Examensarbete i geografisk informationsteknik nr 12

Visning av geotekniska provborrningar i en webbmiljö

Nariman Emamian

Civilingenjörsutbildningen i Lantmäteri
Lunds Tekniska Högskola

Institutionen för Naturgeografi och Ekosystemvetenskap
Lunds Universitet

LUNDS UNIVERSITET
Lunds Tekniska Högskola

Visning av geotekniska provborrningar i en webbmiljö

EXTM05 Masteruppsats, 30hp

Författare:
Nariman Emamian

Handledare:
Lars Harrie, Institutionen för Naturgeografi och Ekosystemvetenskaper
Karin Neland, SWECO Position Malmö
Peter Rothstein, SWECO Position Malmö

Januari 24, 2014

Opponent: Niklas Krave

Examinator: Andreas Persson

Institutionen för Naturgeografi och Ekosystemvetenskaper
Lunds Universitet
Sölvegatan 12
223 62 Lund, Sweden

Telefon: 046-222 30 30
Fax: 046-222 03 21
Hemsida: http://www.nateko.lu.se

Examensarbete i geografisk informationsteknik nr 12
Tryckt av E-huset, 2014

Förord

Detta examensarbete är ett avslutande moment i Civilingenjörsutbildningen Lantmäteri med inriktning mot geografisk informationsteknik. Arbetet är utfört på institutionen för naturgeografi och ekosystemvetenskap vid Lunds Universitet i samarbete med Sweco Position i Malmö.

Först och främst vill jag rikta ett tack till min handledare på universitet, Lars Harrie, som hela tiden varit ett stort stöd i både tekniska och akademiska frågor. Ett tack riktas även till mina handledare på Sweco Position, Karin Neland och Peter Rothstein, utan dem hade arbetet inte varit genomförbart. Möjligheten att få vara hos Sweco Position och skriva arbetet har varit en ovärderlig erfarenhet.

Ett sista tack riktas till alla jag varit i kontakt med på Sweco Position i Malmö. Mottagandet har varit föredömligt och många har, trots sin informella relation till mig, gett svar på frågor och funderingar som dykt upp under vägens gång.

2013-12-12
Nariman Emamian

Abstract

It's important to have knowledge about the existing conditions before proceeding with various types of construction works. Geotechnical investigations are made to determine the soil properties and one method of doing so is to perform borehole drillings. Such drillings are expensive and unfortunately it's not always known if similar procedures have been performed in the surroundings. A geotechnical drilling is interpreted by a geologist that clearly would benefit from knowing the results from previous drillings in the area. Thus, there is a common interest in centralized storage of such data.

Sweco GeoAtlas is a GIS software for managing geotechnical data which also enables advanced geological analysis. Its usage requires installation in a desktop environment and rather good acquaintance of the system. This master thesis use Sweco GeoAtlas as a case study to investigate how presentation of geotechnical data can be performed on the web.

Existing mapping services for managing geotechnical data as well as other services for displaying quantitative point data have been studied to determine their strengths and weaknesses. *The Geological Survey of Sweden* has a newly launched application with relevant purposes but it's still in an early stage. *The Geology of Britain Viewer* is a good example of how a web application for geotechnical data can look like.

The requirement specification for the web application contains features stretching from basic point interaction to the implementation of 3D functionality. All of the requirements haven't been fulfilled due to various reasons but they are all brought up for discussion.

The starting point for the prototype in the case study was to solve the problem with displaying quantitative data. Because of this, the web mapping library *Leaflet* was chosen since it has a well-functioning cluster plugin. Other choices of technology consist of Open Street Map as base map and a dynamic SVG for displaying stratigraphic profile images.

The conclusion of the thesis is that it's possible to move basic interaction with geotechnical borehole drillings to a web environment. Rather than a complete web-GIS the application would instead be an environment to view the data. The target audience of the web application are GeoAtlas users that aren't experts and therefore this could be considered as a reasonable division.

Sammanfattning

Vid olika typer av nyanläggning är det viktigt att ha kännedom om vilka förutsättningar som finns. För att bestämma markens egenskaper utförs geotekniska undersökningar och inom ramen för dessa utförs provborrningar. En provborrning är ett dyrt moment och det är inte alltid känt vilka andra liknande åtgärder som utförts i området. En geoteknisk provborrning tolkas av geologer som kan ha stor nytta av att veta hur tidigare provborrningar sett ut. Det finns således en allmännytta med samlagring av sådan information.

Sweco GeoAtlas är ett desktop GIS bl.a. för hantering av geoteknisk data och som möjliggör avancerade geologiska analyser. Användning kräver installation och god kännedom om programmet. Detta examensarbete använder *Sweco GeoAtlas* som fallstudie för att undersöka hur visning av geoteknisk data kan se ut i en webbmiljö.

Befintliga karttjänster för hantering av geoteknisk data samt tjänster som visar övrig kvantitativ punktdata har studerats för att hämta styrkor och svagheter. *Sveriges Geologiska Undersökningar* har en nystartad kartvisare med relevant ändamål men som är i ett primitivt stadie. *The Geology of Britain Viewer* är ett bra exempel på hur en webblösning för geoteknisk data kan se ut.

Webbapplikationens kravspecifikation innehåller allt från grundläggande punktinteraktion till funktionalitet kopplat till en geologisk 3D-modell. Samtliga punkter i kravspecifikationen har inte uppfyllts, de som blivit lidande av olika anledningar tas dock upp för diskussion.

Utgångspunkten för prototypen i fallstudien var att lösa problemet med visning av kvantitativ data. Teknikvalet för karttjänsten föll därför på öppen källkod-alternativet *Leaflet* som har en väl utformad tilläggsmodul för klusterbildning av punktobjekt. Övriga teknikval inkluderar bakgrundskarta i *Open Street Map* samt visning av stratigrafisk profildata med hjälp av en dynamisk SVG-figur.

Slutsatsen är att det går att flytta visning och grundläggande interaktion av geotekniska provborrningar till en webbmiljö. Snarare än ett GIS i webbmiljö fungerar då applikationen som en datavisningsmiljö. Målgruppen för webbapplikationen är *GeoAtlas*-användare som är icke-experter och därmed kan detta anses vara en rimlig uppdelning.

Förkortningar

AJAX	*Asynchronous JavaScript and XML*: Ett samlingsbegrepp för en grupp tekniker som kan användas för att bygga webbapplikationer.
API	*Application Programming Interface*: Ett bibliotek som tillåter utvecklare att knyta ihop redan beprövade och kvalitetssäkrade funktioner till en helhet.
BGS	*British Geological Survey*: Storbritanniens motsvarighet på SGU.
CSS	*Cascading Style Sheets*: Stilmallar för HTML-dokuments utseende
DOM	*Document Object Model*: Ett objektbaserat gränssnitt för åtkomst och dynamisk uppdatering av ett dokuments innehåll.
JSON	*JavaScript Object Notation*: Ett utbytesformat nära anknutet till skriptspråket JavaScript.
GIS	Geografiska informationssystem
GML	*Geography Markup Language*: En XML-dialekt för att uttrycka geografisk information.
HTML	*Hypertext Markup Language*: Märkningsspråket som webbsidor generellt är uttryckta i.
HTTP	*Hypertext Transfer Protocol*: Standardprotokoll för överföring av webbsidor på Internet.
ISO	*International Organization for Standardization*: En internationell standardiseringsorganisation.
OGC	*Open Geospatial Consortium*: En standardiseringsorganisation för geografisk data.
OSM	*Open Street Map*: En ideellt framtagen (väg-) karta.
PHP	*Hypertext Preprocessor*: Ett populärt skriptspråk för serverprogrammering.
SGU	Sveriges Geologiska Undersökning: Myndighet vars uppdrag är att undersöka, dokumentera och informera om Sveriges geologi.
SVG	*Scalable Vector Graphics*: Ett XML-baserat bildformat med vektoruppbyggnad.
SQL	*Structured Query Language*: Ett programspråk för kommunikation med relationsdatabaser.

