur 3.10 samtliga metoders identiska resultat.



Figur 3.10. Resultatet av SVG-koden i Figur 3.11 - Figur 3.14.

Om stilinformationen placeras i en extern fil kan denna CSS-fil användas för flera SVG-dokument och stilattributen för samtliga element i flera filer kan ändras genom byte av eller ändring i CSS-filen. Om en intern stilmall används finns all information för presentationen samlad i en fil. Om liknande element i samma SVG-dokument ska ha olika stilar är det enklare att använda *style*-attributet eller separata stilattribut i varje element. *Style*-attributet är lämpligt att använda om attributen ska ändras med hjälp av skript och DOM (se 4.2 ECMAScript och 4.1 Document Object Model (DOM)) eftersom alla attribut kan ändras med hjälp av en rad i skriptkoden istället för med en rad för varje attribut vilket annars krävs.

```
rect {
      fill: Gray;
      stroke: Black;
      stroke-width: 3
}
```

```
<?xml version="1.0"?>
<?xml-stylesheet href="stilmall.css" type="text/css"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
      "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
      <rect width="100" height="100"/>
</svg>
```

Figur 3.11. I den övre rutan visas den externa stilmallen "stilmall.css" som används av SVG-dokumentet i den nedre rutan.

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
      "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
      <defs>
            <style type="text/css"><![CDATA[
            rect {
                  fill: Gray;
                  stroke: Black;
                  stroke-width: 3
            }
            ]]></style>
      </defs>
      <rect width="100" height="100"/>
</svg>
```

Figur 3.12. Användning av en intern stilmall.

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
      "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
      <rect width="100" height="100" style="fill:Gray; stroke:Black;
      stroke-width:3"/>
</svg>
```

Figur 3.13. Användande av *style*-attributet.

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
      "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
      <rect width="100" height="100" fill="Gray" stroke="Black"
      stroke-width="3"/>
</svg>
```

Figur 3.14. Användande av separata stilattribut.

3.7.3 SVG och filstorlek

I många sammanhang talas det om att SVG-formatet kräver för mycket lagringsutrymme för att lämpa sig för kartografiska tillämpningar via Internet. Eftersom SVG är ett XML-format lagras SVG-data med textkodning vilket medför större filstorlek än om lagringen hade skett med binärkodning som är fallet för många andra grafiska lagringsformat. Det har utarbetats flera lösningar på problemet att minska SVG-filers storlek och några av dessa lösningar går att kombinera.

Komprimering

I SVG-specifikationen nämns att HTTP 1.1 tillåter sändning av komprimerade data mellan server och klient. Detta kan utnyttjas för att sända komprimerade SVG-filer (W3C 2003b). Både Corel och Adobe har tagit fasta på detta och deras SVG-insticksprogram för webbläsare kan förutom vanliga SVG-filer även hantera komprimerad SVG eller

SVGZ. Enligt Adobe (2003) har en komprimerad SVG-fil tjugo till femtio procent av storleken av motsvarande okomprimerade SVG-fil.

Uteslutande av attributdata
Genom att använda ett *namespace* för attributdata kan dessa lagras i SVG-elementen i en SVG-fil. Om varje grafiskt element ges sitt geografiska objekts samtliga attributdata kan SVG-filerna bli väldigt stora. Genom att endast ge SVG-elementen ett id-attribut vars motsvarighet finns hos motsvarande geografiska objekt kan övriga attributdata hämtas från en server efterhand som de behövs.

Relativa koordinater
Vid användande av plana koordinatsystem utgör ofta värdena på koordinaterna väldigt stora tal. Genom att använda *path*-elementet (se Figur 3.9) och använda gemener för skiljetecknen mellan koordinatpar och därmed relativa koordinater kan filstorleken minskas betydligt. Givetvis måste det första koordinatparet anges med absoluta koordinater.

Avrundning
Avsevärd storleksminskning kan uppnås om koordinaterna i SVG-filer ges lämplig noggrannhet. Om till exempel en kartbild med storleken 500 gånger 500 pixlar som ska visa en hel kontinent efterfrågas, är det i stort sett meningslöst att ge koordinaterna en noggrannhet på under 100 m. Pikkuniemi (2002) har visat att en minskning från sju till fem värdesiffror för relativa koordinater i en SVG-fil kan ge en storleksminskning på uppemot 40 procent.

Generalisering
För att ytterligare begränsa SVG-filernas storlek finns det mycket att vinna inom generaliseringsområdet. Om till exempel en kartbild, med storleken 200 gånger 700 pixlar, som visar hela Sverige efterfrågas är det orimligt att visa alla vägar och alla byggnader i den bilden. Samtidigt kan inte skalbarheten hos SVG-formatet utnyttjas om klienten inte laddar ner data med högre noggrannhet än vad som krävs för den initiala skalan, klienten måste istället begära nya kartdata från servern vid till exempel inzoomning.

Det finns inom Internet-baserade karttjänster i stort sett två olika tekniker för skalning i kartdata. Den vanligaste är att nya kartdata hämtas från en server vid varje skalförändring. En annan metod som ofta används vid mindre dataset i SVG-format är att hämta hela datasetet vid första kontakten med servern och därefter utföra all omskalning i klientprogrammet. När stora geografiska områden ska visas i stora skalintervall är den senare metoden inte lämplig eftersom den innebär att väldigt stora datamängder måste överföras till klienten. Vidare krävs lagringsutrymme och bra beräkningskapacitet hos klienten.

3.7.4 Kartografiska tillämpningar

Kartografiska tillämpningar nämns ofta som ett bra exempel på vad SVG kan vara användbart till. Det finns många karttjänster som använder sig av SVG för att skapa kartbilder men många av dem är endast skapade som exempel på vad som kan göras med SVG (se till exempel carto:net 2004b).

SVG-specifikationen beskriver ett antal funktioner som är användbara inom kartografiska tillämpningar. SVG-element kan animeras vilket kan utnyttjas för att ge ökad läsbarhet i kartbilder och filter kan användas för att till exempel göra tonade fyllningar eller skuggningseffekter. SVG hanterar också olika grader av genomskinlighet vilket kan vara användbart vid placering av informationsrutor i kartbilder.

Något som underlättar hantering av olika koordinatsystem i SVG är transformationsfunktionerna. Transformation kan ske medelst skalning, translation, rotation och skevning eller med hjälp av transformationsmatris. Skevning av x-axeln innebär att det nya koordinatsystemets x-axel är vriden kring origo, vilket i kombination med olika skalning av x- och y-axel samt bristande rätvinklighet innebär affin transformation (Eklundh 2000). Koordinatsystemet i SVG och de flesta andra datorgrafikformat, har origo i fönstrets övre vänstra hörn, y-axeln är positiv nedåt och x-axeln är positiv åt höger.

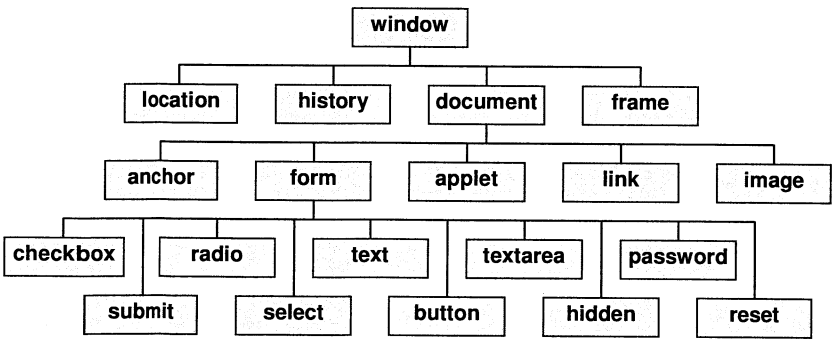
Ett gruppelement kan användas för att gruppera andra element och flera gruppelement kan kombineras för att skapa grupper i grupper. Animeringar, stilattribut, transformationer, med mera, kan appliceras på enskilda element eller på grupper. På så sätt kan till exempel hierarkiska gruppanimationer skapas för att illustrera sammansatta rörelser.

4 Programmeringstekniker

4.1 Document Object Model (DOM)

För att öka möjligheterna för interaktivitet och användbarheten av skriptspråk har W3C definierat Document Object Model (DOM). DOM är ett gränssnitt mellan skriptspråk och dokument av olika slag och det används för att göra dokument dynamiska. Med hjälp av DOM kan ett skript nå och ändra alla delar av ett dokument. Detta innebär till exempel att användarens interaktion med en karttjänst kan förändra utseendet på kartbilden utan att nya kartdata måste hämtas från en webbserver. (W3C 1998), (W3C 2000), (W3C 2003a)

Figur 4.1 visar DOM-strukturen för ett HTML-dokument i en webbläsare. I Figur 4.2 används DOM för att ändra ett attribut hos ett element i en webbsida. På sjätte raden i figuren står följande centrala uttryck: "`window.document.myForm.myText.value`". Läsaren kan i Figur 4.1 följa DOM-anropet från *window* överst i figuren genom hierarkin till *text* i det understa lagret. Anropet utgår från *window*-objektet som representerar den yta i webbläsaren där HTML-dokumentet ritas ut. *Document*-objektet representerar HTML-dokumentet, *form*-objektet representerar *myForm* och *text*-objektet representerar *myText*. *Window*- och *document*-objekten finns bara i en uppsättning för varje webbläsare och kan därför refereras till med sina typnamn. *Form*- och *text*-objekten kan finnas i flera uppsättningar på en webbsida och de refereras därför till med hjälp av de namn de tilldelats av skaparen av HTML-dokumentet. Anropet avslutas med *value* som representerar värdeattributet för *myText*. I Figur 4.3 och Figur 4.4 visas resultatet av DOM-anropet och HTML- och skript-koden i Figur 4.2 (mer om skript i avsnitt 4.2 ECMAScript).



Figur 4.1. DOM-struktur för ett HTML-dokument. (omarbetad från Ek och Eriksson 2001)

```
<html>
    <head>
        <title>test av HTML-DOM</title>
        <script type="text/ecmascript"><!--
        function changeText(){
            window.document.myForm.myText.value="Du klickade i
            inmatningsfältet";
        }
        //-->
        </script>
    </head>
    <body>
        <form name="myForm">
            <input type="text" name="myText" value="Klicka här!"
            onclick="changeText()" size="30">
        </form>
    </body>
</html>
```

Figur 4.2. Exempel på användning av DOM hos ett HTML-dokument.

Figur 4.3. Resultatet av HTML-koden i Figur 4.2.

Figur 4.4. Resultatet när användaren klickat I inmatningsfältet i Figur 4.3.

4.2 ECMAScript

För att göra webbsidor interaktiva kan program som kan tolkas direkt av webbläsaren inlemmas i till exempel HTML-kod eller SVG-kod. De vanligaste skriptspråken på Internet är olika versioner av ECMAScript. ECMAScript är en standardiserad version av språket JavaScript som skapades av Netscape. De flesta webbläsare har en inbyggd skripttolk som kan tolka ECMAScript-kod. (ECMA 1996)

Adobe SVG Viewer såväl som Corel SVG Viewer har en inbyggd skripttolk som implementerar ECMAScript. Vid användande av Adobes insticksprogram kan ett kommando i SVG-filen ange om insticksprogrammets egen skripttolk eller webbläsarens skripttolk ska användas. Om den inbyggda skripttolken används fungerar skript inlemmade i SVG-dokument på samma sätt oberoende av vilken webbläsare som används.

Namnet JavaScript kommer av språkets syntaktiska likheter med programspråket Java. Liksom Java är ECMAScript objektorienterat men i motsats till Java tillämpar ECMAScript lös typhantering vilket innebär att variabler inte behöver vara av någon

speciell typ. En variabel som ursprungligen pekar på ett tal kan när som helst fås att peka på till exempel en textsträng istället. Ännu en olikhet vid jämförelse med Java är att programkoden tolkas direkt av en skripttolk utan kompilering. ECMAScript-kod lagras därför i ASCII-format och går därmed att läsa i ett vanligt textredigeringsprogram. (Jaworski 2000)

4.2.1 ECMAScript och rasterkartor

Det finns ett antal tillvägagångssätt för att skapa interaktiva karttjänster och många av dessa bygger på användande av något skriptspråk på klientsidan. Det finns specialutvecklade gränssnitt för visning av kartdata som bland annat använder sig av ECMAScript eller liknande skriptspråk och rasterkartdata. Ett exempel på ett sådant finns i karttjänsten på Gula Sidorna (Gula Sidorna 2004) (se Figur 2.10) där JavaScript-kod tolkar användarens musklickningar i kartbilden och beställer nya kartbilder från en webbserver när användaren till exempel valt att zooma eller panorera. Beställningen sker med hjälp av Common Gateway Interface-parametrar vilket står att läsa om i avsnitt 3.3 Common Gateway Interface (CGI) ovan.

4.2.2 ECMAScript, DOM och vektorkartor

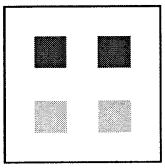
I SVG-dokumentet i Figur 4.5 används ECMAScript-filen i Figur 4.6 för att förändra utseendet på element i SVG-dokumentet. Direkt efter rotelementet i SVG-dokumentet anges adressen till den ECMAScript-fil som ska användas och i rotelementet anges att när muspekaren förs över (*onmouseover*) den renderade SVG-bilden ska funktionen *mouseOverRects* användas med argumentet *evt*. *Evt* (*event*) är namnet på det händelseobjekt som genereras vid användarens interaktion och detta objekt innehåller bland annat information om vilket objekt användaren interagerat med och vid vilka koordinater muspekaren befann sig då händelsen inträffade. I skriptfilen som hänvisas till i SVG-filen finns funktionen *mouseOverRects(evt)*. På första raden i funktionen sätts variabeln *targetId* till värdet för *id*-attributet för det objekt i SVG-dokumentet som muspekaren befann sig över när händelsen inträffade. På nästa rad inleds en villkorssats som uppfylls om *id*-attributet, för det objekt som pekaren hölls över, har värdet *"rect1"* eller *"rect2"*. Om villkoret uppfylls ska *fill*-attributet (färgen som används för det inre av ett element) för gruppen med *id*-attribut *"group2"* sättas till *"Black"*. Figur 4.7 och Figur 4.8 visar SVG-dokumentet renderat före och efter det att användaren fört muspekaren över *rect1* eller *rect2*.

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg width="100px" height="100px" version="1.1"
    xmlns="http://www.w3.org/2000/svg"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    onmouseover="mouseOverRects(evt)">

    <script type="text/ecmascript" xlink:href="skriptfil.es"/>

    <desc>Två grupper med två rektanglar (kvadrater) i varje
    </desc>
    <g id="group1" fill="DimGray">
        <rect id="rect1" x="20" y="20" width="20" height="20"/>
        <rect id="rect2" x="60" y="20" width="20" height="20"/>
    </g>
    <g id="group2" fill="LightGrey">
        <rect id="rect3" x="20" y="60" width="20" height="20"/>
        <rect id="rect4" x="60" y="60" width="20" height="20"/>
    </g>
    <!--Markera ytterkanten med en kvadrat-->
    <path d="M1 1L98 1L98 98L1 98Z" fill="none"
    stroke="Black" stroke-width="1"/>
</svg>
```

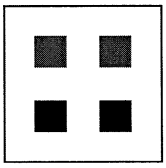
Figur 4.5. Exempel på SVG-syntax.

```
function mouseOverRects(evt){
    var targetId=evt.target.getAttribute("id");
    if(targetId=="rect1" || targetId=="rect2"){
        document.getElementById("group2").setAttribute(
        "fill", "Black");
    }
}
```

Figur 4.6. Exempel på ECMA-skript- och DOM-syntax.



Figur 4.7. SVG-dokumentet före det att användaren fört muspekaren över någon av de mörkgrå kvadraterna.



Figur 4.8. SVG-dokumentet efter det att användaren fört muspekaren över någon av de mörkgrå kvadraterna.