W3C	*World Wide Web Consortium*: Ett industrikonsortium som arbetar med utveckling av tekniska protokoll, standarder och programvara för webben.
W3DS	*Web 3D Service*: Ett protokoll för utbyte av tredimensionell geodata.
WebGL	*Web Graphics Library*: Ett API för rendering av 3D-grafik i webbmiljöer.
WMS	*Web Map Service*: Ett protokoll för visning av kartbilder.
WFS	*Web Feature Service*: Ett protokoll för hämtning, visning och redigering av geografiska företeelser.
X3D	*Extensible 3D*: En XML-dialekt för geometrier i 3D.
XML	*Extensible Markup Language*: Ett märkningsspråk som är anpassat för både maskinell och mänsklig avläsning.

Innehållsförteckning

1. Inledning 1
   1.1. Bakgrund 1
   1.2. Problemformulering 2
   1.3. Syfte 2
   1.4. Metod 3
   1.5. Avgränsningar 3
   1.6. Rapportutformning 3
2. Geotekniska undersökningar 3
3. Visualisering av geotekniska undersökningar 5
   3.1. Sveriges Geologiska Undersökningars kartvisare 5
   3.2. The Geology of Britain viewer 6
4. Systemarkitektur 10
5. Tekniker 10
   5.1. Lagring 10
   5.2. Webbstandarder 11
   5.3. Utbytesformat 20
   5.4. Systemarkitektur Webbapplikation 26
6. Produkter 27
   6.1. Databaser 27
   6.2. Kartpubliceringsverktyg 30
   6.3. Bakgrundskartor 30
   6.4. Exempel från applikationsgränssnitt och färdiga tjänster 33
7. Visualisering av kvantitativa punktdata 36
   7.1. Quadtree 36
   7.2. Multiplicatively Weighted Voronoi Diagram (MWVD) 37
   7.3. Flightradar24 38
   7.4. Marine Traffic 39
   7.5. San Francisco Crimespotting 39
8. Fallstudie – Sweco GeoAtlas 40
   8.1. GeoAtlas idag 40
   8.2. Kravspecifikation 43

8.3. Teknikval 45
8.4. Placering av information i skärmen 48
8.5. Prototyp 50
9. Diskussion 55
9.1. HTML5 55
9.2. Leaflet/OpenLayers 55
9.3. Kluster 55
9.4. Brister och buggar 56
9.5. Tidskomplexitet 56
9.6. Metadatavisning 58
9.7. Utskriftsvänlig sida med Canvas 58
9.8. Stratigrafisk profilskiss – algoritm och buggar 58
9.9. Mobilkompabilitet 60
9.10. Öppen källkod 60
10. Slutsatser 60
Referenser 62

1. Inledning

1.1. Bakgrund

Vid anläggande av exempelvis nya vägar och hus är det viktigt att veta hur de geotekniska förutsättningarna ser ut. Markens egenskaper kan vara avgörande för vilka byggmetoder som ska användas och kan även resultera i att ett bygge inte genomförs. Byggprojekt är således beroende av geotekniska undersökningar där man fastställer jordens byggnadstekniska egenskaper. En sådan undersökning kan beröra många olika parametrar men i sin helhet vill man veta hur marken kommer påverka den tänkta konstruktionens grund. För att få svar på sådana frågor måste man tolka resultatet från en geoteknisk undersökning.

Inom ramen för en undersökning görs det provborrningar i marken. Resultatet av en sådan borrning är svårtolkat för en lekman. Förfarandet i verkligheten är att t.ex. ett konsultföretag utför borrningar och tolkar dessa åt beställaren. Beställaren (som ibland kan vara kommunen) tar då del av rapporter innehållandes borrprotokoll och tolkningar, vilka är den enda informationen som lagras hos dem. Geotekniska rådata stannar oftast hos konsultföretagen, d.v.s. utföraren.

I en kommun är det flera förvaltningar som kan vara beställare i geotekniska ärenden. Exempelvis kan fastighetskontoret beställa en geoteknisk undersökning på en tomt i husbyggnadssyfte. I ett idealfall har då gatukontoret kännedom om detta och utnyttjar informationen då en väg ska anläggas i närheten. Så sker ofta inte i praktiken, vanligt är att varje förvaltning beställer nya undersökningar i samband med nya projekt.

Det är svårt att uppskatta hur många geotekniska borrningar som utförts i Sverige. I Malmö stad rör det sig om mer än 35 000 borrningar för olika syften (anläggande av grundvattenbrunnar medräknat) sedan mitten av 1900-talet. Med kostnader runt 30 000kr för en geoteknisk provborrning är det enkelt att motivera någon form av resurssparande datahantering för att nyttja tidigare undersökningar.

I Malmö sparas resultat från geotekniska undersökningar i ett geografiskt informationssystem (GIS) i form av en geologisk atlas. Arbetet med en sådan började redan på 80-talet och tanken var då att, likt en vägatlas, publicera en bok med information om befintliga provborrningar och geologiska tolkningar. En sådan bok har aldrig publicerats och på 90-talet övergick projektet till digital form genom den Sweco-tillverkade mjukvaran *GeoAtlas*.

Skillnaden mot traditionsenlig behandling av geoinformation är att man i *GeoAtlas*-projektet lagrat geotekniska rådata om varje provborrning och lagrat dem i ett enda system. En beställare på kommunen kan då, innan en ny geoteknisk undersökning beställs, gå in och se

vilken data som redan finns tillgänglig för analys. Med hjälp av denna samordning har mängden punkter blivit så omfattande att man kan konstruera en geologisk 3D-modell över Malmö stad.

1.2. Problemformulering

När förutsättningarna kring ett nytt bygge skall fastställas är det fördelaktigt att veta vilken data som finns i omgivningen. Sådan vetskap kan förbättra de tolkningar som åligger en geolog att utföra. Om inte delning av geoteknisk information sker på ett bra sätt kan det medföra slöseri av resurser. För att ge en överblick över vilka inmätningar som gjorts krävs samordning åtminstone i form av en punktdatabas. Vidare kan geologiska 3D-modeller vara ett bra verktyg för mer avancerad och interaktiv visning. Den typen av funktionalitet är i stort sett begränsad till lokala datormiljöer (engelska: *desktop environment*) men problemet är att mjukvara som stödjer sådan visning ofta är kostsam och data blir därför oåtkomligt för en tredjepart. En webbapplikation är därför ett bra medium för att göra geoteknisk data mer lättillgängligt. En annan fördel är att en webbapplikation kan utformas så att den även kan användas ute i fält.

I ett *desktop-GIS* finns generellt många olika analys- och exportmöjligheter. Att lyfta ut funktionalitet från en sådan till en webbapplikation medför utmaningar bl.a. i form av val av teknik. Utifrån en kravspecifikation måste teknikval göras så alla funktioner är möjliga att implementera. Samtidigt måste hänsyn tas till applikationens målgrupp och användningsområde så att den inte blir onödigt avancerad för ändamålet.

De teknikval som måste göras berör bl.a. lagring, distribuering och visualisering av data. Webbutvecklingens frammarsch har på senare år möjliggjort tredimensionell visualisering utanför lokala datormiljöer. Dessvärre finns idag få exempel där detta utnyttjas i samband med GIS och användning av tekniker så som WebGL har generellt ännu inte tagit fart.

1.3. Syfte

En metod för visualisering av geoteknisk data i webbmiljö ska tas fram med *Sweco GeoAtlas* som fallstudie. Syftet kan delas upp i följande delar:

1. Ta fram förslag på lämplig grafisk utformning till en webbapplikation ämnad för visualisering av data från geotekniska undersökningar.
2. Studera vilka tekniker i form av webstandarder och utbytesformat som stödjer den valda designen.
3. Utifrån en kravspecifikation göra teknikval som möjliggör fullständig implementering av kraven.
4. Skapa en prototyp i form av en applikation för visning av geoteknisk data med Sweco GeoAtlas som grund.

1.4. Metod

Utifrån formulerade delsyften är metoden för detta projekt följande:

1. Skapa en idé över hur applikationen bör utformas rent estetiskt, 2D till en början men även en 3D-aspekt skall tas hänsyn till. Efterforskning sker för att jämföra befintliga exempel likväl som exempel från andra områden. Den utarbetade designen redovisas antingen i ett ritprogram eller som en skiss på ett papper.
2. Inom kategorierna lagring, distribution och visualisering av geografisk data undersöks olika teknikalternativ.
3. Utifrån de bäst lämpade teknikvalen skapas en prototyp som ska ha ingående komponenter med öppen källkod-licens.

1.5. Avgränsningar

En geoteknisk provborrning kan ske för olika syften, t.ex. för att utvinna vätska eller gas. I detta arbete behandlas dock endast de borrningar som är knutna till geotekniska undersökningar, d.v.s. sådana som syftar till att bestämma markens fysiska egenskaper i samband med olika bygg- och infrastrukturprojekt.

Innan projektstart är det bestämt att alla verktyg som används i fallstudien måste ha öppen källkod-licens. Detta är dels en kostnadsfråga men även en nödvändighet för att experiment i detta projekt ska kunna återskapas i efterhand.

1.6. Rapportutformning

Rapporten börjar med en allmän beskrivning om geotekniska undersökningar och geologiska modeller i kapitel 2. I kapitel 3 görs en omvärldsstudie om hur man löst visualisering av geotekniska undersökningar i befintliga applikationer. Systemarkitektur i enkla kartapplikationer tas upp i kapitel 4 följt av de ingående tekniker som kan tänkas användas, vilka redogörs för i kapitel 5. Exempel på GIS-produkter och applikationsgränssnitt tas upp i kapitel 6. I kapitel 7 studeras exempel där problemet med kvantitativa punktdata finns innan en fallstudie baserad på Sweco GeoAtlas påbörjas i kapitel 8.

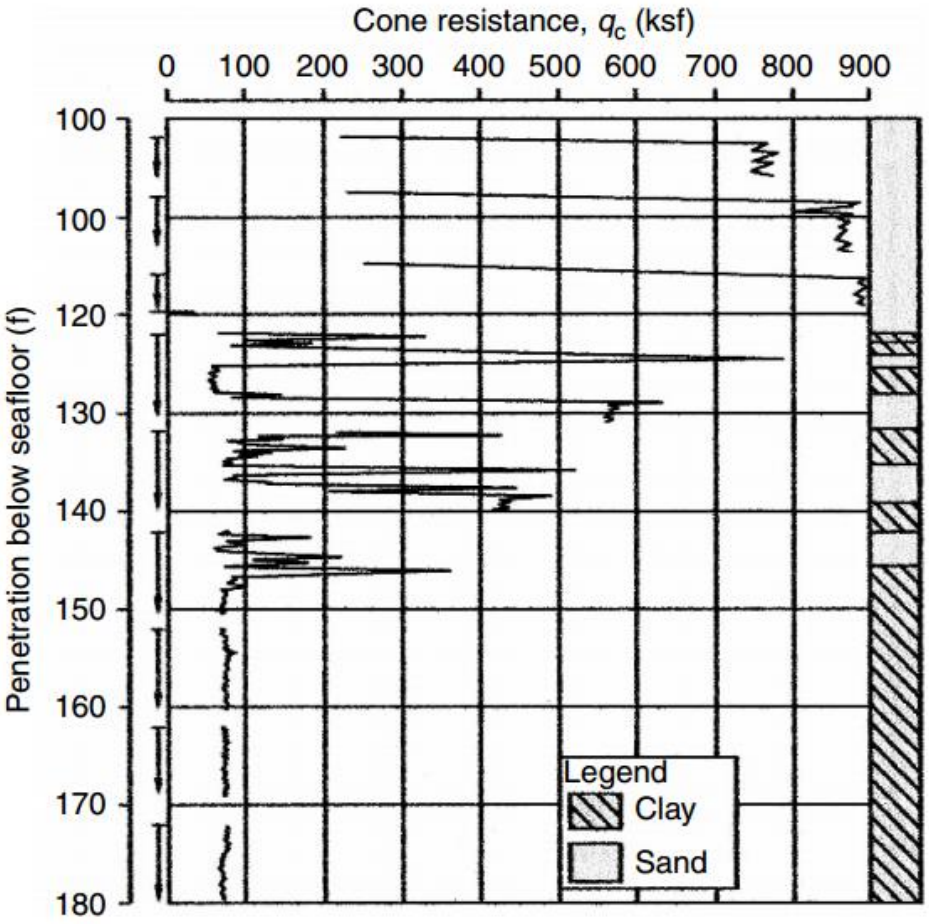
2. Geotekniska undersökningar

Vid konstruktion av nya byggnader, broar etc. måste geotekniska undersökningar utföras för att säkerställa markens fysiska egenskaper. Detta är en av byggprocessens viktigaste faser och ligger till grund för hur ett projekt skall fortsätta. Egenskaperna bestäms med hög noggrannhet lokalt kring byggarbetsplatsen och med en geologs hjälp skaffar man sig även en regional överblick (Hunt 2006). Mer specifikt är målet med olika undersökningar att:

- Definiera omkringliggande fördelning av jordarter och berggrund kring den föreslagna byggnadsåtgärden.

- Definiera grundvattenförhållanden.
- Upptäcka geologiska risker, så som sättningar i marken.
- Ta prover av geologiska material för att sedan mäta de tekniska egenskaperna i laboratoriemiljö.
- Utföra tester på plats för att mäta de tekniska egenskaperna.

För att, på plats, testa markens egenskaper kan man utföra ett s.k. penetrationstest. En stav slås då ner i marken med ett bestämt tryck och motståndet registreras. Mätresultaten loggas och redovisas bl.a. i diagram (som t.ex. i figur 2.1 nedan) och utifrån det kan man uppskatta vilka material som penetrerats (Hunt 2006).

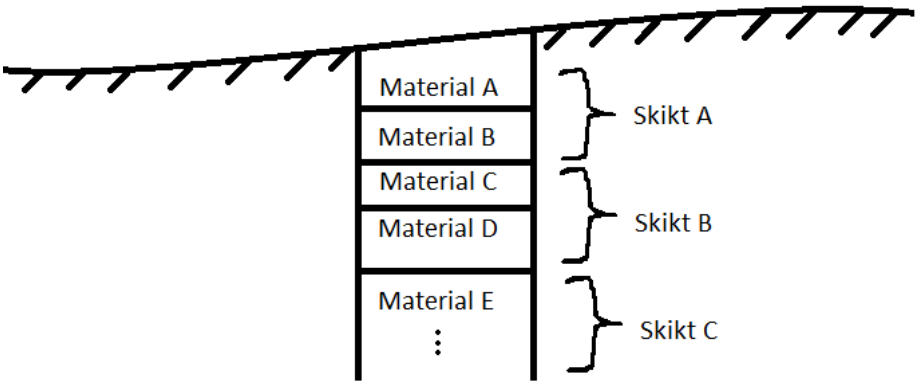


Figur 2.1: Diagram över motståndet i ett CPT (*Cone Penetration Test*) (Hunt 2006 s.65).

Borrningsresultaten sammanställs av geotekniker och ligger till grund för en geologs tolkning av informationen. Tidigare har sådan information redovisats i tvärsnitt, alltså i 2D, men det blir allt mer vanligt med 3D-visualisering.

För att kunna tillverka en geologisk 3D-karta krävs en konceptuell lokal geologisk modell. Geologer tar fram en, för området, gällande modell som innefattar generaliseringar av de ingående jordlagren. I figur 2.2 nedan syns hur en sådan modell kan vara uppbyggd: de

geotekniska lagren beskriver material som lera, morän, sand medan den geologiska modellen innehåller generaliseringar av lager som fyllning och övermorän.