4.3 Java

Java är en öppen standard som definierar ett högnivåspråk för objektorienterad programmering. Java-program kräver i motsats till JavaScript att en kompilator kompilerar programkoden innan programmet kan exekveras. När programkoden kompileras skapas bytekod som kan läsas av Java-programtolkare på datorer med olika

operativsystem. Eftersom Java är plattformsoberoende och relativt säkert har det blivit populärt för utveckling av webbtjänster. (Gosling m fl 2000)

Många webbläsare har ett insticksprogram som fungerar som Java-programtolk och som kan exekvera en viss typ av Java-program som kallas *Java-Applet*. *Java-Applet*-program kan inte utnyttja samtliga delar av Java-specifikationen eftersom detta skulle kräva en mycket större programtolkare i insticksprogrammet (Holm 1999) men de kan göras mer kraftfulla än programsekvenser skrivna med hjälp av skriptspråk.

4.3.1 Java-Applet-program

Ett exempel på ett *Java-Applet*-program i en kartografisk tillämpning finns hos map24 (map24 2004). *Java-Applet*-programmet på denna webbsida hämtar vektordata hos en server, renderar dessa data och ritar ut en kartbild. Genom att hämta vektordata på en kartserver och rendera dessa data i realtid kan Java-applikationen på den här webbsidan utföra steglös zoomning och panorering.

- Programmet hämtar vektordata och renderar den initiala kartbilden
- När användaren väljer att zooma in skalar programmet om de kartdata som laddades ner för den initiala kartbilden och renderar en ny bild.
- Detta upprepas tills den större skalan kräver en högre detaljrikedom.
- Programmet efterfrågar nya kartdata men fortsätter att skala om och rendera de kartdata som finns till hands medan ytterligare data hämtas från servern.
- Nya geografiska objekt adderas till kartbilden efterhand som data tas emot från servern.

5 Standarder från Open GIS Consortium

För att underlätta samarbete inom geografisk informationsbehandling har ett konsortium av företag, universitet och myndigheter inom GIS-branschen bildats. Sammanslutningen kallas Open GIS Consortium (OGC) och den arbetar bland annat med att ta fram öppna standarder för geografisk informationsbehandling. Stora delar av Open GIS Consortiums arbete har anknytning till och bygger på standarder från andra organisationer, bland andra W3C.

OGC har skapat en serie standarder för distribution av geografiska data och kartdata på webben. Dessa standarder är konstruerade för att fungera i samverkan med varandra. Till exempel kan en Web Feature Service-server (se avsnitt 5.3 Web Feature Service (WFS)) användas för att leverera geografiska data till en Web Map Service-tjänst (se avsnitt 5.2 Web Map Service (WMS)) och eftersom standarderna är öppna kan flera olika aktörer samverka genom att använda varandras tjänster. I detta kapitel gås några av dessa standarder igenom. Tonvikten ligger på Web Map Service-standarden.

5.1 Geography Markup Language (GML)

För att standardisera utbyte av geografiska data har OGC skapat en öppen standard som heter Geography Markup Language (GML). GML är en tillämpning av XML och formatet används främst som ett överföringsformat för geografiska data och i viss mån även för lagring. Specifikationen för GML definierar regler för hur XML-scheman för GML-format ska skrivas. Detta innebär att en organisation som vill använda GML kan skriva sitt eget XML-Schema för att definiera sin egen tillämpning av GML. Nedan följer ett exempel som visar hur GML-notation kan se ut och som där visas kan såväl spatiella som ickespatiella attribut till geografiska objekt lagras i en GML-fil. (Open GIS Consortiums 2003)

Figur 5.1 visar ett exempel på GML-syntax för beskrivning av en skolbyggnad. Beskrivningen inleds med en klartextbeskrivning av objektet som talar om att det handlar om Norreportskolan i Ystad. Därefter följer de ickespatiella egenskaperna *NumFloors* och *NumStudents*. Sist beskrivs byggnadens geografiska utsträckning med hjälp av en polygon i form av ett *LineString*-element vars koordinater anges i *CData*-elementet.

```
<Feature fid="329" featureType="school">
    <Description>Norreportskolan Ystad</Description>
    <Property name="NumFloors" type="Integer" value="3"/>
    <Property name="NumStudents" type="Integer" value="533"/>
    <Polygon name="extent" srsName="epsg:3021">
        <LineString name="extent" srsName="epsg:3021">
            <CData>
                1388690.40,6065932.06 1388722.39,6065945.90
                1388710.52,6065992.21 1388690.42,6065933.55
                1388721.40,6065910.32 1388760.52,6065922.26
                1388690.40,6065932.06
            </CData>
        </LineString>
    </Polygon>
</Feature>
```

Figur 5.1. Exempel på GML-syntax.

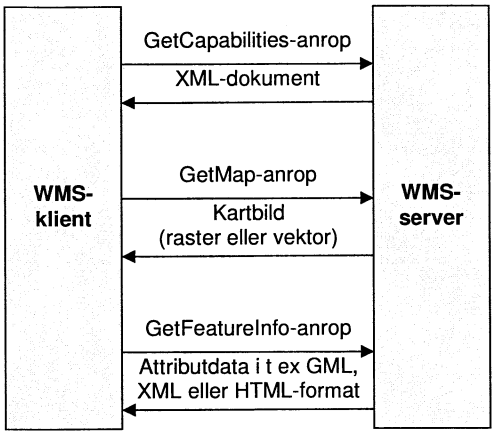
Ett svenskt exempel på användande av GML är Målbild 2000 som är ett samverkansprojekt mellan Lantmäteriet och Kommunförbundet. Man har även inlett ett uppföljningsprojekt som kallas Införande av Målbild 2000. En del av det senare har målsättningen att införa ett GML-baserat gränssnitt kallat Gränssnitt 2000 för överföring av geografiska data mellan de kommunala lantmäterimyndigheterna och Lantmäteriet. (Lantmäteriet 2004a)

5.2 Web Map Service (WMS)

Detta avsnitt bygger i huvudsak på Open GIS Consortiums (2002b) specifikation av Web Map Service (WMS) 1.1.1.

Open GIS Consortium har skapat en specifikation som definierar hur kartdata ska efterfrågas och vad som ska returneras vid en förfrågan efter kartdata via Internet. Standarden heter Web Map Service och är en i raden av flera standarder som utarbetats av OGC för hantering av geografiska data via Internet. Kommunikationen mellan klient och server sker med hjälp av HTTP-gränssnittet och för att ange variabler som krävs för serverns behandling av anropen används CGI-parametrar.

Tanken är att klienten ska göra förfrågningar med hjälp av de tre funktioner specifikationen beskriver i följande ordning: *GetCapabilities*, *GetMap* och *GetFeatureInfo*. Svaret på en *GetCapabilities*-förfrågan berättar för klienten vilka kartbilder som kan efterfrågas med hjälp av *GetMap*-anrop. Genom att referera till den av *GetMap*-förfrågningen resluterande kartbilden och koordinater i denna kan attributdata för föremål i kartbilden efterfrågas med hjälp av *GetFeatureInfo*-funktionen (se Figur 5.2).



Figur 5.2. Flödesschema för kommunikation mellan WMS-klient och WMS-server.

En WMS-server som implementerar *GetCapabilities*- och *GetMap*-funktionerna kallas *Basic* WMS, en WMS-server som dessutom implementerar *GetFeatureInfo*-funktionen kallas *Queryable* WMS. Ytterligare en typ av WMS-serverimplementation är *Cascading* WMS vilket innebär att den fungerar som en WMS-klient gentemot andra WMS-servrar och som en WMS-server gentemot WMS-klienter. Därigenom kan en *Cascading* WMS kombinera och överlagra kartbilder från olika WMS-servrar. Alla kartlager som kan visas av en *Cascading* WMS verkar ur ett klientperspektiv komma från samma WMS-server.

5.2.1 GetCapabilities

GetCapabilities-förfrågningar görs för att hämta metadata för tjänsten i fråga. Svaret på ett *GetCapabilities*-anrop talar bland annat om vilka informationsskikt eller kartlager som finns, vilka visningsstilar som finns för dessa samt vilka bildformat och referenssystem det går att få dem i.

```
http://www2.demis.nl/mapserver/request.asp?
      SERVICE=WMS&
      VERSION=1.1.1&
      REQUEST=GetCapabilities
```

Figur 5.3. GetCapabilities-anrop. I vanliga fall står hela anropet på en rad men det är här uppdelat för ökad läsbarhet. (Resultatet från anropet upptar i det här fallet cirka fem A4-sidor och det presenteras därför inte här.)

Den första raden i Figur 5.3 är den URL som pekar på den resurs på Internet där tjänsten finns. Resterande rader är CGI-parametrar med önskat värde. En resurs på webben kan innehålla flera olika tjänster och för att just WMS-serverprogrammet ska ta emot och behandla anropet finns parametern *service* som en del av alla WMS-anrop. *Service*-parametern ska alltid vid WMS-anrop sättas till "WMS".

Även *version*-parametern kan finnas med i alla WMS-anrop. Parametern berättar för servern vilken version av WMS-specifikationen klienten arbetar enligt. Det finns regler

för vilka versioner som ska vara kompatibla och om servern och klienten inte är kompatibla ska servern meddela detta i ett felmeddelande som returneras i stället för efterfrågade data. För *GetCapabilities*-anrop ska alla versioner vara kompatibla med varandra och därför är versionsparametern frivillig vid dessa anrop.

Request-parametern anger vilken funktion som efterfrågas i anropet, de möjliga värdena är *GetCapabilities*, *GetMap* och *GetFeatureInfo*. Det finns även en frivillig parameter för angivande av vilken version av *Capabilities*-dokumentet för en specifik server som finns hos klienten. Om klienten gör ett *GetCapabilities*-anrop och anger en äldre version av *Capabilities*-dokumentet än det som finns på servern returneras det nyare dokumentet, annars returneras ett undantagsmeddelande.

5.2.2 GetMap

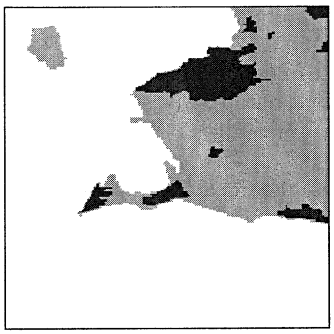
I Figur 5.4 visas ett exempel på ett *GetMap*-anrop och i Figur 5.5 visas resultatet av anropet. Anropet bygger på resultatet av det *GetCapabilities*-anrop som gjordes i Figur 5.3. Förklaring av de i förfrågningen ingående parametrarna följer här.

De första tre parametrarna beskrivs i 5.2.1 GetCapabilities. Med den fjärde parametern, *SRS* (*Spatial Reference System*), anger klienten i vilket koordinatreferenssystem kartbilden ska returneras. *SRS*-parametern anges med hjälp av en fyrställig sifferkombination enligt ett system definierat av European Petrolium Survey Group (EPSG). Den önskade kartbildens geografiska utsträckning anges med *BBOX*-parametern som utgörs av värdena x-min, y-min, x-max och y-max angivna i det koordinatsystem som angetts med hjälp av *SRS*-parametern.

Layers-parametern anger vilka kartlager som önskas. Geografiska informationssystem har vanligtvis kartdata indelade och lagrade i olika kategorier. Exempel på kategorier kan vara vägar, vattendrag, bebyggelse och sevärdheter. Flera lager från samma tjänsteleverantör kan efterfrågas i samma förfrågning och överlagras med varandra och dessutom kan en klient göra förfrågningar till olika leverantörer och själv överlagra de olika resultaten. Överlagring sker i den ordning lager efterfrågas med det först efterfrågade lagret underst. *Layers*-parameterns värde ska vara en kommaseparerad lista. Kartdata från olika lager i samma *GetMap*-förfrågan returneras alltid i samma geografiska referenssystem och eftersom klienten i varje förfrågning anger geografiskt referenssystem minskar risken för felaktig överlagring. Överlagring sker i den ordning som kartlagren anges i förfrågningen vilket innebär att det lager som anges sist ritas ut över alla andra lager.

```
http://www2.demis.nl/mapserver/request.asp?
      SERVICE=WMS&
      VERSION=1.1.1&
      REQUEST=GetMap&
      SRS=EPSG:4326&
      BBOX=12.7,55.2,13.2,55.7&
      LAYERS=Countries,Builtup areas&
      STYLES=default,default&
      WIDTH=200&
      HEIGHT=200&
      FORMAT=image/gif
```

Figur 5.4. *GetMap*-anrop. I vanliga fall står hela anropet på en rad men det är här uppdelat för ökad läsbarhet.



Figur 5.5. Resultatet från *GetMap*-anropet ovan (den svarta ramen är tillagd i efterhand för att tydliggöra kartbildens utsträckning).

Leverantören av karttjänsten kan skapa olika stilar för presentation av de kartdata som tjänsten tillhandahåller. Vilken stil som ska användas anger klienten med *styles*-parametern. *Styles*-parameterns värde ska vara en kommaseparerad lista med en stil för varje lager och om ingen stil anges för ett lager används en standardstil. OGC har även utvecklat en specifikation för ett gränssnitt som slår fast hur en klient i detalj ska kunna styra presentationen av kartdata (se avsnitt 5.5 Styled Layer Descriptor (SLD)). *Width*- och *height*-parametrarna anger i enheten pixlar vilken bredd respektive höjd kartbilden ska ha.

I *Capabilities*-dokumentet anges i vilka format WMS-servern kan leverera data. *Capabilities*-dokumentet ska alltid levereras i XML-format men svar på *GetMap*- och *GetFeatureInfo*-anrop kan levereras i ett flertal olika format. Grafikformat som kan användas för leverans av kartbilder är rasterformat och vektorformat. Specifikationen anger inga krav på format utöver detta men ger som exempel på rasterformat: GIF, PNG och JPEG. Och på vektorformat ges exemplen: SVG och *Web Computer Graphics Metafile* (WebCGM). Angående rasterformat påpekas också att en tjänst bör kunna leverera kartbilder i åtminstone ett rasterformat som kan hantera genomskinlighet. Detta för att möjliggöra överlagring av kartlager med varandra.

Ovanstående parametrar är obligatoriska i en *GetMap*-förfrågan, det finns också parametrar för genomskinlighet, bakgrundsfärg, höjdvärde, tid och dimension. Som

exempel på olika dimensioner anges satellitbilder över samma område vid samma tillfälle men med information från olika våglängdsband. Klienten kan även ange i vilket format undantagsinformation ska levereras. Det finns också möjlighet för tjänsteleverantören att definiera parametrar som är specifika för den egna tjänsten. Dessa leverantörsspecifika parametrar innebär att standarden kan användas även i specialicerade tillämpningar. För att bevara sin status som standard kräver WMS-specifikationen att en WMS-server som använder sig av leverantörsspecifika parametrar även ska kunna hantera anrop som endast innehåller de obligatoriska parametrarna.

5.2.3 GetFeatureInfo

GetFeatureInfo-funktionen används för att tillgängliggöra attributdata tillhörande de kartdata som visas av en WMS. *GetFeatureInfo*-anrop ska innehålla ungefär samma parametrar och värden som *GetMap*-anropet som gav de kartdata utifrån vilka attributförfrågningen görs. *Request*-parametern ska här ha värdet *GetFeatureInfo* och versionsparametern behöver inte ha samma värde i *GetFeatureInfo*-förfrågningen som i *GetMap*-förfrågningen. Utöver parametrarna i *GetMap*-förfrågningen ska *GetFeatureInfo*-förfrågningen innehålla en parameter som berättar vilket eller vilka kartlagers attributdata som efterfrågas samt x- och y-koordinat för pixeln, i kartbilden, för vilken information efterfrågas. Dessutom kan klienten ange vilket format attributdata ska returneras i samt antalet objekt för vilka attributdata ska returneras. Även vid *GetFeatureInfo*-anrop kan tjänstspecifika parametrar definieras. I Figur 5.6 visas ett *GetFeatureInfo*-anrop och i Figur 5.7 visas resultatet av anropet. Anropet bygger på resultatet av det *GetMap*-anrop som gjordes i Figur 5.5.

```
http://www2.demis.nl/mapserver/request.asp?
      SERVICE=WMS&
      VERSION=1.1.1&
      REQUEST=GetFeatureInfo&
      LAYERS=Countries,Builtup areas&
      STYLES=default,default&
      SRS=EPSG:4326&
      BBOX=12.7,55.2,13.2,55.7&
      WIDTH=200&
      HEIGHT=200&
      FORMAT=image/gif&
      QUERY_LAYERS=Countries,Builtup areas&
      FEATURE_COUNT=8&
      INFO_FORMAT=text/html&
      X=129&
      Y=42
```

Figur 5.6. *GetFeatureInfo*-anrop. I vanliga fall står hela anropet på en rad men det är här uppdelat för ökad läsbarhet.