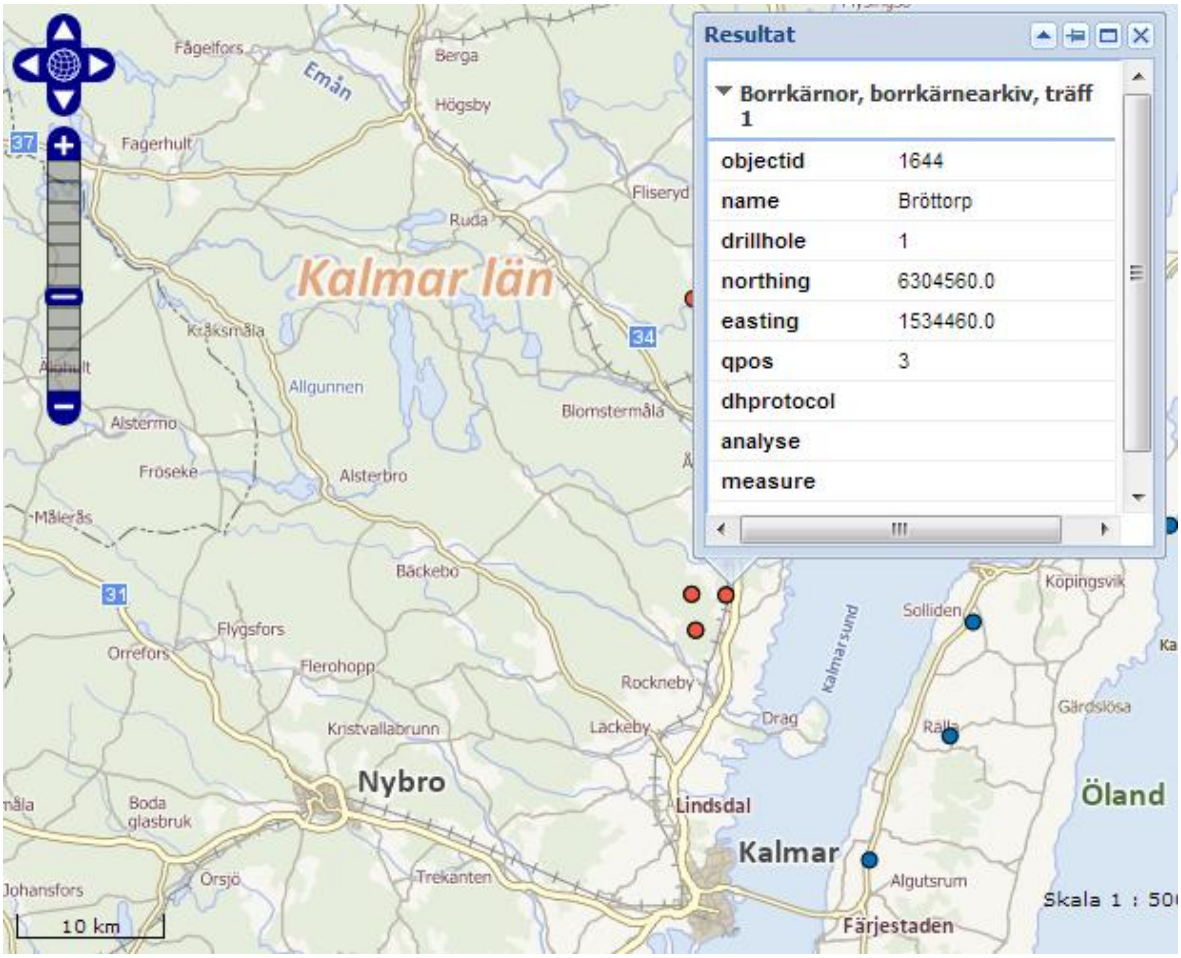
Figur 1.2: Material från en provborrning (A-E) generaliseras till en geologisk modell (A-C).

3. Visualisering av geotekniska undersökningar

Tidigare har visualisering av geotekniska undersökningar endast skett i lokala datormiljöer. Det är först på senare år som tekniken möjliggjort bra presentationer i en webbmiljö. Nedan följer ett par exempel på sådana webbapplikationer.

3.1. Sveriges Geologiska Undersökningars kartvisare

I Sverige har arbetet med en nationell karttjänst för visualisering av borrhål precis börjat. Det är Sveriges Geologiska Undersökning (SGU) som driver den, men täckningen av data är fortfarande gles nationellt. Själva applikationen är enkelt uppbyggd och den enda funktionaliteten är visning av attribut knutna till borrningarna enligt figur 3.1 nedan. Informationen är knapphändig och någon länkningsmöjlighet till rapporter etc. finns inte.

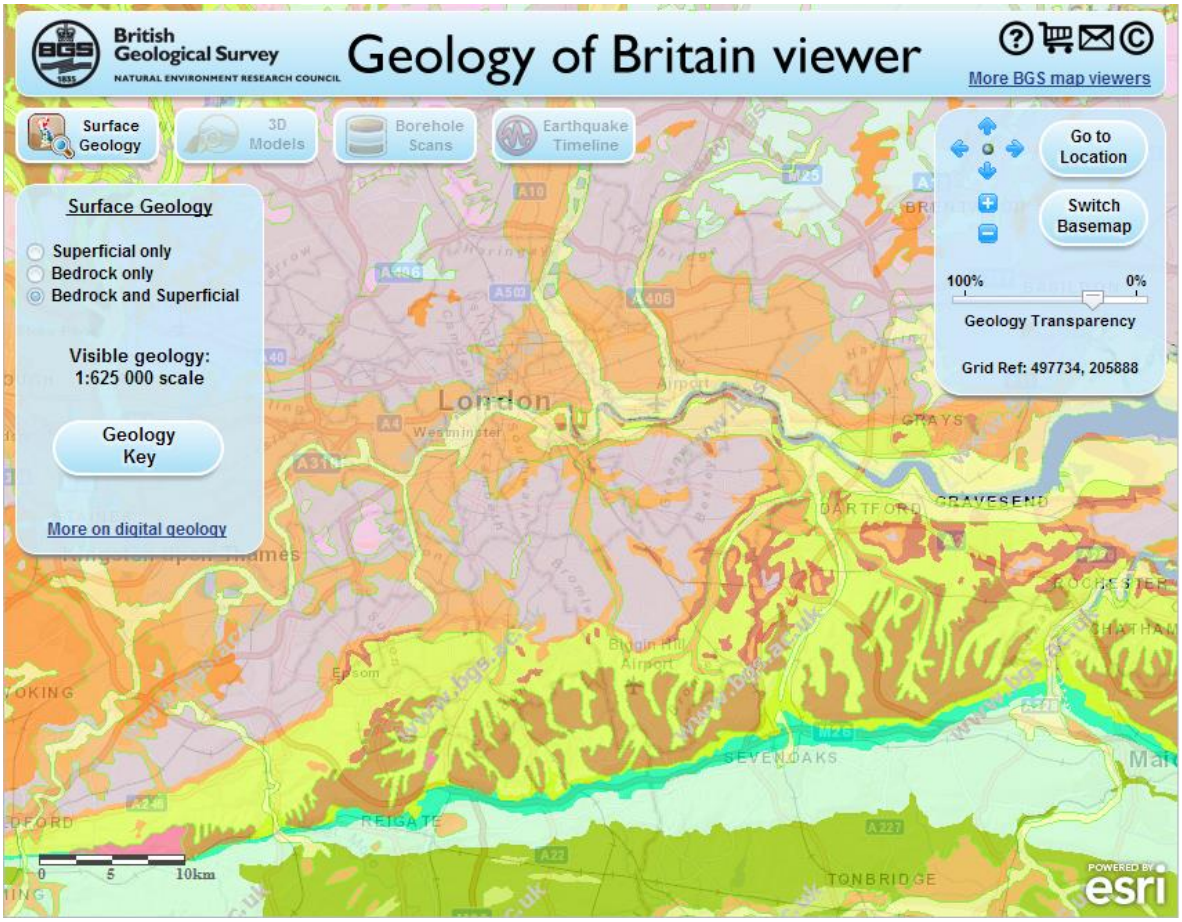


Figur 3.1: SGU:s kartvisare för geotekniska provborrningar. (SGU 2013).

3.2. The Geology of Britain viewer

I Storbritannien har man kommit långt på denna front. *British Geological Survey (*BGS)förvaltar *The Geology of Britain viewer* som är en väl genomarbetad karttjänst som finns till allmänhetens förfogande. Figurerna som följer i detta avsnitt visar hur man på ett bra sätt kan visualisera viktiga geologiska data.

Standardvyn i tjänsten är en geologisk 2D-modell över Storbritannien (se figur 3.2). Valmöjlighet finns att visa endast jordarter, endast berggrunden eller både och. Geologins transparens kan regleras och en genomgående teckenförklaring öppnas när användaren trycker på knappen *Geology Key.*

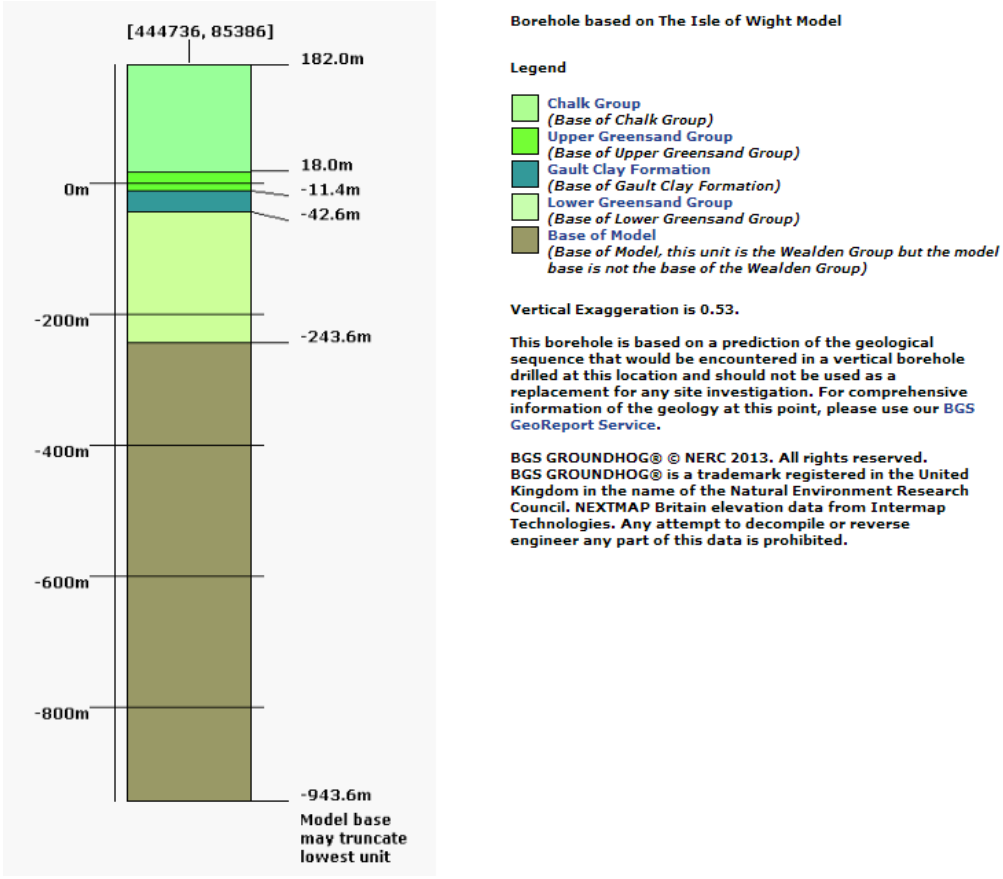


Figur 3.2: Startfönstret i the *Geology of Britain viewer* (BGS 2013)

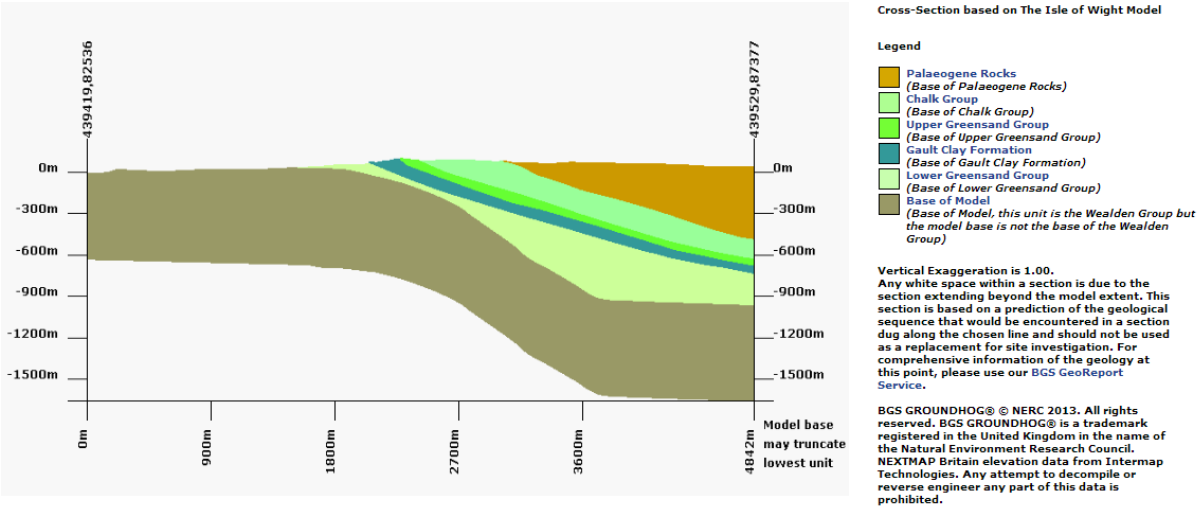
Det som är intressant, sett ur denna studie, är hur geotekniska provborrningar visualiseras. Under menyn *Borehole Scans* kan man se ett punktskikt med alla borrningar som gjorts. De är klassificerade i fyra färger: tre av dem har med borrdjupet att göra och den sista färgen, svart, kategoriserar någon form av hemlighetsstämplad data. När man trycker på en punkt dyker ett poppuppfönster upp med ett par grundläggande attribut samt länk till en rapport kopplat till borrhålet (se figur 3.3).

För fem olika områden i Storbritannien har man dessutom definierat en geologisk 3D-modell och gjort interaktion möjlig. Här uppfylls på något sätt idealfallet för en applikation av detta slag, då man tillåter användaren utföra fiktiva provborrningar och skapa både horisontella och vertikala tvärsnitt för visualisering av jordlager. Exempel på detta syns i figur 3.4–3.6.

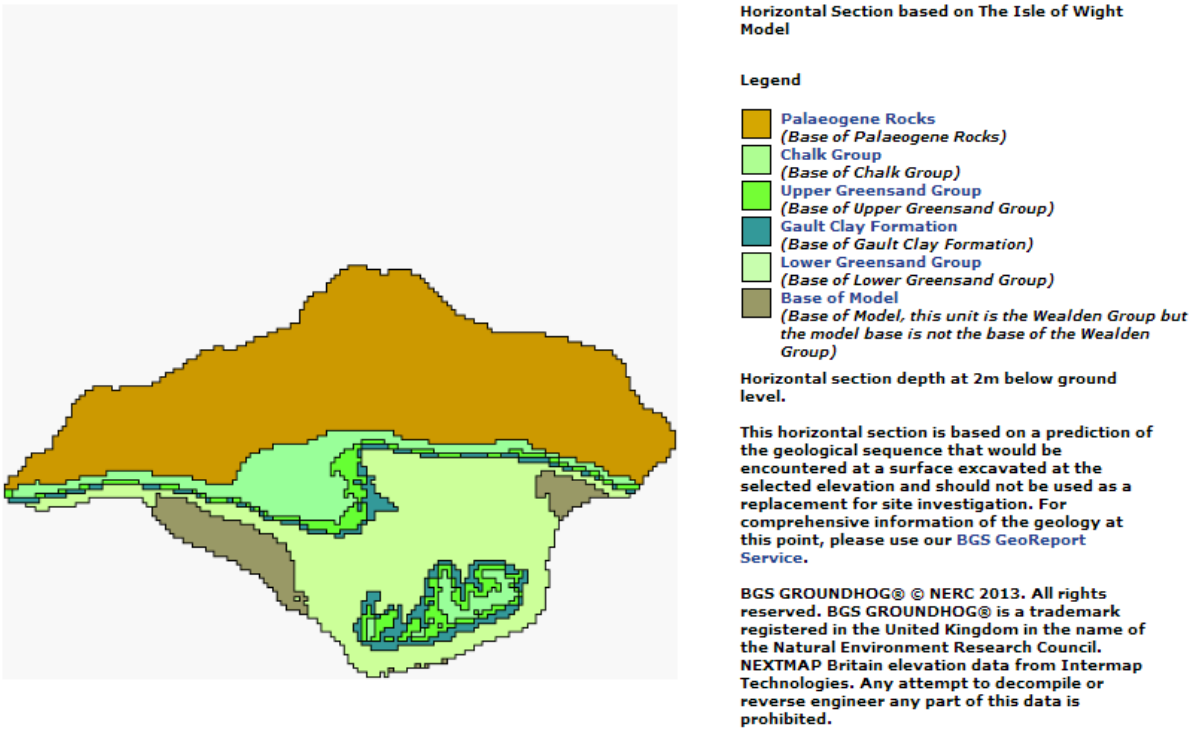
Figur 3.3: Fliken Borehole Scans i *Geology of Britain viewer*. Här kan man visualisera och trycka på olika borrhål. (BGS 2013)