Layer	ID	Description	Value
Builtup areas	5087	MALMO	
Countries	SW	Sweden	

Figur 5.7. Resultatet från *GetFeatureInfo*-anropet ovan.

Specifikationen av *GetFeatureInfo*-funktionen inriktar sig på rastergrafik och definierar inte funktionen i vektorfallet. SVG hanterar enskilda objekt som objekt även i renderingen vilket innebär att attributdata för specifika objekt skulle kunna efterfrågas istället för att attributdata för en pixel i en kartbild efterfrågas. Man uppger i specifikationen av WMS 1.1.1 att man vill utvärdera *GetFeatureInfo*-funktionen i den här versionen och eventuellt i kommande versioner specificera funktionen ytterligare med inriktning på vektorformat.

5.3 Web Feature Service (WFS)

Detta avsnitt bygger i huvudsak på Open GIS Consortiums (2002b) specifikation av WFS 1.0.0.

Web Feature Service (WFS) är en standard för överföring av geografiska data via webben. Förfrågningar efter data görs med hjälp av CGI-parametrar med värden i form av XML-syntax. Förfrågningar till en WFS-server efter geografiska data ska besvaras i GML-format. WFS-servrar kan även konstrueras för att ta emot anrop för att utföra ändringar av de data den tillhandahåller.

WFS-gränssnittet innehåller följande fem funktioner som alla anropas genom förfrågningar enligt URI-standarden och CGI på ungefär samma sätt som anrop till en WMS-server (Figur 5.2) sker:

- *DescribeFeatureType*-anrop utförs av klienten för att få en beskrivning av en, flera eller alla objektstyper som tjänsten kan leverera.
- För att efterfråga data utför klienten *GetFeature*-anrop. Klienten kan i ett *GetFeature*-anrop ange villkor för vilka objekt data ska returneras och vilka data som ska returneras för de objekt som uppfyller villkoren. Villkor som kan anges är till exempel typ av objekt, identitet och geografiskt läge.
- *Transaction*-anrop gör klienten för att utföra ändringar av de data som en WFS-server tillhandahåller.
- *LockFeature*-anrop används när klienten ska utföra ändringar på ett objekt. Innan klienten ska utföra ändringen gör den ett *LockFeature*-anrop sedan ändrar klienten objektet och när ändringen är utförd ges andra klienter åter tillgång till objektet.
- Innan något annat anrop görs från en WFS-klient till en WFS-server bör ett *GetCapabilities*-anrop utföras. Svaret på *GetCapabilities*-anropet är ett XML-dokument som innehåller metadata för tjänsten. Dokumentet innehåller bland annat information om vilka objektstyper tjänsten tillhandahåller, inom vilket geografiskt område de olika objektstyperna finns representerade, vilka koordinatreferenssystem som kan användas och så vidare. Dokumentet berättar också huruvida tjänsten implementerar *Transaction-* och *LockFeature*-funktionerna (se punkt tre respektive fyra).

I Figur 5.8 visas ett exempel på en *GetFeature*-förfrågan i XML-format till en WFS-server. När förfrågningen görs skrivs den på en rad och vissa tecken byts tillfälligt ut för att förfrågningen ska uppfylla URI-standarden. Rotelementet är *wfs:GetFeature* och som attribut till detta element anges *service*-parametern som talar om för serverdatorn att anropet ska tas omhand av en WFS-tjänst, *version*-parametern som anger vilken version

av WFS-standarden klienten implementerar samt *output*-parametern som anger i vilket format resultatet av förfrågningen önskas. I rotelementet deklareras också ett antal *namespace* (se 3.5.3 Namespace) och adressen till ett *XML-Schema*-dokument (se 3.5.2 DTD och XML-Schema) anges. I det andra elementet anges vilka data som önskas samt för vilka objekt dessa data efterfrågas. Vilka objekt som är aktuella anges med hjälp av elementet *ogc:Filter*. I det här fallet önskar klienten få veta egenskaperna *myns:NumFloors*, *myns:NumStudents* och *myns:geom* för objektet med id-nummer 329.

```
<?xmlversion="1.0"?>
<wfs:GetFeature
    service="WFS"
    version="1.0.0"
    outputFormat="GML2"
    xmlns:myns="http://www.someserver.com/myns"
    xmlns:wfs="http://www.opengis.net/wfs"
    xmlns:ogc="http://www.opengis.net/ogc"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/wfs../wfs/1.0.0/WFS-
    basic.xsd">
    <wfs:QuerytypeName="myns:school">
        <ogc:PropertyName>myns:NumFloors</ogc:PropertyName>
        <ogc:PropertyName>myns:NumStudents</ogc:PropertyName>
        <ogc:PropertyName>myns:geom</ogc:PropertyName>
        <ogc:Filter>
            <ogc:FeatureIdfid="329"/>
        </ogc:Filter>
    </wfs:Query>
</wfs:GetFeature>
```

Figur 5.8. Exempel på en *GetFeature*-förfrågan till en WFS-server.

5.4 OpenGIS Location Services (OpenLS)

Detta avsnitt bygger i huvudsak på Open GIS Consortiums (2004a) specifikation av OpenLS 1.0.

OpenGIS Location Services är en samling av flera standarder som, implementerade, i samverkan skapar ett nätverk av tjänster som utnyttjar varandra för att orientera användare med mobila terminaler. *Open GIS Core Services* utgör de grundläggande tjänsterna och består av följande fem specifikationer:

Directory Service (*eng.* katalogtjänst) definierar en tjänst som utifrån användarens geografiska läge eller en önskad position talar om var i närheten en efterfrågad produkt eller tjänst finns. Gula Sidorna är ett exempel på en katalogtjänst.

För att få reda på användarens position skickar *Directory Service*-klienten en förfrågan till en *Gateway Service*. Det finns flera olika sätt definierade för att ange en användares position men en *Gateway Service*-server behöver vid ett anrop bara kunna ange en användares position i ögonblicket för anropet. Andra möjligheter är att tjänsten återkommande anger en eller flera användares positioner. Positionen kan till exempel anges som upptagningsområdet för den basstation användarens terminal använder eller

beräknas mer noggrant genom utnyttjande av signalstyrka vid kommunikation med flera basstationer.

En *Location Utility Service* används för att översätta en adress eller ett platsnamn till en position. Översättningen ska kunna ske i båda riktningarna och dessutom ska tjänsten kunna returnera alla adresser inom ett angivet avgränsat geografiskt område. För att komplettera ofullständiga adressuppgifter ska tjänsten returnera fullständig adressinformation även om översättning från adress till position görs.

Presentation Service används av de andra OpenLS-tjänsterna för att presentera sina resultat. *Presentation Service* tar vid anrop fram en kartbild för aktuellt område och överlagrar medsänd information på kartbilden. Den information som de andra tjänsterna sänder kan till exempel vara ett område, en position eller en rutt.

Route Service används för att beräkna den i något avseende bästa vägen som förbinder två eller flera punkter. En rutt kan till exempel beräknas mellan användarens position för tillfället och en plats som användaren fått vetskap om genom en förfrågan till en *Directory Service*.

5.5 Styled Layer Descriptor (SLD)

En *Styled Layer Descriptor* är ett XML-dokument som beskriver för en WMS-server hur den ska rendera geografiska data. En SLD kan användas av tjänsteleverantören för att definiera de stilar som ska finnas tillgängliga genom WMS-anrop med hjälp av *Layers*- och *Styles*-parametrarna enligt WMS-specifikationen (se avsnitt 5.2.2 GetMap). Beroende på hur WMS-servern är konstruerad kan SLD-dokument användas för att låta klienten definiera stilarna på objekten i de beställda kartbilderna. Klienten kan också tillåtas att ange adressen till till exempel en WFS-server som innehåller geografiska data som klienten vill ha renderade. I samma anrop kan klienten ange adressen till ett SLD-dokument som definierar vilka data som ska ingå i olika kartlager och stilarna som ska användas för dessa kartlager. (OGC 2002d)

6 GiMoDig-projektet

Geospatial info-mobility service by real-time data-integration and generalisation (GiMoDig) är ett projekt som syftar till att undersöka metoder för att överföra spatiella data i vektorformat från de nationella kartverkens databaser till mobila användare. Överföringen ska ske med hjälp av XML-baserade standarder. Ett mål är också att undersöka möjligheterna att tillämpa Open GIS Consortiums standarder för webbaserade karttjänster. GiMoDig startades 2001 och ska pågå under tre år. Projektet finansieras av Information Society Technologies (IST) (GiMoDig 2003). IST är ett organ inom Europeiska Unionen som administrerar forskning inom teknikrelaterade områden (IST 2003).

Deltagarna i GiMoDig är:

- Finska geodetiska institutet (FGI) (projektkoordinator).
  http://www.fgi.fi
- Universität Hannover (UHANN), institut für kartographie und geoinformatik (IKG). Institutionen för kartografi och geoinformatik vid Universitetet i Hannover.
  http://www.ikg.uni-hannover.de
- Bundesamt für Kartographie und Geodäsie (BKG). Det tyska statliga verket för kartografi och geodesi.
  http://www.bkg.bund.de
- Kort og Matrikelstyrelsen (KMS). Det danska kartverket.
  http://www.kms.dk
- Lantmäteriet.
  http://www.lantmateriet.se
- Finska lantmäteriverket.
  http://www.maanmittauslaitos.fi

GiMoDigs mål är att möjliggöra distribution av sömlös kartinformation från de ingående aktörernas nationella kartverk till mobila användare. Kartinformationen ska integreras och generaliseras i realtid vilket tillgängliggör ständigt uppdaterade kartdata till slutanvändarna. Vidare finns följande delmål (GiMoDig 2003):

- Utveckling av metoder och användandepraxis för generalisering av grafiska representationer av geospatiella data i realtid, för anpassning för visning i varierande skala på små mobila enheter med olika upplösning.
- Utredning av problemen vid samkörning av nationella primära geospatiella databaser vilka ofta är inbördes heterogena i tematiska definitioner samt utveckling av metoder för realtidsharmonisering av data.
- Analys av mobila användningsfall för anpassande av realtidsgeneralisering och -harmonisering av geospatiella data till en användares krav i olika användningssituationer.
- Utveckling av metoder för realtidstransformation av spatiella data från olika nationella geodatabaser till ett gemensamt EUREF-baserat koordinatsystem.

- Utredning och utveckling av metoder för överföring av spatiella data i vektorformat till mobila användare med hjälp av nya standarder som XML och testande av tillämpbarheten hos standarder för webbaserade spatiella tjänster i ett internationellt pilotprojekt med nationella primära geodataset.
- Utveckling och implementation av ett prototypsystem som kan användas som testplattform för de utvecklade metoderna.

I denna rapport beskrivs dock endast de tekniska delarna och främst själva systemarkitekturen.

6.1 Systemarkitektur

Detta avsnitt följer främst Lehtos (2003) beskrivning av GiMoDig-projektets systemarkitektur.

GiMoDig-projektet har definierat en systemarkitektur bestående av fem lager där varje lager handhar en viss del av en förfrågan efter geografiska data (Figur 6.1). De lager som ingår i systemet är ett klientlager, ett portallager, ett databearbetningslager, ett dataintegrationslager och ett dataservicelager. Portallagret tar emot en förfrågan från något klientprogram och vidarebefordrar efter behandling förfrågningen via övriga lager till ett eller flera nationella kartverk. Det eller de nationella kartverk som mottagit förfrågningen svarar med att returnera efterfrågade kartdata som på vägen tillbaka genom lagerstrukturen behandlas innan de skickas till klientprogrammet av portallagret.

6.1.1 Förfrågan efter kartdata

Ett klientprogram (Figur 6.1 – Client) används för att med hjälp av till exempel knappsats eller mus och tangentbord låta användaren beskriva vilka kartdata som efterfrågas. Klientprogrammet tolkar användarens inmatningar och översätter förfrågningen till WMS- eller *OpenLS Presentation Service*-gränssnittet och skickar den till GiMoDig-tjänstens portallager.

Portallagret (Figur 6.1 – Portal) mottar förfrågningen från klientlagret och skickar en förfrågan till databearbetningslagret (Figur 6.1 – Data Processing) som efterfrågar de geografiska data med lämplig generalisering som krävs för att framställa den av klienten efterfrågade kartbilden. Det finns inom GiMoDig tankar på att i linje med Open GIS Consortiums webbtjänster skapa en standard, med tillhörande gränssnitt, för generaliseringstjänster. Detta gränssnitt är tänkt att användas för kommunikationen mellan portal- och databearbetningslager.

Databearbetningslagret skickar i sin tur en förfrågan till dataintegrationslagret (Figur 6.1 – Data Integration) efter geografiska data enligt en gemensam datamodell och i samma geografiska referenssystem. Denna kommunikation sker enligt WFS-standarden. Dataintegrationslagret efterfrågar de kartdata det behöver från dataservicelagret (Figur 6.1 – Data Service) som består av de medverkande ländernas nationella geografiska databaser med tillhörande WFS-servrar för distribution av geografiska data.

6.1.2 Behandling av kartdata

Grundstommen för hela projektet är de ingående ländernas nationella geografiska databaser vilka tillsammans med var sin WFS-server utgör det första lagret som kallas datatjänstlager (Figur 6.1 – Data Service). Kommunikationen mellan datatjänstlagret och dataintegrationslagret som är det andra lagret i systemarkitekturen ska ske med hjälp av Web Feature Service-standarden (WFS) från OGC.

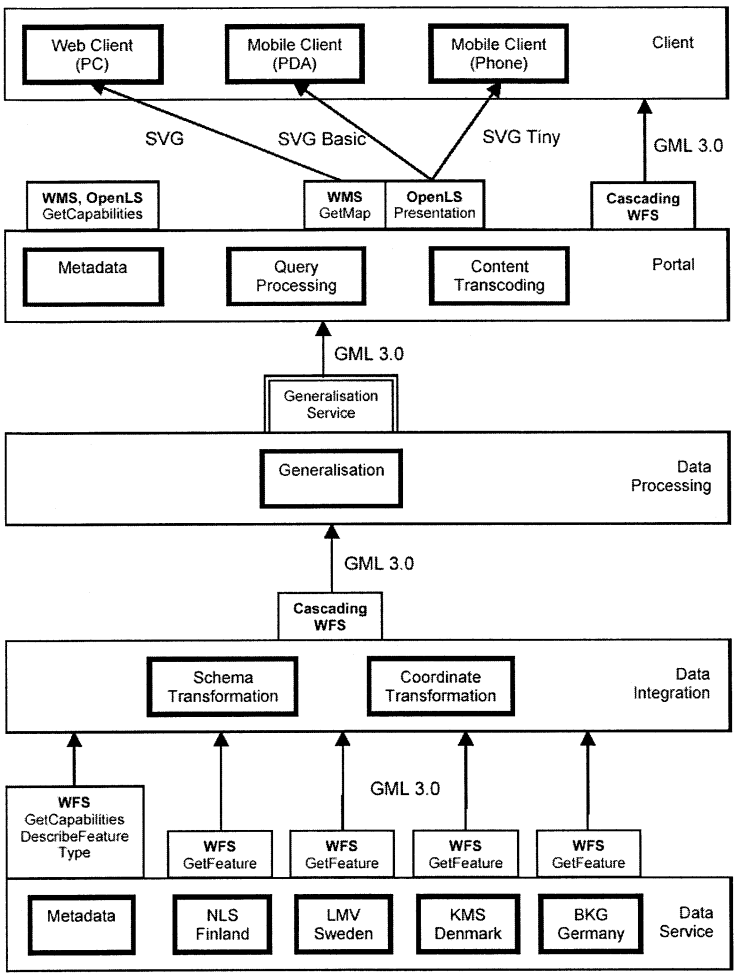
För att distribuera likvärdig kartinformation från alla länder har man utarbetat en gemensam datamodell som de ingående ländernas respektive datamodeller medger konvertering till. Dataintegrationslagret (Figur 6.1 – Data Integration) sköter transformation av de olika ländernas kartdata från respektive lands geografiska datamodell till denna gemensamma datamodell genom en XSL-transformation (XSLT). Lagret sköter också koordinattransformationer från respektive lands geografiska referenssystem till koordinatsystemet ETRS89 och UTM-projektion som ska användas för alla geografiska data inom projektet. ETRS89 realiseras i Sverige av SWEREF99 (Jivall m fl 2001).