Figur 3.4: En fiktiv provborrning utförd på den geologiska 3D-modellen för Isle of Wright. (BGS 2013)



Figur 3.5: Ett vertikalt tvärsnitt i den geologiska 3D-modellen för Isle of Wright. (BGS 2013)



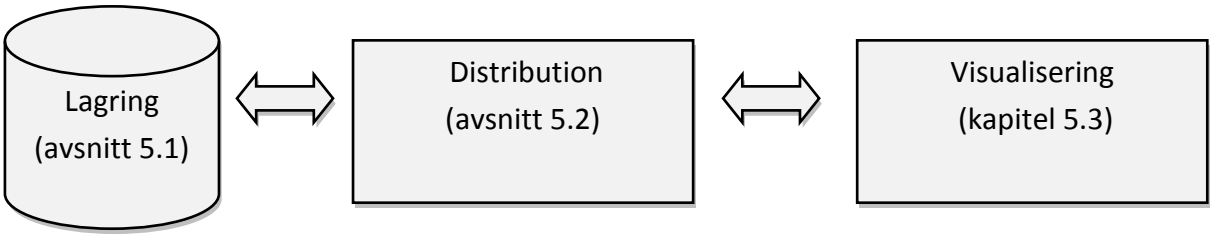
Figur 3.6: Ett horisontellt tvärsnitt i den geologiska 3D-modellen för Isle of Wright. Användaren får möjlighet att bestämma på vilket djup tvärsnittet ska ske (BGS 2013).

Detta exempel är unikt och det är därför intressant att testa om liknande resultat kan uppnås med andra tekniker än den man använt i *The Geology of Britain viewer*. Information om den bakomliggande tekniken är dock knapphändig. I en kortfattad summering på *British Geological Survey*s hemsida framgår det att man använt sig av:

- ArcGIS JavaScript-tillägg till Google Maps API
- ArcGIS Server
- Dojo JavaScript verktygslåda för transperensreglaget och applikationens poppuppfönster.

4. Systemarkitektur

I stora drag består informationsflödet i en enkel karttjänst på webben av tre steg: lagring, distribution och visualisering av data. Som grund krävs någon form av databashanteringssystem som möjliggör interaktion med grunddata, som då både kan vara i databaser eller i filsystem. Nästa steg är att publicera lokalt lagrade data på webben. För att göra detta krävs någon form av serververktyg som är anpassat för geografisk information. När tillgängligheten på webben är löst väljs vilka standarder som skall användas för visualisering. Utöver själva hemsidan och uppbyggnaden av kartapplikationen krävs nämligen ett definierat protokoll för att nå data på servern. I figur 4.1 syns en grov skiss över hur arkitekturen kan se ut i en kartapplikation på webben.



Figur 4.1: Förenklad bild över hur en webbkarta är uppbyggd.

5. Tekniker

I detta kapitel redogörs för ingående komponenter hos en kartapplikation på webben. Viktiga delar är här band annat standarder, utbytesformat och produkter. Teknikerna som tas upp ligger sedan till grund för en diskussion och val av metod i kommande fallstudie.

5.1. Lagring

5.1.1. Filsystem

ESRI:s *shape* är ett exempel där lagring delvis sker i datorns filsystem. Formatet har sina rötter i tidigt 90-tal och släpptes i samband med ArcVIEW GIS 2. En *shape*-fil består av tre obligatoriska filer: en huvudfil (.shp), en indexeringsfil (.shx) och en attributfil (.dbf). Tillsammans utgör de en meningsfull uppbyggnad av geografisk data och hänger ihop sekventiellt, d.v.s. att objekt nummer *N* i följd korresponderar mot samma objekt i de tre delfilerna (ESRI 1998).

Shape-formatet är ett exempel på hybrid lagringsstruktur, d.v.s. en kombination mellan lagring i filsystem och databas. Man kan lagra obegränsad mängd objekt, men de kommer alla hamna i samma dbf-fil som innehåller en relationsdatabastabell. Varje objekt har ett 1-1 förhållande till de geometrier som shp-filen innehåller. En nackdel med shape-formatet är inte flera personer kan arbeta med filerna samtidigt. För att göra det på ett säkert sätt måste någon form av databashanteringssystem användas.

5.1.2. Databasmodeller

Att hantera stor mängd information är ett problem som snabbt skenar iväg om man inte använder sig av någon form av struktur. Idén med databaser har funnits länge och redan på 60-talet, i samband med NASA:s Apolloprogram, finns exempel på det som anses vara en av de första databastyperna: Hierarkiska databaser. De kan närmast liknas vid en trädstruktur där man får följa olika noder för att hitta till rätt data (Rob & Coronel 2009). Ett annat tidigt exempel är relationsdatabaser, vilket även är ett av de mest använda alternativen idag. Här organiseras information i tabeller där varje kolumn utgör ett attribut och varje rad utgör ett objekt.

Objektorienterade databaser är ett kraftfullt alternativ som kan hantera komplexa datatyper. Här har varje objekt ett unikt id och tillhör en objektklass med definierade attribut och metoder. Geografisk information är en typisk tillämpning där objektorienterade databaser kan användas. Tekniken används dock idag i stort sett endast i nischade sammanhang p.g.a. dess komplexitet.

En kombination av objekt- och relationsdatabaser, kallat objektrelationella databaser har på senare vunnit mer och mer mark inom bl.a. GIS. Grunden är baserad på en relationsdatabas där data lagras i tabeller men det är även möjligt att definiera egna datatyper, så kallade abstrakta datatyper (engelska: *user defined type*) (Harrie & Svensson 2013). Med abstrakta datatyper kan mer komplexa strukturer uppnås och en databas kan t.ex. anpassas till att hantera geometrier. Det är även möjligt att enkapsulera komplexiteten med hjälp av funktioner så att en vanlig användare enkelt kan utnyttja systemet. Objektrelationella databaser med tillägg för geometrier är normalt det som refereras till som spatiala databaser (Urban & Dietrich 2009). Ett exempel på detta är Orace Spatial där enkapsulationen finns i införandet av datatypen *sdo_geometry,* som dels innehåller geometrier men även har definierade spatiala funktioner.

5.2. Webbstandarder

Vikten av standarder i samhället är svår att bestrida. Tankarna förs snabbt mot ISO-standarder men faktum är att alla människor dagligen använder produkter som följer någon form av standard. Vare sig det är en glödlampas utformning eller en hemsidas strukturella villkor så finns de där och påverkar det vi gör. En standard gör det enkelt för olika parter att veta vad som förväntas av en produkt och bidrar till bättre konkurrens då konsumenter får lov att välja glödlampan som är bäst lämpad för det egna hushållet. Produktionskostnader hålls nere eftersom tillverkarna slipper utforma en specifik glödlampa för den svenska marknaden och en helt annan för den danska.

Under slutet av 90-talet pågick det s.k. webbläsarkriget mellan Internet Explorer och Netscape. Som en motsats till dagens standardiseringar pågick det då många försök att

införa webbläsarspecifika funktioner. En följd av detta blev att hemsidor fick en av två olika loggor: *”Best viewed in Internet Explorer”* eller *”Best viewed in Netscape”*. För utvecklare var detta en mardröm och man tvingades anpassa sina kodrader till en av läsarna eller göra två versioner av samma hemsida (Clark m.fl. 2012).

I samma veva startades WaSP (*Web Standards Project*) vars syfte var att uppmuntra användandet av format som rekommenderats av W3C (*World Wide Web Consortium*). Med ovan nämnda argument för standardiseringar nådde man 2001 sitt huvudmål: att övertala Microsoft, Netscape och Opera (m.fl.) att helt och hållet stödja HTML 4.01 / XHTML 1.0, CSS1 och ECMAScript. På så vis strömlinjeformades webbutvecklingen och möjliggjorde den semantiska webbens existens (Web Standards Project 2003).