Databearbetningslagret (Figur 6.1 – Data Processing) ska generalisera efterfrågade kartdata till en lämplig noggrannhetsnivå med tanke på kartskala, klientens mål med kartbilden och klientens hård- och mjukvara. Generaliseringen ska i enkla fall ske med hjälp av XSLT och i mer komplicerade fall med hjälp av kombinationer av XSLT (Lehto och Kilpeläinen 2000) och en Java-applikation (Harrie och Johansson 2003).

Portallagret (Figur 6.1 – Portal) mottar geografiska data från databearbetningslagret i GML-format och skapar utifrån dessa data en kartbild som passar klientprogrammet. Portallagret ska skicka kartdata till klienterna i SVG-format (eller något rasterformat). Beroende på klientplattformens kapacitet ska profilerna SVG, SVG Basic och SVG Tiny användas.

Tanken är att GiMoDig-systemet ska ha ett antal olika versioner av klientlagret för att kunna tillgodose olika klienthårdvara. Datorer utrustade med webbläsare med SVG-insticksprogram ska kunna använda tjänsten. Även mobila klienter i form av avancerade mobiltelefoner och PDA-enheter (*Personal Digital Assistant*) ska kunna använda tjänsten. För att överlämna en del av beräkningsarbetet åt klienthårdvaran kan en Java-applikation för klientbruk skapas.

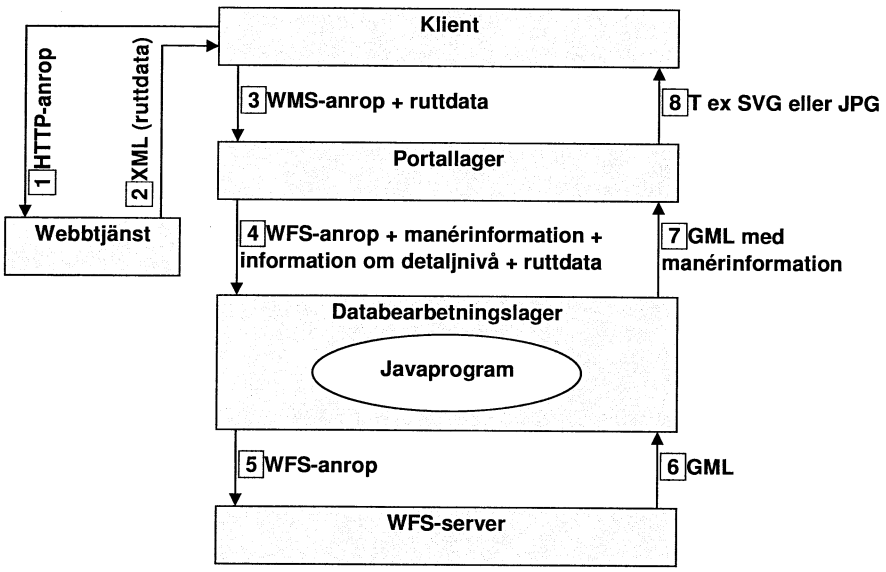
Eftersom GiMoDig-projektets systemarkitektur bygger på kommunikation mellan lagren med hjälp av öppna gränssnitt finns det också möjlighet att låta klienter kommunicera med portallagret med egenproducerade versioner av klientprogram. Användandet av öppna standarder möjliggör även kommunikation mellan klienter och andra lager i systemet än portallagret. Om en användare till exempel vill implementera motsvarigheten till GiMoDig-systemets portallager skulle användarens klientprogram kunna fås att kommunicera med generaliseringstjänsten i GiMoDig-systemets databearbetningslager.



Figur 6.1. GiMoDig-projektets systemarkitektur. (Lehto 2003)

6.2 Användande av andra tjänster

Tjänsteleverantörer kan använda GiMoDigs infrastruktur för att visualisera resultat av förfrågningar till den egna tjänsten. Till exempel kan en ruttningstjänst som beräknar den, i något avseende, bästa vägen mellan två eller fler punkter beräkna en rutt och visa den valda vägen i en karta som beställs från GiMoDig. Vid användande av externa tjänster sker först en förfrågning till den externa tjänsten och resultatet bifogas sedan lämpligen i något XML-baserat format till GiMoDig-tjänstens portallager. Databearbetningslagret sköter inpassningen av de data som bifogats från ruttningstjänsten (Figur 6.2). (Tuvesson och Harrie 2003)



Figur 6.2. Extern tjänst som använder sig av GiMoDig för visualisering i kartbild. (omarbetad från Tuvesson och Harrie 2003)

Praktisk del

7 Utveckling av klientprogram

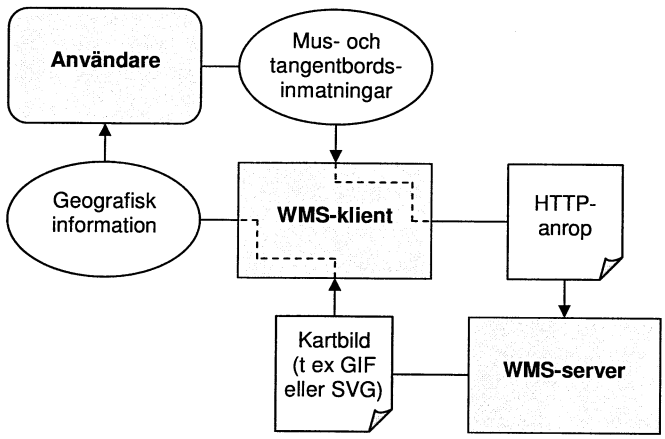
En Web Map Service-klient för visning av kartbilder och attributdata från WMS-servrar har skapats inom ramen för examensarbetet. Klienten är webbaserad och har konstruerats främst med hjälp av SVG och JavaScript. Klientapplikationen har utvecklats för att fungera på en dator med operativsystemet Microsoft Windows 2000, webbläsaren Microsoft Internet Explorer 6 och Adobe SVG Viewer 3.01. Klienten kan även fungera på andra plattformar men detta har inte utvärderats praktiskt.

Kapitlet inleds med en översiktlig beskrivning av klientapplikationen, därpå följer en beskrivning av användargränssnittet och de funktioner som finns tillgängliga för användaren. I avsnittet därefter beskrivs klientapplikationens tekniska uppbyggnad, här beskrivs också hur WMS-standardens GetCapabilities-funktion har implementerats genom hårdkodning av variabler. Sedan följer en beskrivning av hur applikationen hanterar kartdata i raster- respektive SVG-format. Sist beskrivs hur applikationen implementerar GetFeatureInfo-funktionen.

I denna del av rapporten omtalas ofta "WMS-servern" som datakälla för klienten men i själva verket kan flera WMS-servrar verka som källa för kartbilder visade i klienten. En kartbild som efterfrågas från en WMS-server kan vara en sammanställning av bilder från flera WMS-servrar om servern som kontaktas av klienten är en *cascading* WMS. En annan möjlighet är att klienten efterfrågar kartdata från flera servrar och själv överlagrar resultatet.

7.1 Översikt

Klientens uppgift är att omvandla användarens inmatningar, främst i form av musklickningar, till HTTP-GET-förfrågningar enligt WMS-standarden, förmedla dessa till en WMS-server, ta emot resultatet och infoga det i användargränssnittet så att det kan visas för användaren (Figur 7.1).



Figur 7.1. Schematisk bild av användarens interaktion med WMS-klienten och klientprogrammets interaktion med en WMS-server.

7.2 Användargränssnitt

Klientens grafiska användargränssnitt innehåller en kartbild, ett antal knappar, ett antal symboler, en skalstock samt en norrpil (Figur 7.2). Knapparna används för följande funktioner:

- Inzoomning
- Utzoomning
- Inställning av zoomfaktor
- Panorering
- Mätning av sträckor i kartbilden
- Nollställning av uppmätt sträcka
- Frånkoppling av alla funktioner
- Attributförfrågningar.

De data som visas intill kartbilden är:

- Pekarens x-koordinat
- Pekarens y-koordinat
- Eventuell uppmätt sträcka
- Id-attribut för det objekt i kartbilden som muspekaren befinner sig över.

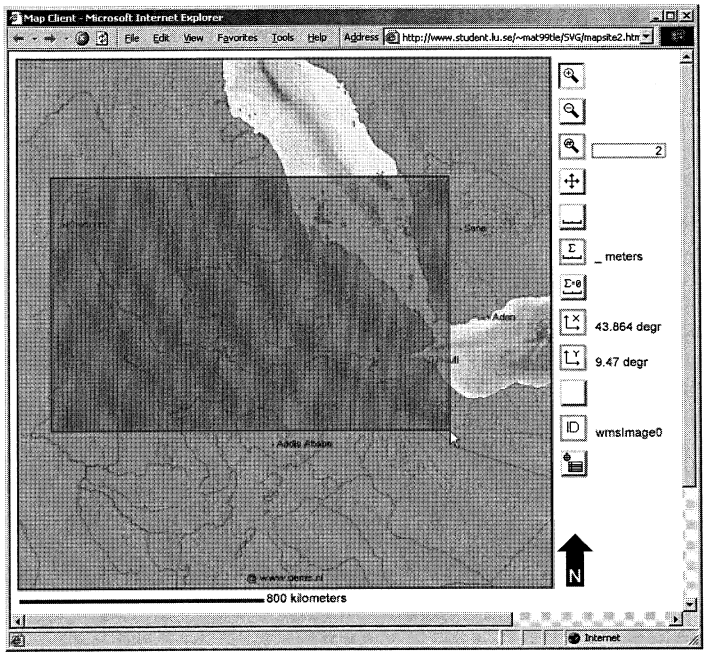
Alla funktioner väljs genom musklickning på knapparna i menyn vid kartbildens högra sida. På högra sidan om de symboler som saknar funktion visas resultat av användarens användning av vissa funktioner. Här följer en beskrivning av funktioner och symboler:

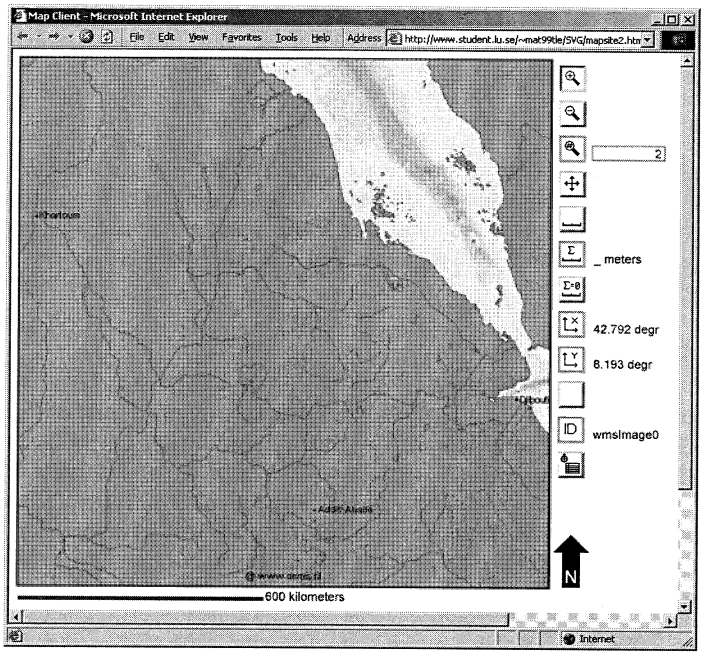
 *Inzoomning*

Inzoomning kan ske på två sätt. Användaren kan välja att klicka i kartbilden vilket medför att den nya kartbilden centreras kring klickningen och angiven zoomfaktor avgör förstoringsgraden. Om användaren trycker ner musknappen, håller den nere och flyttar

muspekaren visas en skuggad rektangel. Det område som syns i rektangeln när knappen släpps kommer att visas i den nya kartbilden. Om rektangeln inte har samma form som kartbilden visas minst det område som syntes i rektangeln. Den nya kartbilden centreras kring rektangelns mitt.

I Figur 7.2 har användaren skapat en zoomruta i form av en liggande rektangel. Den nya kartbilden visar zoomrutans innehåll i öst-västlig riktning men i nord-sydlig riktning visas mer kartdata än vad zoomrutan innehöll.





Figur 7.2. Inzoomning med hjälp av zoomrektangel (föregående sida) och resultatet av inzoomningen (ovan). Som synes befinner sig Addis Abeba utanför inzoomningsrektangeln men zoomrutan är proportionellt bredare än kartbilden. Därför avgör zoomrutans bredd även höjden på det område som ska visas i den nya kartbilden och därmed visas även till exempel Addis Abeba i denna.

 *Utzoomning*

Även utzoomning kan ske genom klickning och zoomrektangel. Om användaren väljer att klicka centreras den nya kartan kring platsen för klickningen och zoomfaktorn används för att beräkna bildens geografiska storlek. Vid användande av zoomruta centreras den nya kartbilden kring zoomrutans centrum och innehållet i den nuvarande kartbilden får i den nya kartbilden zoomrutans storlek. Om zoomrutan inte har samma form som kartfönstret ges den nuvarande kartbildens innehåll den storlek i den nya bilden som ges av rektangelns längre sidpar.

Exempel: kartbilden är kvadratisk, användaren skapar en zoomruta i form av en liggande rektangel. Den nya kartbilden kommer att visa föregående kartbilds innehåll med zoomrutans bredd.

 *Zoomfaktor*

Zoomfaktorn ställs in genom att användaren klickar i rutan som visar zoomfaktorn, håller kvar pekaren över rutan, skriver önskad zoomfaktor och trycker ned *enter*-tangenten - nu kan muspekaren flyttas fritt. För att avbryta inmatningen kan *escape*-tangenten användas.

För angivande av decimaltal används decimalpunkt. Vid inzoomning divideras kartbildens geografiska bredd och höjd med zoomfaktorn och vid utzoomning multipliceras desamma med zoomfaktorn.

 *Panorering*

Panorering kan ske genom klickning eller genom att användaren trycker ner musknappen, drar kartbilden till önskat läge och släpper knappen. Aktuell kartbild visas under förflyttningen men kartinformation utanför kartbildens kanter visas först när knappen släppts. Om användaren klickar i kartbilden kommer den nya kartbilden att centreras kring platsen för klickningen.

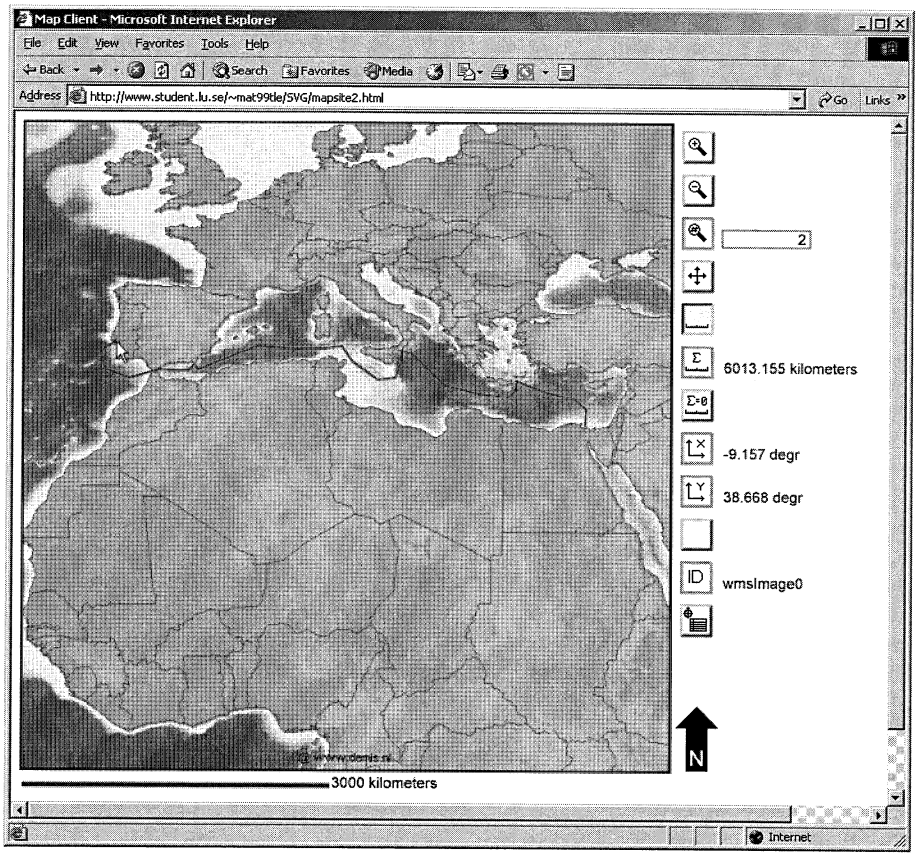
 *Mätning av sträckor*

Mätning av sträckor sker genom klickning i kartbilden. Användaren klickar på startpunkten för den önskade sträckan och en röd linje med startpunkt i mätningens startpunkt och slutpunkt vid muspekaren ritas ut. Användaren klickar där en brytpunkt önskas och den röda linjen får en brytpunkt i samma punkt (Figur 7.3). Observera att riktigheten hos den uppmätta sträckan beror på kartprojektionen i fråga. Vissa kartprojektioner innebär längdriktighet längs vissa linjer i kartan (till exempel längs medelmeridianen eller en parallellcirkel) men en karta kan aldrig vara längdriktig i alla riktningar i alla områden (Lantmäteriet 2004b).

 9.165 kilometers *Visning av summerad uppmätt sträcka*

 *Nollställning av uppmätt sträcka*

För att avbryta mätningen kan användaren välja någon annan funktion och för att nollställa den summerade uppmätta sträckan kan användaren klicka på knappen som är avsedd för detta. Om nollställningsknappen används är sträckmätningsverktyget fortfarande valt och det kan användas direkt igen. Om något annat verktyg väljs och används försvinner den röda linjen men den summerade sträckan nollställs inte och sträckmätningen kan fortsätta även om panorering eller zoomning har skett. Om mätningen fortsätts kommer en ny linje att påbörjas.



Figur 7.3. Mätning av sträckan för en båtfärd från Port Fouad till Lissabon.

Frånkoppling av funktioner
Genom att trycka på den blanka knappen slår användaren ifrån alla funktioner.

Attributförfrågningar
Användaren kan göra attributförfrågningar genom att klicka i kartbilden. Ett nytt fönster innehållande attributdata öppnas. Denna funktion kräver att WMS-servern som levererar kartbilder implementerar GetFeatureInfo-funktionen.

1383237.843 meters *Visning av x-koordinat i enheten meter*

55.842 degr *Visning av y-koordinat i enheten grader*
Vid symbolerna visas aktuell x- respektive y-koordinat, med aktuell enhet, för den position i kartan muspekaren befinner sig på. Koordinatvärdena uppdateras kontinuerligt när användaren rör muspekaren över kartbilden. Enheten för koordinatvisningen

hårdkodas i skriptfilen i samband med att parametrar från *Capabilities*-dokumentet hårdkodas (se 7.4 GetCapabilities-anrop). Observera att beräkningen av koordinaterna bygger på att koordinatsystemet som används är rätvinkligt och har konstant skala i x- och y-led vilket sällan är fallet (Lantmäteriet 2004b). Koordinaterna som visas är således sällan helt korrekta.

Visning av identitetsattribut

Om kartbilden är i SVG-format visas id-attributet för det objekt muspekaren befinner sig över. I detta fall visas id-attributet för den rasterbild som visas. Om kartbilden består av rasterbilder i flera lager visas id-attributet för den översta rasterbilden. Id-attributet återfinns i SVG-dokumentet.

Visning av skala

När kartskalan ändras beräknas en lämplig sträcka för visning med skalstocken och skalstockens längd ändras för att visa rätt skala. Med lämplig sträcka menas hela tiotal om skalstocken ska visa sträckor som är 10 längdenheter eller längre och kortare än 100 längdenheter, hela hundratal om en sträcka som är 100 längdenheter eller längre och kortare än 1000 längdenheter ska visas och så vidare. Dessutom visas sträckan i enheten meter om längden är under en kilometer och i enheten kilometer för sträckor däröver. Observera att riktigheten hos visad skala beror på kartprojektionen i fråga. Vissa kartprojektioner innebär längdriktighet längs vissa linjer i kartan (till exempel längs medelmeridianen eller en parallellcirkel) men en karta kan aldrig vara längdriktig i alla riktningar i alla områden (Lantmäteriet 2004b).

Visning av norriktning

Klientapplikationens norrpil är endast symbolisk, den är hårdkodad och visar alltid norriktningen rakt uppåt på skärmen.

7.3 Använd teknik

WMS-klienten består av tre filer

- en HTML-fil (wms_client.html)
- en SVG-fil (gui.svg)
- en JavaScript-fil (functions.js).

SVG-filen innehåller det grafiska användargränssnittet med knappar, skalstock, norrpil och plats för kartbild. JavaScript-filen innehåller funktioner för avläsning av användarens inmatningar, för kommunikation med WMS-servrar, för beräkningar samt för utritning av information i det grafiska användargränssnittet. SVG-filen bäddas in i en HTML-fil för att enkelt kunna visas som en webbsida.

I SVG-filen finns en referens till den externa JavaScript-filen functions.js. Eftersom SVG-filen är inbäddad i HTML-filen leder referensen till denna. I HTML-filen finns också en referens till samma externa JavaScript-fil. Denna referens utnyttjas av SVG-filen och leder till filen functions.js.

Ytterligare två filer kan sägas höra till klienten, de är skapade av Adobe och används för att utföra automatisk installation av SVG-insticksprogrammet till vissa webbläsare. Dessa filer är:

- svgcheck.js
- svgcheck.vbs

och finns tillgängliga för nedladdning på Adobes hemsida (Adobe 2004). Dessutom används bilder i GIF-format för symbolerna och knapparna i användargränssnittets meny.

7.4 GetCapabilities-anrop

Klienten implementerar endast GetMap- och GetFeatureInfo-funktionerna, GetCapabilities-anrop måste utföras manuellt och adressen till WMS-resursen och ett antal parametrar för möjliggörande av GetMap- och GetFeatureInfo-förfrågningar måste därför hårdkodas i JavaScript-koden. Dessa parametrar fås från Capabilities-dokumentet.

De parametrar som måste hårdkodas i förväg i JavaScript-koden för varje WMS-server är:

- SRS — koordinatreferenssystem
- BBOX — kartbildens geografiska utsträckning i utgångsläget
- LAYERS — lista med önskade kartlager och lista med önskade lager som tillåter attributförfrågningar
- STYLES — lista med fördefinierade presentationsstilar, en för varje kartlager (ej obligatoriskt)
- FORMAT — bildformat för kartbilden.
- EXCEPTION — önskat format för undantagsinformation från servern för GetMap-förfrågningar och önskat format för undantagsinformation från servern för GetFeatureInfo-förfrågningar
- FEATURE_COUNT — det antal objekt för vilket attributdata önskas
- INFO_FORMAT — önskat format för attributdata.

Även URL till respektive WMS-server samt enhet för koordinatangivelser (meter eller grader) måste hårdkodas i förväg

Med hjälp av de metoderna som beskrivs i 7.5.1 Kartdata i SVG-format skulle även *GetCapabilities*-funktionalitet kunna implementeras. *Capabilities*-dokumentet skulle automatiskt efterfrågas när klienten laddades och programmet skulle sedan identifiera tillgängliga koordinatreferenssystem, bildformat, kartlager, kartlager med tillgängliga attributdata samt presentationsstilar för olika kartlager. Ett exempel på en WMS-klient som fungerar på detta sätt har konstruerats av Intergraph (Intergraph 2004a).

7.5 GetMap-anrop

När klientwebbsidan laddas skapas och skickas en GetMap-förfrågan till angiven WMS-server. Kartbilden som returneras används som utgångsläge i kartfönstret. Alla funktioner

som därefter förändrar kartbildens geografiska utsträckning använder sig också av GetMap-förfrågningar. Om användaren till exempel väljer att zooma in utförs beräkningar i skriptet och en vektor, innehållande den önskade kartbildens begränsningar motsvarande användarens zoomruta, konstrueras. Värdena i denna vektor infogas som värde för BBOX-parametern i en textsträng som utgör en GetMap-förfrågan som skickas till WMS-servern. Se Appendix A Programstruktur och Appendix B Programkod för beskrivning av händelseförloppet vid användning samt skript- och SVG-kod.

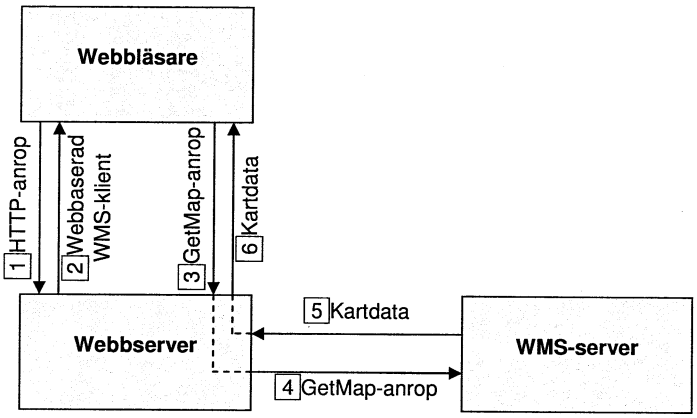
När användaren gör något val som kräver att en ny kartbild efterfrågas från WMS-servern beräknar skriptet värden för parametrarna i en GetMap-förfrågan och sammanställer dessa med parameternamnen till en textsträng. Med hjälp av DOM sätts värdet på ett *xlink:href*-attribut, för ett *image*-element i gui.svg, till textsträngens värde. Om kartdata efterfrågas från flera WMS-servrar infogas varje GetMap-förfrågan i ett *image*-element.

7.5.1 Kartdata i SVG-format

Även kartdata i SVG-format kan visas med metoden som beskrivs i 7.5 GetMap-anrop men om ett SVG-dokument hänvisas till i ett *image*-element visas SVG-dokumentet utan någon funktionalitet, SVG-läsaren ges inte tillgång till elementen i dokumentet. Detta innebär att skriptet inte får tillgång till SVG-kartbildens DOM och kan inte ändra utseende på objekt i kartbilden eller hantera elementens attribut.

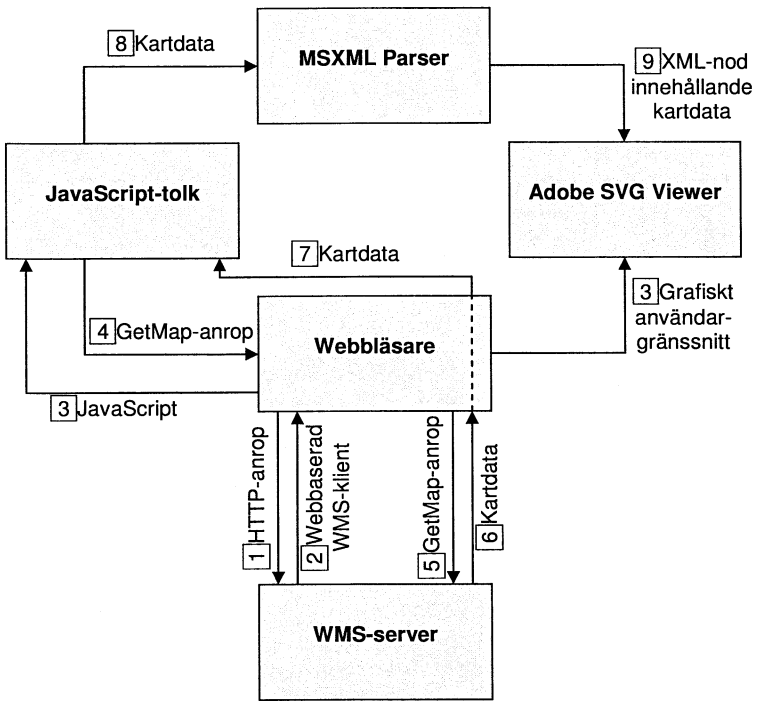
Det finns ett generellt problem med att göra SVG-kartbildens DOM tillgänglig för skriptkoden. Adobes SVG-insticksprogram innehåller funktionalitet utöver vad som krävs av specifikationen för DOM. Två av dessa tillägg är funktionerna *getURL(url, callback)* och *parseXML(text, doc)*. *getURL* laddar dokumentet som anges med argumentet *url*, argumentet *callback* ska peka på en funktion som hanterar resultatet av hämtningen. *parseXML* skapar en DOM-nod av textsträngen *text* i dokumentet *doc*. Den resulterande noden kan sedan adderas till ursprungsdokumentets DOM. Denna metod kan användas för att addera en kartbild i SVG-format till klienten. Men *getURL* tillåts av säkerhetsskäl endast ladda data som finns på samma server som dokumentet dessa data ska laddas till (Grey Magic 2004). Detta innebär att klienten måste finnas på samma server som WMS-tjänsten finns på. Detta skulle innebära att klienten endast skulle kunna visa SVG-kartdata från en WMS-server vilket innebär en alltför stor begränsning för att detta ska vara en tillfredsställande lösning. Det finns åtminstone två metoder att kringgå problemet.

En möjlighet att kringgå detta problem är att på servern som klienten placeras på, skapa ett serverprogram som hanterar förfrågningar till önskade WMS-servrar (Figur 7.4). Klienten laddas från servern till en användares dator, serverprogrammet tar från klienten emot alla GetMap-förfrågningar som efterfrågar SVG-kartbilder och skickar dem vidare till WMS-servern. WMS-servern returnerar kartbilden till servern där serverprogrammet tar emot efterfrågade kartdata och skickar dem vidare till klienten. Eftersom kartbilden skickats från klientens egen server tillåter SVG-insticksprogrammet att getURL-funktionen behandlar kartbilden. En nackdel med denna metod är att den bara fungerar om användaren har installerat Adobes SVG-insticksprogram.



Figur 7.4. GetMap-förfrågan via webbservern som huserar klientapplikationen.

En annan metod (Figur 7.5) att kringgå problemet är att använda Microsoft XML Parser (MSXML Parser) för att läsa in önskade kartdata i SVG-format med funktionen *load(url)*, som fungerar på samma sätt som *getURL*, och addera den till *gui.svg* med *parseXML* som ovan. Nackdelen med denna metod är att den endast fungerar om Microsoft Internet Explorer 5.0 eller senare används som webbläsare. Även här krävs SVG-insticksprogrammet från Adobe. Anledningen till att metoden endast fungerar med Internet Explorer 5 eller senare och MSXML Parser är att funktionen *load* är ett tillägg till specifikationen av DOM, på samma sätt som *getURL* och *parseXML* i Adobe SVG Viewer, samt att MSXML Parser endast fungerar i samverkan med Internet Explorer 5 eller senare.



Figur 7.5. Kartdata i form av ett SVG-dokument formateras om till en XML-DOM-nod som adderas till det grafiska användargränssnittet.

De båda ovanstående metoderna har undersökts i arbetet med utvecklingen av klientapplikationen. För att undersöka den första metoden har Internet Information Server använts som webbserver och Active Server Pages (ASP) i kombination med Visual Basic Script har använts för att tillföra funktionalitet på serversidan. Även andra webbservrar som hanterar ASP kan användas för att utnyttja denna metod. Ytterligare ett alternativ är att använda språket Java server Pages (JSP) i samverkan med ett serverprogram som kan hantera JSP. Den andra metoden har undersökts med hjälp av Microsoft XML Parser 4.0 i kombination med Microsoft Internet Explorer 6.0.