Idag är det fortfarande problematiskt att anpassa en sida till samtliga webbläsare. Det finns två aspekter att ta hänsyn till: bakåtkompabilitet till gamla webbläsare samt att webbläsare de facto implementerar nya standarder olika snabbt. Bakåtkompabiliteten är t.ex. ett problem när nyare hemsidor har stilar uttryckta i CSS3 (se avsnitt 5.2.2.) eller när moderna HTML5-taggar (som t.ex. video-taggen) används. En del problem kan lösas genom införande av externa JavaScript-bibliotek. För att t.ex. få vissa CSS3-taggar att fungera i gamla webbläsare kan biblioteket *Respond.js* användas (Respond 2013).

5.2.1. HyperText Markup Language - HTML

HTML är det dominerande märkningsspråket inom webbutveckling. Dagens version, HTML4, är ursprungligen designat för märkning av statiska dokument och länkar mellan dem. Språket var tillräcklig för tio år sedan men under 2000-talet växte webbutvecklare snabbt ur HTML4. Hemsidor tog mer dynamiska former och man började blanda in andra tekniker som PHP, JavaScript och Flash för att åstadkomma detta.

Redan i slutet av 90-talet uttryckte W3C att framtiden inom webbutveckling låg i XHTML, vilket kan beskrivas som HTML med XML-syntax. XHTML 1.0 mötte inte mycket motstånd men innehöll samtidigt inga stora förändringar. XHTML 2.0 gick det däremot sämre för. En del nya funktioner introducerades men bakåtkompabilitet saknades. Märkningsspråket svarade inte mot det arbete webbutvecklare utför i praktiken utan bestod snarare av W3C:s ideal. Som en följd grundades en rörelse som skulle verka för en bättre framtid på webben, WHATWG (*Web Hypertext Application Technology Working Group*) (Clark m.fl. 2012).

WHATWG bestod av professionella webbutvecklare från Apple, Mozilla och Opera och grundades som ett svar mot HTML:s långsamma utveckling. W3C:s beslut att överge HTML mot XHTML bestreds och man tillsatte en egen arbetsgrupp som skulle arbeta med en HTML-uppföljare, kallad *Web Applications 1.0*. Apple, Mozilla och Opera föreslog år 2007 att W3C skulle använda *Web Applications 1.0* som grund för kommande HTML5. Förslaget accepterades en månad senare och blev då officiellt grunden till HTML5 (Clark m.fl. 2012).

En läxa var lärd och förutom att vara bakåtkompatibel innehåller HTML5 en rad förbättringar och nya objekttyper. Möjlighet finns att integrera ljud, video och andra applikationer utan användandet av externa insticksmoduler. HTML5 har idag bra stöd i alla stora webbläsare och är på så vis inte bunden till en specifik plattform. Med verktyg som t.ex. WebGL (nyhet i HTML5) introduceras avancerad grafik till handhållna enheter på ett enklare sätt. Jämförelsevis har t.ex. Apple sedan tidigare aktivt tagit avstånd från Flash, en vanlig teknik i dynamiska hemsidor.

5.2.2. Cascading Style Sheets 3 – CSS3

CSS fungerar som formatmall för hemsidor och används för att bestämma utseende och placering på ingående komponenter. CSS, eller *Cascading Style Sheet*, har funnits i utveckling sedan 1994 och har länge varit en standard kopplad till HTML. En av dess stora fördelar är att man med CSS kan sätta samma utseende på flera sidor, vilket ofta är önskvärt när en webbplats består av flera hoplänkade sidor.

Metodiken vid webbutveckling är i många fall att alla funktionella och strukturella aspekter implementeras och sedan appliceras en grafisk stil uttryckt i CSS. I takt med standardens utveckling har arbetsflödet underlättats för webbutvecklare mer och mer, tid som tidigare hade behövts spenderas i Photoshop kan nu istället läggas på märkningsspråket och formatmallen med liknande resultat (Clark m.fl. 2012). I CSS3 märks denna skillnad i och med funktioner som rundade kanter, större urval av typsnitt och färgtransparens. Detta innebär t.ex. att en knapp kan utformas helt och hållet med HTML och CSS istället för införandet av en bild som är tillverkad Photoshop. Numera finns även funktionalitet för animationer och gradienter, vilka tidigare krävde JavaScript för att uppnå. Nedan följer ett exempel på hur en enkel stil kan byggas upp med CSS.

Uppbyggnaden av en stil sker i ett eget block innanför <*head*>-taggen:

```
<style>
        Väljare     { Egenskap: värde;}
</style>
```

Om man till exempel vill sätta en bakgrund med rundande kanter för en rubrik-tagg, kan det se ut enligt följande:

```
<style>
        h1 {
                background: red;
                border-radius: 20px;
            }
</stvle>
```

Exempel 5.1: En enkel stil definierad för HTML-taggen h1. Innehållet kommer få röd bakgrund och rundade kanter med radien 20 pixlar.

5.2.3. JavaScript och JQuery

JavaScript var från början utvecklat av Netscape som ett scriptspråk till för att knyta ihop olika komponenter i deras webbläsare. Året var 1995 och språket fick snabbt spridning, inom ett år var det exponerat genom s.k. omvänd ingenjörskonst (engelska: *Reversed engineering*, vilket ungefär innebär att tekniska egenskaper hos en produkt blottas genom analys eller isärplockning). Den fria versionen fick ett eget namn, Jscript, och hade stöd i alla stora webbläsare på den tiden. Olika nischer av språket började dyka upp och på Netscape ansåg man att språket började bli mer och mer ”smutsigt”. Som en följd av detta lämnade man in en ansökan till ECMA (*European Computer Manufacturers Association*) varpå språket JavaScript fastslogs som en standard. Tillväxten tilltog hädanefter ordentligt och ECMA fortsatte leverera uppdateringar till språket. Den viktigaste i ledet anses vara införandet av XMLHttpRequest-objektet (XHR) vilket lade grund för AJAX (som kort sagt är en samlingsterm för användandet av bl.a. DOM, HTML, CSS och XML tillsammans inom webbutveckling) (Lee m.fl. 2012).

Vanliga sätt att använda JavaScript på är t.ex. att hålla reda på innehållet i formulär samt ändra innehåll utan att uppdatera en webbsida. Nedan syns exempel på det sistnämnda.

En webbsida innehållandes JavaScript-kod som befinner sig inom script-taggen.

```
<!DOCTYPE html>
<html>
<body>

<a onclick="HelloWorld()">Start the function</a>

<script>
function HelloWorld() {
                        document.write("Hello, World!");
        }
</script>
</body>
</html>
```

Exempel 5.2: En enkel webbsida innehållandes texten *"Start the function"*. Efter att ha klickat på texten körs JavaScript-funktionen *HelloWorld()* som ändrar texten i webbsidan till *"Hello, World!"*.

I takt med ökad komplexitet och popularitet har även behovet av ramverk ökat. JQuery är ett bibliotek med fördefinierad JavaScript-kod för vanliga funktioner. Dess användning förenklar utvecklingsprocessen för en applikation och minskar antalet felkällor. Bibliotekstänket bakom JQuery är densamma som vissa exempel som tas upp i avsnitt 6.4., skillnaden är att det i de exemplen rör sig om mer komplicerade och tillämpade GIS-bibliotek. JavaScript används även i samband med *Canvas* (se avsnitt 5.2.4.), vilket kommer användas senare i fallstudien.

5.2.4. Canvas

Ordet *canvas* för tankarna till tavlor där betydelsen är en uppspänd duk. Liknelser finns inom webbutveckling där man kan tänka sig att *Canvas* utgör en genomskinlig uppspänd duk över en bestämd yta i webbsidan. Hur ett enkelt element skapas visas i exempel 5.3, storleken är där definierad och skulle i annat fall sättas till standardvärdet 300 x 150 pixlar. Dess position beror på placeringen i HTML-koden men kan, precis som andra HTML-element, flyttas runt med hjälp av CSS (Hawkes 2011).