Det relativt sett större plattformsoberoendet som medges av den första metoden talar för denna. En nackdel med denna metod är att den som vill använda klientapplikationen tvingas använda den via Internet om personen i fråga inte har tillgång till en egen webbserver som kan hantera skript eller programmering på serversidan. Den andra metoden innebär att klienten inte behöver laddas från en webbserver. Båda metoderna har undersökts med lyckat resultat men det har i arbetet inte funnits utrymme att implementera metoderna i klientapplikationen.

Det finns exempel på karttjänster där SVG används och där anrop av skript för förändring av kartbilden och för rapportering av attributdata placeras i varje element i SVG-dokumentet. För att möjliggöra lagring av andra attribut i SVG-dokumentet än de som

definieras i SVG-specifikationen används ett eget *namespace*. Vid användande av denna metod styr tjänsteleverantören vad som ska kunna ändras i kartbilden och placeringen av skript-anropen i SVG-dokumentet medför att filstorlekarna för levererade kartdata ökar. (Intergraph 2004b)

7.6 GetFeatureInfo-anrop

Om användaren väljer att göra en attributdataförfrågan registreras koordinaterna för användarens klickning i kartbilden. Dessa och andra nödvändiga parametrar infogas i en textsträng som utgör en *GetFeatureInfo*-förfrågan. Förfrågningen skickas till WMS-servern och svaret visas i ett nytt webbläsarfönster. Genom att utnyttja metoderna som nämndes i avsnitt 7.5.1 Kartdata i SVG-format skulle klienten kunna hantera svar från *GetFeatureInfo*-anrop och visa resultatet i det grafiska användargränssnittet istället för att, som nu är fallet, visa resultatet i ett nytt webbläsarfönster.

WMS-specifikationens beskrivning av GetFeatureInfo-funktionen inriktar sig främst på tillämpningar med rastergrafik och slår inte fast hur attributdataförfrågningar ska ske med utgångspunkt från en vektorkartbild. Det är därför intressant att undersöka andra möjligheter än de som ges i rasterfallet att efterfråga attributdata.

En möjlighet att förmedla attributdata genom SVG-formatet är att i varje grafiskt element i det av WMS-servern konstruerade SVG-dokumentet ange alla attributdata för motsvarande geografiskt objekt. Detta kan utföras med hjälp av separata *namespace* för attributdata och SVG-attribut. Denna metod innebär att alla attributdata finns tillgängliga hos klienten och de går att nå med DOM. Metoden innebär också att geografiska data med extensiva attributdata medför stora SVG-dokument och därmed långa överföringstider.

En annan möjlighet är att tillämpa samma metod som i rasterfallet vilket innebär att en *GetMap*-förfrågan görs, en pixel i resultatbilden väljs ut och huvuddelen av *GetMap*-förfrågningen, pixelns koordinater i bilden och kartlager som information önskas från, anges i en *GetFeatureInfo*-förfrågning. Denna metod innebär att serverprogrammet på något sätt måste räkna ut vilket geografiskt objekt som efterfrågas.

En tredje möjlighet är att varje geografiskt objekt som tillgängliggörs av en WMS-tjänst ges ett id-attribut. Id-attributet ges också till motsvarande grafiskt objekt när SVG-dokumentet skapas. *GetFeatureInfo*-förfrågningar skulle nu kunna göras genom att klienten efterfrågar attributdata för ett objekt med ett visst objekt-id. Användaren skulle med den här metoden kunna försäkra sig om att attribut för rätt objekt efterfrågas genom att klientprogrammet markerar objektet för vilket attributdata hämtas genom att till exempel ändra objektets färg i kartbilden.

Ett problem med denna metod är hur id-attribut ska hanteras när kartbilden består av geografiska data från flera olika WFS-servrar eller från en *cascading* WMS. En lösning kan vara att objekt från olika servrar ges olika prefix vid inläsning av kartdata till klienten. Ett annat problem uppstår när grafiska objekt helt täcks av andra objekt. I rasterfallet löses detta genom att klienten tillåts ange för hur många objekt attributdata

ska efterfrågas. En liknande lösning är tänkbar i SVG-fallet men urvalet är då enklast att utföra på klientsidan vilket skulle innebära att klienten skulle göra separata attributdataförfrågningar för valda objekt.

Situationer där ett grafiskt objekt i kartbilden representerar flera geografiska objekt kan innebära svårigheter vid efterfrågande av attributdata. Ett exempel på detta kan vara att ett antal vägsegment i de geografiska data som ska visualiseras representeras av en sammanhängande väglinje i kartbilden. En lösning på detta kan vara att låta det grafiska objektets id-attribut referera till alla de ingående vägsegmenten. En mer naturlig lösning är att varje geografiskt objekt representeras av ett och endast ett grafiskt objekt.

8 Test av WMS-klientens kompatibilitet

Eftersom WMS är tydligt definierad som en öppen standard bör en klientapplikation som skapats med utgångspunkt från denna standard kunna kommunicera med alla serverimplementationer av samma standard. För serverimplementationer av WMS-standarden finns ett testprogram som tillhandahålls av OGC (OGC 2004b). Någon liknande tjänst finns inte för klientimplementationer.

I examensarbetet har kommunikation med följande serverimplementationer undersökts:

- deegree WMS
- MapServer
- Demis WMS
- Intergraph GeoMedia WebMap Professional WMS Adaptor Kit
- ESRI ArcIms WMS Connector/OGC Interoperability Add-On
- GiMoDig.

Nedan följer en kortfattad beskrivning av respektive implementation, vem som administrerar den installation av respektive server som använts samt eventuella problem vid kommunikation med respektive server.

8.1 deegree

deegree är ett projekt som drivs av GIS- och fjärranalysenheten vid Bonns universitets geografiska avdelning och lat/lon som är en avknoppning av GIS-gruppen på Bonns universitet. deegree har utvecklat den officiella referensimplementationen av WMS-standarden vilket innebär att klientprogrammets kompatibilitet gentemot den WMS-server som utvecklats i deegree bör vara god. WMS-servern från deegree kan leverera kartdata i formaten PNG, JPG, TIF, BMP, GIF, JPG och SVG. deegree har implementerat samtliga delar av WMS-standarden. (deegree 2004)

Den installation av deegree WMS som testats är projektets egen demonstrationsserver. All kommunikation mellan klientapplikationen och deegrees WMS-server fungerade problemfritt.

8.2 MapServer

MapServer är ett projekt med öppen källkod som drivs av avdelningen för skogsresurser vid universitetet i Minnesota i samarbete med NASA och myndigheter i delstaten Minnesota. MapServer implementerar WMS-standarden och kan leverera kartdata i formaten GIF, PNG, JPG, BMP, TIF och SWF och kan fungera som en *cascading* WMS och kan även rendera geografiska data från en WFS-server. (MapServer 2004)

Den MapServer-installation som testats i arbetet finns hos Helsingborgs stad. Kommunen har installerat MapServer på försök bland annat för att hantera kartdata i kommunens webbaserade version av översiktsplanen. (Pikkuniemi 2004)

Attributdataförfrågningar gentemot denna server lyckades inte när önskat format för attributdata sattes till HTML men förfrågningarna fungerade när något av de två andra i *Capabilities*-dokumentet annonserade formaten valdes. Inga andra problem har uppstått vid kommunikation med MapServers WMS-server.

8.3 Demis

Demis är ett Holländskt företag som utvecklar kartbaserade användargränssnitt till stöd för beslutsfattande inom områdena vatten, trafik, luft och energi. Demis WMS-server kan leverera kartdata i formaten GIF, PNG, BMP, JPG och SWF. (Demis 2004)

Den installation av Demis WMS som testats är företagets egen demonstrationsserver.

Demis WMS-server har använts som testserver genom hela utvecklingsarbetet och kommunikationen har genomgående fungerat utan problem.

8.4 Intergraph – GeoMedia WebMap Professional WMS Adaptor Kit

Intergraph har en GIS-programsvit som kallas GeoMedia. Som en del av denna svit finns GeoMedia WebMap som är ett programpaket för publicering av karttjänster på Internet som bygger på att tjänsteleverantören använder GeoMedia som GIS-plattform. *WMS Adaptor Kit* är en del av WebMap som kan användas för att göra kartdata tillgängliga via WMS-gränssnittet. Intergraphs WMS-server kan leverera kartdata i formaten PNG och JPG. (Intergraph 2004b)

Den installation av Intergraphs WMS-server som testats finns hos Kristianstads kommun. Kommunen har installerat WMS-tillägget i utvärderingssyfte. (Grimheden 2004)

Kommunikationen med Intergraphs WMS-server har fungerat problemfritt.

8.5 ESRI – OGC Interoperability Add-On och ArcIMS WMS Connector

ESRI har liksom Intergraph utvecklat ett programpaket för att publicera karttjänster på Internet och har möjliggjort användande av WMS-standarden för visning av kartdata från sina system. ESRIs WMS-server kan leverera kartdata i formaten JPG, GIF och PNG. (ESRI 2004a)

Den installation av ESRI:s produkter som testats i arbetet finns hos Geography Network (2004), en webbsida som tillhandahålls av ESRI för tillgängliggörande av geografisk information.

Vid förfrågningar efter attributdata returnerades dessa i ArcXML-format oavsett vilket av de tre tillgängliga formaten (*text/plain*, *text/html*, *text/xml*) som angavs som önskat. ArcXML är en XML-tillämpning som utvecklats av ESRI för uppmärkning av data för utbyte mellan företagets produkter (ESRI 2004b). I *Capabilities*-dokumentet för tjänsten som användes angavs att attributförfrågningar inte var möjliga och därför läggs ingen vikt vid tjänstens användande av icke önskat format för attributdata. Inga övriga problem har uppstått vid kommunikation med ESRI:s WMS-server.

8.6 GiMoDig

Anrop till en partiell implementation av GiMoDig-projektets systemarkitektur har testats. WMS-servern som utgör en del av portallagret i denna implementation använder sig av det fria biblioteket Batik från Apache för att leverera kartbilder i JPG- och PNG-format. Kartbilder i SVG-format skapas genom XSL-transformation av GML-data som efterfrågas av WMS-servern från en WFS-server. WMS-serverprogrammet består av en servlet skapad i Java-miljö. (Lehto 2004)

WMS-servern administreras av GiMoDig.

Eftersom funktionalitet för att hantera SVG-kartbildens DOM inte implementerats i klienten visas kartbilden i stort sett på samma sätt som en rasterbild. För att möjliggöra avläsning av musklickningar i kartbildens bakgrund (ej på något objekt) infogades en vit rektangel under kartbilden i SVG-dokumentet som utgör det grafiska användargränssnittet. När användaren klickar i bakgrunden registreras klickningen med hjälp av den tillagda rektangeln. Efter att bakgrundsrektangeln lagts till fungerade kommunikationen mellan klienten och GiMoDig:s WMS-server utan problem. Funktionen *GetFeatureInfo* är ännu inte implementerad varför endast GetMap-anrop har testats.

8.7 Utvärdering av kompatibilitetstest

Vid kommunikation med ovan nämnda WMS-servrar har *GetCapabilities*-förfrågning utförts manuellt medan *GetMap*- och *GetFeatureInfo*-förfrågningar har utförts av klientapplikationen. *GetMap*-förfrågningar har i samtliga fall fungerat problemfritt men *GetFeatureInfo*-förfrågningar gentemot två av servrarna uppvisade problem angående val av format för den efterfrågade informationen. I ett av fallen fungerade inte attributdataförfrågningar vid användande av ett av tre, enligt *Capabilities*-dokumentet, tillgängliga format. I det andra fallet angavs i *Capabilities*-dokumentet att attributdataförfrågningar ej var möjliga varför detta problem ej kan ligga server eller klient till last.

9 Diskussion

Genom att använda ett öppet och väldokumenterat gränssnitt kan kartografiska data från många oberoende källor användas i samma karttjänst och i samma kartbild. Samtidigt innebär användandet av en gemensam standard vissa begränsningar. I WMS-standarden har man övervunnit vissa av dessa begränsningar genom att tillåta leverantörsspecifika parametrar i anropen. Samtidigt kräver specifikationen att en WMS-server ska kunna hantera anrop som endast innehåller de obligatoriska parametrarna.

Det skulle innebära vissa fördelar att kunna efterfråga attributdata för enskilda objekt i en kartbild istället för att efterfråga attributdata för alla objekt i vissa angivna kartlager i närheten av en punkt. Vid användande av vektorgrafik finns de tekniska förutsättningarna för att implementera en sådan lösning på klientsidan. Det är dock tveksamt om OGC kommer att specificera ett förfarande som detta i nästa eller någon version av WMS-specifikationen.

Det finns i Adobe SVG Viewer skriptkommandon för att läsa filer och lägga till informationen från dessa filer i ett SVG-dokument. Till exempel kan en kartbild i SVG-format genereras av en WMS-server och läggas till i en webbaserad WMS-klient. Dessa kommandon tillgängliggör filer på datorn på vilken de används och man har därför i den senaste versionen av Adobe SVG Viewer begränsat dessa kommandon till att endast ge tillgång till filer på den dator varifrån skriptet hämtades. Detta medför att tillgång inte heller ges till filer från en WMS-server om inte den webbaserade klienten kommer från samma server.

I examensarbetet har det visats att det går att kringgå denna begränsning utan förlorad säkerhet genom vidarebefordran av förfrågningar efter SVG-kartdata genom den server som huserar klientapplikationen. När denna metod används verkar det för SVG-läsaren som om de kartdata som behandlas kommer från klientapplikationens värddator trots att de kan komma från WMS:er på olika servrar. En annan metod som också undersökts är att låta något program, som inte har denna begränsning, hantera tolkningen av XML-data. Huruvida metoden att använda andra program för tolkning av XML-data (i det här fallet Microsoft XML Parser) är tillförlitlig i fråga om säkerhet, har inte undersökts. Möjligtvis kan det vara intressant för W3C att i nästa version av SVG-standarden specificera om, och kanske i också hur, funktioner liknande Adobes *getUrl* och *parseXML* ska implementeras.

För att SVG ska vara ett lämpligt format för kartdata krävs det att leverantören av kartdata är noggrann i angivandet av skalor för visning av olika kartskikt och eventuellt att denne använder sig av multipel representation och/eller automatisk generalisering. För att det ska vara intressant att distribuera kartdata i SVG-format från en WMS-tjänst är det dessutom i stort sett avgörande att klientapplikationen har tillgång till SVG-dokumentets DOM så att förändringar av kartbilden kan göras och attribut lagrade i SVG-dokumentet tillgängliggörs. Om så inte är fallet finns ingen anledning att inte istället använda rasterformat. Undersökningen av möjligheter att hantera attributdata från SVG-dokument har försvårats av avsaknaden av WMS-servrar som distribuerar kartdata i SVG-format.

Användandet av SVG eller andra XML-baserade vektorgrafikstandarder kommer att bero mycket på webbläsarleverantörernas stöd för dessa standarder. I dagsläget finns det endast en webbläsare som innehåller SVG-funktionalitet och de vanligaste webbläsarna kräver installation av insticksprogram för hantering av SVG.

Det vore som en fortsättning på arbetet intressant att implementera en av eller båda de metoder som i arbetet tagits fram för tillgängliggörande av attributdata från SVG-dokument erhållna från WMS-servrar. Det vore också intressant att implementera *GetCapabilities*-funktionen för automatisk konfigurering av klienten. För att göra klienten mindre kan de bildfiler som används för symbolerna i användargränssnittet ersättas av SVG-kod.

10 Slutsatser

Web Map Service-standarden från Open GIS Consortium (OGC) specificerar hur kartdata kan efterfrågas via Internet och hur förfrågningar efter kartdata ska besvaras. Standarden beskriver också hur förfrågningar efter attributdata för objekt i kartbilden kan göras.

Litteraturstudien visar att det finns standarder som tillsammans möjliggör skapandet av en klientapplikation för kommunikation med serverimplementationer av WMS-standarden. I den praktiska delen har en sådan klientapplikation skapats. Applikationen har ett grafiskt SVG-baserat användargränssnitt och innehåller funktioner för att panorera, zooma och göra förfrågningar efter attributdata för objekt i kartbilden vilket innebär att klientprogrammet implementerar WMS-standardens *GetMap*- och *GetFeatureInfo*-funktioner. Användaren kan dessutom mäta sträckor i kartan och avläsa muspekarens geografiska koordinater. *GetCapabilities*-förfrågningar måste utföras manuellt och resultatet måste hårdkodas i klientapplikationen.

Arbetet har även visat att användandet av kombinationer av vissa av de undersökta standarderna är under utveckling. Under arbetets gång har ett flertal fritt tillgängliga WMS-tjänster på Internet provats men ingen av dessa levererar kartdata i SVG-format. Det finns två huvudsakliga orsaker till detta:

Visning av stora områden och många kartlager i SVG-format utan generalisering innebär stora SVG-dokument och därmed långa överföringstider. Om en standard för generaliseringstjänster fanns skulle detta problem inte vara lika stort som den är i nuläget.

Visualisering med SVG innebär upphovsrättsliga problem eftersom koordinatdata skickas till slutanvändaren. Visserligen är koordinaterna ofta relativa och inte angivna i ett geografiskt koordinatsystem men absoluta koordinater i lämpligt koordinatsystem går att beräkna. Organisationer som publicerar kostnadsfria, icke upphovsrättsligt skyddade data berörs inte av detta.

WMS-specifikationen ger inte någon vägledning om hur attributdataförfrågningar ska ske då SVG eller vektorgrafikformat i allmänhet används. En möjlighet som diskuterats i arbetet är att i SVG-koden infoga identitetsattribut för varje objekt i kartbilden. När användaren klickar på ett objekt efterfrågas önskade attributdata för det geografiska objekt som har samma identitetsattribut.

Det praktiska arbetet visar att det med relativ enkelhet är möjligt att skapa en webbaserad klientapplikation enligt WMS-standarden för visning av kartdata i rasterformat (GIF och PNG). Klienten kan även utföra visning av kartdata i SVG-format och tester som utförts i arbetet visar att visning av attributdata lagrade i SVG-koden är möjlig. För att detta ska vara praktiskt möjligt i klienten måste dock visst utvecklingsarbete utföras.

Klienten består i huvudsak av tre filer: En SVG-fil som innehåller det grafiska användargränssnittet, en HTML-fil som bäddar in SVG-filen och möjliggör visning av den i en webbläsare via Internet samt en JavaScript-fil som innehåller funktioner för

hantering av användarens inmatningar i användargränssnittet, kommunikation med WMS-servrar, beräkningar och visning av kartbild och annan information i användargränssnittet. Klientens totala storlek är cirka 47 kilobyte vilket innebär en nedladdningstid på cirka sju sekunder vid en nedladdningshastighet på 56 kb/s.

Klientens kompatibilitet har testats gentemot ett antal WMS-servrar. WMS-servrarna som använts är både gratisprogramvara och kommersiella produkter. Bland andra har kommunikation med den officiella referensimplementationen av WMS-specifikationen testats. *GetMap*-funktionen fungerade gentemot samtliga testade WMS-servrar och *GetFeatureInfo*-funktionen fungerade i alla fall utom två. I det ena av dessa fall angavs det i *Capabilities*-dokumentet för tjänsten att attributdata inte var tillgänglig för de skikt för vilka attributdata efterfrågades men attributdata levererades, dock ej i önskat format. I det andra fallet kunde inte attributdata levereras i ett av de två format som fanns angivna i tjänstens *Capabilities*-dokument.

Referenser

Litteratur

Cecconi, A., och Galanda, M., 2002. Adaptive Zooming in Web Cartography, *Computer Graphics Forum*, Volym 21, Nummer 4, Sidorna 787-799.

ECMA, 1996. *ECMAScript Language Specification* (tillgänglig på: http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf).

Ek, J. och Eriksson, U., 2001. *Lättpocket om JavaScript*, Pagina Förlags AB, Sundbyberg.

Eklundh, L., 2000. *Geografisk informationsbehandling: metoder och tillämpningar*, Byggforskningsrådet, Stockholm.

Gonzalez, R. C. och Woods, R. E., 2002. *Digital image processing*, Prentice Hall, Upper Saddle River, USA.

Gosling, J., Joy, B., Steele, G. och Bracha, G., 2000. *The Java Language Specification, second edition*, Addison-Wesley Pub Co, Harlow, Storbritannien (tillgänglig på: http://java.sun.com/docs/books/jls/second_edition/html/j.title.doc.html).

Hall, E., 2000. *Internet Core Protocols: The Definitive Guide*, O'Reilly, Sebastopol, Californien, USA.

Harold, E. R. och Means, W. S., 2001. *XML in a Nutshell*, O'Reilly, Sebastopol, Californien, USA.

Harrie, L. och Johansson, M., 2003. Real-time data generalisation and integration using Java. *Geoforum Perspektiv*, Februari, 2003, Sidorna 29-34.

Holm, P., 1999. *Objektorienterad programmering och Java*, andra upplagan, Studentlitteratur, Lund.

Jaworski, J., 2000. *Mastering JavaScript*, Sybex Inc., Alameda, USA.

Jivall, L., Lindberg, M., Lilje, M. och Reit, B.-G., 2001. *Transformationssamband mellan SWEREF 99 och HT 90/RH 70*, Lantmäteriet, Gävle.

Kraak, M.-J. och Brown, A., 2000. *Web Cartography: developments and prospects*, Taylor & Francis Group, London, Storbritannien.

Köbben, B. J. och Kraak, M-J., 1999. Webcartography: Dissemination of Spatial Data on the Web, *Proceedings of the 2nd AGILE conference on Geographic Information Science*,

Rom, Italien, Sidorna 14-18
(tillgänglig på:
http://kartoweb.itc.nl/kobben/publications/AGILErome/Maps%20and%20the%20Web.pdf).

Lehto, L., och Kilpeläinen, T., 2000. Real-Time Generalization of Geodata in the Web. *International Archives of Photogrammetry and Remote Sensing*, Volym XXXIII, del B4, Amsterdam, Sidorna 559-566.

Lehto, L., 2003. *GiMoDig System Architecture*
(tillgänglig på: http://gimodig.fgi.fi/pub_deliverables/Gimodig_D4_1_1-Arch_Spec.pdf).

Musciano, C. och Kennedy, B., 2002. *HTML and XHTML: The Definitive Guide, fifth edition*, O'Reilly, Sebastopol, USA.

NCSA, 1998. *The Common Gateway Interface*
(tillgänglig på: http://hoohoo.ncsa.uiuc.edu/cgi/interface.html).

OGC, 2002a. Styled *Layer Descriptor Implementation Specification version 1.0.0*
(tillgänglig på: http://www.opengis.org/docs/02-070.pdf).

OGC, 2002b. *Web Map Service Implementation Specification version 1.1.1*
(tillgänglig på: http://www.opengis.org/docs/01-068r3.pdf).

OGC, 2002c. *Web Feature Service Implementation Specification version 1.0.0*
(tillgänglig på: http://www.opengis.org/docs/02-058.pdf).

OGC, 2002d. *Styled Layer Descriptor Implementation Specification version 1.0.0*
(tillgänglig på: http://www.opengis.org/docs/02-070.pdf).

OGC, 2003. *Geography Markup Language (GML) Implementation Specification version 3.0*
(tillgänglig på: http://www.opengis.org/docs/02-023r4.pdf).

OGC, 2004a. *Location Services (OpenLS): Core Services version 1.0*
(tillgänglig på: http://portal.opengis.org/files/?artifact_id=3418).

Peng, Z.-R. och Tsou, M.-H., 2003. *Internet GIS : distributed geographic information services for the internet and wireless networks*, Wiley, Hoboken, USA.

Pikkuniemi, A., 2002. *Tillämpning av Scalable Vector Graphics i webbaserade karttjänster*, Examensarbete vid Luleå Tekniska Universitet, Luleå.

Tuvesson, H., och Harrie, L., 2003. Integration of navigational and cartographic data. *Proceedings of the 21st International Cartographic Conference (ICC), Cartographic Renaissance,* 10 - 16 Augusti 2003, Durban, Sydafrika, Sidorna 2572-2578.

W3C, 1998. *Document Object Model (DOM) Technical Reports*
(tillgänglig på: http://www.w3.org/DOM/DOMTR#dom1).

W3C, 1999. *Hypertext Transfer Protocol -- HTTP/1.1*
(tillgänglig på: ftp://ftp.isi.edu/in-notes/rfc2616.txt).

W3C, 2000. *Document Object Model (DOM) Technical Reports*
(tillgänglig på: http://www.w3.org/DOM/DOMTR#dom2).

W3C, 2001a. *XML Schema*
(tillgänglig på: http://www.w3.org/TR/xmlschema-0/).

W3C, 2001b. *URIs, URLs, and URNs: Clarifications and Recommendations 1.0*,
(tillgänglig på: http://www.w3.org/TR/uri-clarification/).

W3C, 2003a. *Document Object Model (DOM) Technical Reports*
(tillgänglig på: http://www.w3.org/DOM/DOMTR#dom3).

W3C, 2003b. *Scalable Vector Graphics (SVG) 1.1 Specification*
(tillgänglig på: http://www.w3.org/TR/SVG11/).

W3C, 2004. *Extensible Markup Language (XML) 1.0, third edition*
(tillgänglig på: http://www.w3.org/TR/WD-xml-961114.html).

Internet

Adobe, 2003. *Saving Compressed SVG (SVGZ)*, Adobe Systems Incorporated,
http://www.adobe.com/svg/illustrator/compressedsvg.html, besökt 2003-12-03.

Adobe, 2004. *About the SVG Auto-Installer*, Adobe Systems Incorporated,
http://www.adobe.com/svg/workflow/autoinstall.html, besökt 2004-03.

Apache, 2004. *Xalan Java version 2.6.0*, Apache,
http://xml.apache.org/xalan-j/index.html, besökt 2004-03-08.

carto:net, 2004a. *Interactive Topographic Map Tuerlersee*,
http://www.carto.net/papers/svg/tuerlersee/, besökt 2004-03-15.

carto:net, 2004b. *Links to SVG and mapping sites*,
http://www.carto.net/papers/svg/links/, besökt 2004-03-19.

deegree, 2004.
http://deegree.sourceforge.net/, besökt 2004-03-23.

Demis, 2004.
http://www.demis.nl/, besökt 2004-03-23.

ESRI, 2004a. *Interoperability Technology Download Center*,
http://www.esri.com/software/opengis/interopdownload.html, besökt 2004-03-29.

ESRI, 2004b. *ArcXML Programmer's Reference Guide 4.0.1*,
http://support.esri.com/index.cfm?fa=downloads.samplesUtilities.viewSample&PID=16
&MetaID=73, besökt 2004-03-29.

Geography Network, 2004.
http://www.geographynetwork.com, besökt 2004-05-20.

GiMoDig, 2003.
http://gimodig.fgi.fi, besökt 2003-12-01.

Grey Magic, 2004. *ASV 3.0 Security Issue*,
http://www.greymagic.com/security/advisories/gm004-mc/, besökt 2004-03-29.

Gula Sidorna, 2004.
http://www.gulasidorna.se/, besökt 2004-03-15.

Intergraph, 2004a. *Intergraph OGC WMS Viewer*,
http://www.wmsviewer.com, besökt 2004-03-22.

Intergraph, 2004b. *GeoMedia WebMap Professional*,
http://imgs.intergraph.com/gmwp/, besökt 2004-03-23.

ISO, 2004. International Organization of Standarization,
http://www.iso.org/iso/en/ISOOnline.frontpage, besökt 2004-03-15.

IST, 2003. Information Society Technologies,
http://www.cordis.lu/ist/, besökt 2003-12-01.

Lantmäteriet, 2004a. *Målbild 2000*,
http://www.malbild2000.lm.se/, besökt 2004-03-06.

Lantmäteriet, 2004b. *Kartprojektioner*
http://www.lm.se/geodesi/kartprojektion/projektion.htm, besökt 2004-03-26

Malmö, 2004. *Malmö Stadskarta*,
http://malmo.se/html/www/malmokarta/karta/lev1/, besökt 2004-03-15.

map24, 2004.
http://www.map24.com/, besökt 2004-03-16.

MapServer, 2004.
http://mapserver.gis.umn.edu/, besökt 2004-03-23.

Mozilla, 2003. *Mozilla SVG Project*,
http://www.mozilla.org/projects/svg/, besökt 2003-12-15.

OGC, 2004b. *Resources - Compliance Testing*,
http://www.opengis.org/resources/?page=testing, besökt 2004-05-20.

OpenSWF.org, 2004. *SWF Format Specification*,
http://www.openswf.org/spec.html, besökt 2004-05-21.

RFC, 2004. *The Common Gateway Interface - RFC Project Page*,
http://cgi-spec.golux.com/, besökt 2004-02-09.

U.S. Census Bureau, 2004.
http://tiger.census.gov/cgi-bin/mapbrowse-tbl, besökt 2004-03-15.

Unisys, 2004. *LZW Patent and Software Information*,
http://www.unisys.com/about__unisys/lzw, besökt 2004-03-19.

W3C, 2003c. *W3C Scalable Vector Graphics (SVG) – History*,
http://www.w3.org/Graphics/SVG/History, besökt 2003-12-16.

Personlig korrespondens

Grimheden, Mikael. Stadsingenjörskontoret, Kristianstads kommun, mars 2004.

Lehto, Lassi. Finska Geodetiska Institutet, mars-maj 2004.

Pikkuniemi, Anders. Stadsbyggnadskontoret, Helsingborgs stad, mars 2004.

Appendix

Appendix A. Programstruktur

I det följande kommer kommunikationen mellan olika programdelar vid användande av applikationen att beskrivas. Beskrivningen behandlar händelseförloppet vid inzoomning med hjälp av zoomruta. Se även Tabell 1 för en kortfattad beskrivning. Delar av händelseförloppet är gemensamt för samtliga funktioner.

I SVG-dokumentet som utgör applikationens grafiska användargränssnitt finns instruktioner för vad som ska hända beroende på vilka inmatningar användaren gör. När användaren klickar på knappen för inzoomning genereras ett händelseobjekt och funktionen *onClickMenu(evt)* anropas för att hantera händelsen. I händelseobjektet *evt* finns information om vilket objekt som genererade händelsen och i det här fallet var det zoomknappens grafiska objekt i SVG-dokumentet. I funktionen ges variabeln *zoomIn* värdet *true*. I det ögonblick som musknappen trycks ned anropas funktionen *mDownMenu(evt)* som ändrar utseendet på den nedtryckta knappen så att den ser ut att vara intryckt.

När användaren nu håller muspekaren över kartbilden och trycker ner musknappen anropas funktionen *onMDownMap(evt)* som sätter variabeln *mDown* till *true* registrerar pekarens koordinater. Hela tiden när användaren rör muspekaren över kartbilden anropas funktionen *mMoveMap(evt)*. I SVG-dokumentet finns ett rektangelelement vars position och storlek förändras så länge användaren håller musknappen nere (*mDown=true*) och flyttar muspekaren. Rektangelns ena hörn placeras i positionen för användarens klickning och det motstående hörnet sätts kontinuerligt till muspekarens aktuella position genom att rektangelns bredd och höjd sätts till skillnaden mellan pekarens aktuella koordinater och koordinaterna i ögonblicket då musknappen trycktes ner. Pekarens aktuella koordinater rapporteras kontinuerligt genom funktionen *mMoveMap(evt)* och händelseobjektet *evt*.

När användaren släpper musknappen anropas funktionen *onMUpMap(evt)* som bland annat ger pekarens koordinater när musknappen släpps och sätter variabeln *mDown* till *false*. Funktionen anropar sedan funktionerna *panFunc(evt, panType)* och *zoomFunc(evt, zoomType)* för att utföra panorering respektive inzoomning enligt zoomrutan.

Pekarens koordinater vid tidpunkten när musknappen trycktes ned respektive när den släpptes upp används för att beräkna den önskade kartbildens geografiska utsträckning. Följande stycke beskriver hur muspekarens geografiska x-koordinat.

Först beräknas pekarens koordinat i kartbilden i enheten pixlar genom att kartbildens position subtraheras från pekarens koordinater i SVG-dokumentet. Sedan divideras kartbildens geografiska bredd i till exempel enheten meter med bildens bredd i enheten pixlar vilket kan sägas vara kartbildens skala i enheten meter per pixel. Detta värde multipliceras med pekarens x-koordinat och till resultatet adderas kartans västliga begränsning. Kartbildens geografiska bredd fås från föregående *GetMap*-förfrågan och kartbildens position i SVG-dokumentet erhålls från SVG-dokumentet med hjälp av dess DOM

Funktionen *panFunc(evt, panType)* beräknar med utgångspunkt från aktuell kartbilds läge och zoomrutans läge däri den önskade kartbildens centrum och ändrar värdena i vektorn *bBox* så att dessa beskriver kartbildens geografiska begränsningar efter panorering till nytt önskat centrum men en ny kartbild efterfrågas först efter att även skalförändring utförts av funktionen *zoomFunc(evt, zoomType)*.

Funktionen *zoomFunc(evt, zoomType)* beräknar med utgångspunkt från befintlig kartbilds och zoomrutas storlek önskad kartbilds geografiska vidd och höjd och ändrar värdena i vektorn *bBox* så att dessa beskriver kartbildens geografiska begränsningar efter inzoomning. *zoomFunc(evt)* anropar funktionen *setWmsQuery(evt)*.

setWmsQuery(evt) sammanfogar URL:en till aktuell WMS-server och parametrar för att efterfråga av användaren önskad kartbild till en WMS-fråga. Som värde för BBOX-parametern i WMS-frågesträngen används värdena i *bBox*-vektorn. I SVG-dokumentet finns ett *image*-element som har ett attribut vars värde är URL:en till aktuell kartbild. Värdet på detta attribut sätts med hjälp av DOM till WMS-frågesträngen och den av klienten önskade kartbilden syns i WMS-klientens grafiska användargränssnitt.

Händelse i det grafiska användargränssnittet (gui.svg)	Anropad funktion i skriptfilen (functions.js)	Utförd åtgärd (i algoritmisk form)
Musklickning i menyn	onClickMenu	zoomIn=true
Musknappen trycks ned över menyn	mDownMenu	SVG-DOM: nedtryckt menyknapps utseende ändras
Musknappen trycks ned över kartbilden	onMDownMap	mDown=true x-koordinat lagras y-koordinat lagras
Muspekaren flyttas över kartbilden	mMoveMap	SVG-DOM: zoomrutans form ändras
Musknappen släpps upp över kartbilden	onMUpMap	mDown=false x-koordinat lagras y-koordinat lagras panFunc anropas zoomFunc anropas
	panFunc	bBox=#,#,#,#
	zoomFunc	bBox=#,#,#,# setWmsQuery anropas
	setWmsQuery	SVG-DOM: ny URL för kartbilden anges

Tabell 1. Händelsegång inom klientapplikationen vid användarens nyttjande av zoomverktyget (zoomruta).

Appendix B. Programkod

Ur filen *functions.js*

```
/** Anropas när musknappen trycks ner över kartbilden
 */
function onMDownMap(evt){
  if(!mDown){
    if(zoomIn || zoomOut){
      /** Sätt zoomBox-elemetets x- och y-attribut
       */
      zBoxX=evt.clientX;
      zBoxY=evt.clientY;
      svgDoc.getElementById("zoomBox").setAttribute("visibility",
      "visible");
    }
    if(pan){
      mDownX=evt.clientX;
      mDownY=evt.clientY;
      lastX=evt.clientX;
      lastY=evt.clientY;
    }
  }
  mDown=true;
}
```

```
/** Anropas då pekaren är i rörelse över kartbilden
 */
function mmoveMap(evt){
  xInMap=evt.clientX-mapPosX
  yInMap=evt.clientY-mapPosY
  /** Visar pekarens koordinater i kartans koordinatsystem
   */
  xCoord=bBox[0]+xInMap*(bBox[2]-bBox[0])/mapWidth;
  yCoord=bBox[1]+(mapHeight-yInMap)*(bBox[3]-bBox[1])/mapHeight;
  svgDoc.getElementById("coordX").firstChild.setData(Math.round(xCoord*
  1000)/1000+" "+coordUnit);
  svgDoc.getElementById("coordY").firstChild.setData(Math.round(yCoord*
  1000)/1000+" "+coordUnit);
  /** Om musknappen hålls nere över kartilden när muspekaren är
   *  i rörelse ska olika funktioner anropas beroende på vilket
   *  verktyg i menyn som är valt
   */
  if(mDown){
    if(zoomIn || zoomOut){
      zoomBox=svgDoc.getElementById("zoomBox");
      zoomBox.setAttribute('x', Math.min(evt.clientX, zBoxX)+"");
      zoomBox.setAttribute('width', Math.abs(zBoxX-evt.clientX)+"");
      zoomBox.setAttribute('y', Math.min(evt.clientY, zBoxY)+"");
      zoomBox.setAttribute('height', Math.abs(zBoxY-evt.clientY)+"");
    }
    if(pan){
      /** Flytta befintlig kartbild så att föremål i kartan
       *  hamnar i den posittion som de får i den nya kartbilden
       */
```

```
      vb[0]=vb[0]-(evt.clientX-lastX);
      vb[1]=vb[1]-(evt.clientY-lastY);
      setVbString(evt);
      lastX=evt.clientX;
      lastY=evt.clientY;
    }
  }
  /** Här har ett antal kommandon klippts bort
   *  eftersom de inte behövs för förståelsen av programmet
   */
}
```

```
/** Anropas när musknappen släpps efter att ha
 *  hållits nere över kartbilden
 */
function onMUpMap(evt){
  zoomBox=svgDoc.getElementById("zoomBox");
  mDown=false;
  if(zoomIn || zoomOut){
    zoomBox.setAttribute("visibility", "hidden");
    /** Använd panFunc(evt, panType) och
     *  zoomFunc(evt, zoomType) för att panorera
     *  och zooma enligt zoomBox
     */
    zBoxW=Math.abs(zBoxX-evt.clientX);
    zBoxH=Math.abs(zBoxY-evt.clientY);
    /** Om en liten zoomRuta skapas tolkas inmatningen
     *  som en klickning
     */
    if(zBoxW>5 || zBoxH>5){
      panFunc(evt, "box");
      zoomFunc(evt, "box");
    }
    if(zBoxW<=5 && zBoxH<=5 && zBoxW>0 && zBoxH>0){
      panFunc(evt, "click");
      zoomFunc(evt, "click");
    }
    zoomBox.setAttribute('x', "0");
    zoomBox.setAttribute('width', "0");
    zoomBox.setAttribute('y', "0");
    zoomBox.setAttribute('height', "0");
  }
  if(pan){
    /** Använd panFunc(evt, panType) för att panorera
     *  enligt pekarens förflyttning
     */
    lastX=evt.clientX;
    lastY=evt.clientY;
    vb[0]=0;
    vb[1]=0;
    vb[2]=mapWidth;
    vb[3]=mapHeight
    setVbString(evt);
    panFunc(evt, "drag");
  }
}
```

```
/** Panorerar kartan så att positionen för användarens
 *  musklickning centreras, så att kartbilden hamnar på
 *  positionen dit användaren dragit den eller så att
 *  zoomrutans centrum centreras
 */
function panFunc(evt, panType){
  /** Bestämmer nytt geografiskt centrum för kartbilden
   */
  if(panType=="click"){
    centerX=bBox[0]+(evt.clientX-mapPosX)*(bBox[2]-bBox[0])/mapWidth;
    centerY=bBox[1]+(mapHeight-(evt.clientY-mapPosY))*(bBox[3]-bBox[1])
    /mapHeight;
  }
  if(panType=="drag"){
    centerX=bBox[0]-(bBox[2]-bBox[0])*(evt.clientX-mDownX)/mapWidth+
    (bBox[2]-bBox[0])/2;
    centerY=bBox[1]-(bBox[3]-bBox[1])*(-(evt.clientY-mDownY))/mapHeight
    +(bBox[3]-bBox[1])/2;
  }
  if(panType=="box"){
    xSign=1;
    if(evt.clientX<zBoxX)
      xSign=-1;
    ySign=1;
    if(evt.clientY<zBoxY)
      ySign=-1;
    centerX=bBox[0]+(zBoxX+zBoxW/2*xSign-mapPosX)*(bBox[2]-bBox[0])/
    mapWidth;
    centerY=bBox[1]+(mapHeight-(zBoxY+zBoxH/2*ySign-mapPosY))*(bBox[3]-
    bBox[1])/mapHeight;
  }
  /** Den nya bBox-vektorn beräknas
   */
  xDif=bBox[2]-bBox[0];
  yDif=bBox[3]-bBox[1];
  bBox[0]=centerX-xDif/2;
  bBox[1]=centerY-yDif/2;
  bBox[2]=centerX+xDif/2;
  bBox[3]=centerY+xDif/2;
  /** Om panoreringsfunktionen anropats av zoomfunktionen
   *  ska wms-frågan skickas till servern först när både
   *  panorering och zoomning beräknats
   */
  if(pan){
    setWmsQuery(evt);
  }
}
```

```
/** Förstorar eller förminskar kartan med aktuell zoomfaktor
 *  eller enligt zoomruta
 */
function zoomFunc(evt, zoomType){
  /** Den nya bBox-vektorn beräknas. De kartdata som den nya bBoxen
   *  innefattar får bredd (xMax-xMin)/(zoomfaktor^zoomriktning)
   *  zoomriktningen är 1 (inzoomning) eller -1 (utzoomning)
   *  zoomfaktorn är ett tal >= 1
```

```
   *  Exempel:
   *  Aktuell geografisk bredd: 1000 km
   *  Inzoomning (zoomriktning: 1) med zoomfaktor: 4
   *  Ny geografisk bredd = 1000/4^1 m = 250 m
   */
  if(zoomType=="click"){
    newWidth=(bBox[2]-bBox[0])/Math.pow(zoomFactor, zoomDirection);
    newHeight=(bBox[3]-bBox[1])/Math.pow(zoomFactor, zoomDirection);
  }
  if(zoomType=="box"){
    if(zBoxW/mapWidth>=zBoxH/mapHeight){
      newWidth=(bBox[2]-bBox[0])/Math.pow(zBoxW, -zoomDirection)/
      Math.pow(mapWidth, zoomDirection);
      newHeight=(newWidth/mapWidth)*mapHeight;
    }
    else{
      newHeight=(bBox[3]-bBox[1])/Math.pow(zBoxH, -zoomDirection)/
      Math.pow(mapHeight, zoomDirection);
      newWidth=(newHeight/mapHeight)*mapWidth;
    }
  }
  bBox[0]=bBox[0]/1+(bBox[2]-bBox[0])/2-newWidth/2;
  bBox[1]=bBox[1]/1+(bBox[3]-bBox[1])/2-newHeight/2;
  bBox[2]=bBox[0]+newWidth;
  bBox[3]=bBox[1]+newHeight;
  setWmsQuery(evt);
  updateScale(evt);
}
```

```
/** Genererar WMS-frågan och sätter in den som xlink:href-attribut
 *  i det svg-element som motsvarar kartbilden
 */
function setWmsQuery(evt){
  /** Generar GetMap-frågesträngen
   */
  bboxString=bBox[0]+","+bBox[1]+","+bBox[2]+","+bBox[3];
  wmsString=serverList[listNbr]+
    'service='+'WMS'+
    '&version='+versionString+
    '&request='+'getMap'+
    '&layers='+layerList[listNbr]+
    '&styles='+styleList[listNbr]+
    '&srs='+srsList[listNbr]+
    '&bbox='+bboxString+
    '&width='+mapWidth+
    '&height='+mapHeight+
    '&format='+formatList[listNbr]+
    '&exceptions='+mapExceptionFormatList[listNbr];

  /** Sätter kartbildens xlink:href-attribut till WMS-
   *  frågesträngens värde
   */
  svgDoc.getElementById("wmsImage0").setAttribute("xlink:href",
  wmsString);

  /** Här har ett antal kommandon klippts bort eftersom de inte behövs
```

```
    *  för förståelsen av programmet
    */
}
```

Ur filen *gui.svg*

```
<!-- Vissa delar har klippts bort eftersom de inte behövs för
förståelsen av programmet-->
<svg onmousemove="mmoveMap(evt)" onmousedown="onMDownMap(evt)"
onmouseup="onMUpMap(evt)" x="10" y="10" width="600" height="600"
viewBox="">
  <rect id="mapBack" width="600" height="600" fill="white"/>
  <image id="wmsImage0" width="600" height="600" xlink:href=""/>
</svg>
```

Appendix C. Ordlista

Förkortning/ Begrepp	Förkortning utläst	Förklaring
CGI	Common Gateway Interface	Standard som möjliggör interaktivitet i webbsidor genom förmedlande av parametrar från webbläsare till program på webbservrar
CGM	Computer Graphics Metafile	Vektorgrafikstandard utvecklad av W3C
CSS	Cascading Style Sheets	Standard från W3C för angivande av presentationsstilar för elementen i till exempel HTML-dokument
DOM	Document Object Model	Standard från W3C för skriptspråks åtkomst till bland annat XML- och HTML-dokument
DTD	Document Type Definition	Språk för angivande av de regler som ska uppfyllas av ett XML-dokument för att det ska vara giltigt enligt en viss XML-tillämpning (jämför med XML-Schema)
ECMA	European Computer Manufacturers Association	Standardiseringsorgan
ECMAScript		Skriptspråk baserat på JavaScript och JScript, standardiserat av ECMA
EPSG	European Petroleum Survey Group	Organisation som bland annat ajourhåller en databas med uppgifter om koordinatreferenssystem från hela världen
FTP	File Transfer Protocol	Standard för överföring av filer
geografiska data		Spatiella data, eventuellt kopplade till icke spatiella data
GIF	Graphics Interchange Format	Rastergrafikformat med ickeförstörande komprimering
GiMoDig	Geospatial info-mobility service by real-time data-integration and generalisation	IST-projekt som syftar till att skapa en metod för realtidsgeneralisering och -integrering av geografiska data från olika länder för mobila användare
GIS	Geografiska informationssystem (Geographic Information System)	Datoriserade system för insamling, lagring, analys och presentation av geografiska data
GML	Geography Markup Language	XML-tillämpning från OGC för uppmärkning av geografiska data
HTML	Hypertext Markup Language	Standard för uppmärkning av information för presentation i webbläsare

HTTP	Hypertext Transfer Protocol	Standard för överföring av hypertextdokument
hypertext		Text med klickbara länkar till information med anknytning till texten
Internet		Världsomspännande datornätverk
ISO	International Organisation of Standardization	Organisation som arbetar med standardisering i allmänhet
IST	Information Society Technologies	Organisation som administrerar forskning inom Europeiska Unionen
Java		Objektorienterat plattformsoberoende programmeringsspråk
JavaScript		Skriptspråk utvecklat av Netscape för användning på webbsidor, viss syntaktisk likhet med Java
JPEG	Joint Photographic Experts Group	Rastergrafikformat som oftast används med förstörande komprimering
JPEG2000	Joint Photographic Experts Group 2000	Rastergrafikformat som oftast används med förstörande komprimering
JScript		Microsofts motsvarighet till JavaScript
kartdata		Grafisk framställning (visualisering) av geografiska data
Namespace		Standard för möjliggörande av användning av flera XML-tillämpningar i samma XML-dokument
OGC	Open GIS Consortium	Organisation som genom standardisering verkar för underlättande av utbyte av geografiska data
OpenLS	OpenGIS Location Services	Samling av standarder från OGC för lägesbundna tjänster för mobila användare
PDA	Personal Digital Assistant	Handdator
PNG	Portable Network Graphics	Rastergrafikformat med ickeförstörande komprimering
SGML	Standard Generalized Markup Language	Standard för uppmärkning och strukturering av information med hjälp av taggar
SLD	Styled Layer Descriptor	Standard för styrning av utseendet hos de kartbilder som renderas av en WMS-server
SRS	Spatial Reference System	Parameter i WMS-förfrågan för angivande av koordinatreferenssystem
SWF	Shock Wave Flash	Vektorgrafikformat utvecklat av Macromedia
SVG	Scalable Vector Graphics	XML-tillämpning från W3C för uppmärkning av vektordata med inriktning mot presentation