Ett canvas-objekt skapas med hjälp av *Canvas*-taggen, där man även kan bestämma dess storlek.

```
<canvas width="400" height="400">
        <!-- Lägg till bakåtkompatibel kod här -->
</canvas>
```

Exempel 5.3: Ett enkelt *Canvas*-objekt. Inuti *Canvas*-taggen anges eventuell bakåtkompatibel kod, vilket exempelvis kan bestå av en textsträng som låter användaren veta att hans webbläsare ej stödjer HTML5.

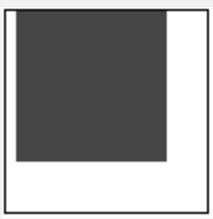
I *Canvas*-taggen kan vissa saker bestämmas, så som storlek och kantlinjefyllning. Vidare åtgärder sker dock inte på taggen. Dess innehåll hämtas istället hjälp av JavaScript för att sedan kunna bearbetas ytterligare (se exempel 5.4 nedan).

En webbsida innehållandes ett *Canvas*-objekt. Det som händer i JavaScript-koden är följande:

- Variabeln *canvas* pekar på *Canvas*-taggen.
- Variabeln *context* hämtar innehållet i *Canvas*-taggen.
- *fillStyle* sätter färgen och *fillRect* fyller i en rektangel på bildkoordinaten (*x,y*)=(5,0) med höjden och bredden 75 x 75 pixlar.

```
<!DOCTYPE html>
<html>
<body>
        <canvas id="myCanvas" width="100" height="100" style="border:1px solid
        #000000;">
                Your browser does not support the HTML5 canvas tag.
        </canvas>
        <script>
                var canvas=document.getElementById("myCanvas");
                var context=canvas.getContext("2d");
                context.fillStyle="#000000";
                context.fillRect(5,0,75,75);
        </script>
</body>
</html>
```

Resulterande figur:



Exempel 5.4: En webbsida som innehåller ett 100 x 100 pixlar stort *Canvas*-objekt, som i sin tur innehåller en svart kvadrat på 75 x 75 pixlar (modifierad från W3S 2013a).

Härefter är det endast fantasin som sätter gränsen för vad man kan åstadkomma. Ovan exempel riktar sig mot *2d-Canvas* vilket bör vara tillräckligt i fallet med karttjänster. Faktum är att fler och fler 3D-implementeringar, främst i form av spel, dyker upp för HTML5. För att använda sig av *Canvas* i 3D gör man istället *getContext*-funktionen på ”WebGL” istället för ”2d” som i exemplet ovan. Då öppnar man API:n för WebGL vilket möjliggör fortsatt avancerad programmering i 3D (se avsnitt 5.2.6.).

5.2.5. Scalable Vector Graphics - SVG

SVG är ett XML-baserat grafiskt vektorformat för att distribuera och visualisera 2D-data. Formatet underhålls av W3C och är en öppen standard som idag har stöd i många GIS-applikationer. Som namnet antyder sker lagring av vektordata, men formatet kan även distribuera raster och textobjekt. Möjlighet finns därför till att distribuera stora mängder information på ett effektivare sätt än vad t.ex. ett rasterformat som JPEG kan göra (W3C 2004). Fördelar finns bl.a. i lagringsstolekar men ännu viktigare är filens sammansättning. Snarare än en bild är det matematiska uttryck som lagras vilket möjliggör skalning. Inom kartframställning är detta en viktig faktor eftersom man står inför problemet att visa rimliga mängder kartografisk information på en mobiltelefon såväl som på en större datorskärm. (Peng & Zhang 2004). En SVG-fil kan även vara interaktiv och dynamisk, se exempel 5.5 nedan. Funktionalitet finns bland annat för händelsehanterare så som ”när muspekaren ligger över objektet” och ”när objektet klickas på”.

En SVG-fil som beskriver en kvadrat som hela tiden tonar mellan vit och blå färg.

```
<svg>
        <rect x="20" y="20" width="250" height="250" style="fill:blue">
                <animate attributeType="CSS" attributeName="opacity"
                from="1" to="0" dur="5s" repeatCount="indefinite" />
        </rect>
</svg>
```

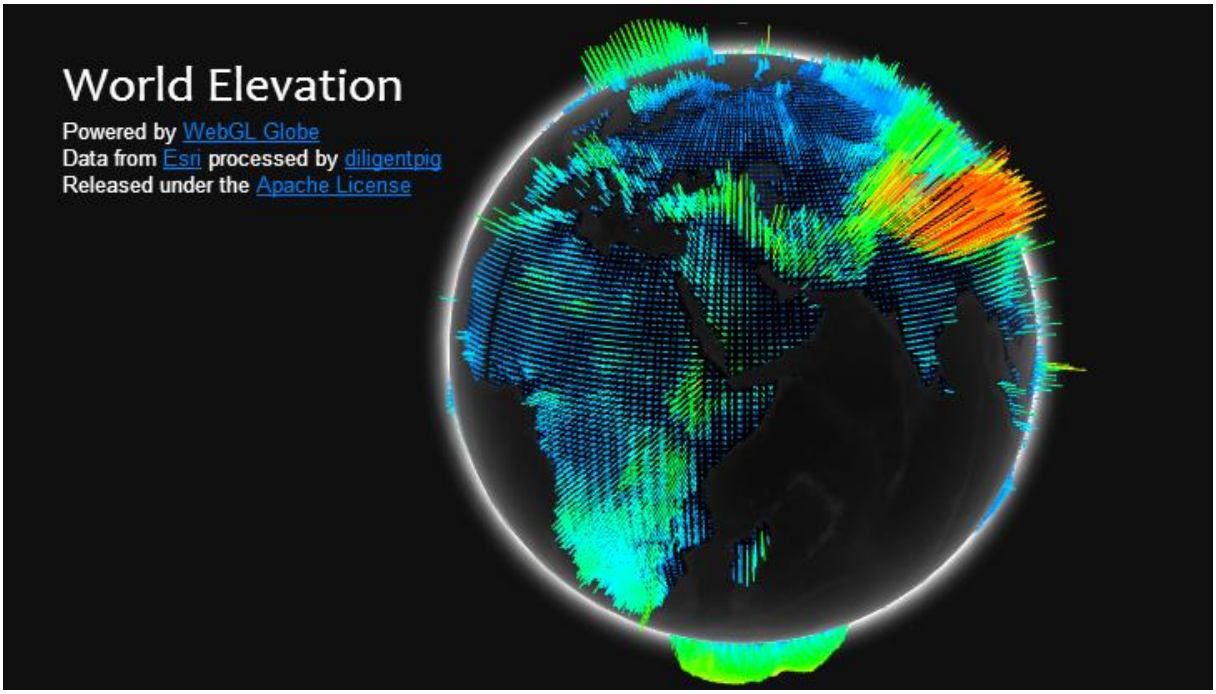
Exempel 5.5: En dynamisk SVG fil. Först skapas en blå rektangel och sedan läggs animering till i form av skiftande genomskinlighet (W3S 2013b).

5.2.6. Web Graphics Library - WebGL

Med WebGL:s framfart har beräkningstunga grafiska operationer blivit möjliga att utföra i vanliga webbläsare med ett *Canvas*-objekt som inkörningsport. WebGL är baserat på OpenGL som är ett API för rendering av grafik i skrivbordsmiljöer. OpenGL 1.0 släpptes 1992 och på den tiden var det en öppen standard som utvecklades parallellt med dess kommersiella medtävlare. Idag har OpenGL nått version 4.4 men WebGL baseras på en tidigare version: OpenGL ES 2.0. Förkortingen står för *Embedded Systems*, är baserat på OpenGL 2.0 och är riktad till mobila enheter med sämre prestanda än vanliga datorer.

Följden är att språket är mer restriktivt och har ett mindre API än dess fullfjädrade föregångare. Trots detta kan man åstadkomma ungefär samma saker i de båda alternativen!

WebGL ställer höga krav på utvecklaren. Förutom goda matematik- och programmeringskunskaper behövs även förståelse om bl.a. *shader*-programmering. Användandet av WebGL är ännu i ett tidigt skede men kommer inom de närmaste åren troligen innebära framsteg inom spelutveckling m.m. på webben. Ett exempel på detta finns i utvecklarforumet *Chrome Experiments* (se figur 5.1). WebGL Globe heter experimentet och går ut på att man tillhandhåller ett fundament i form av en jordglob för användare att visualisera sin data på.



Figur 5.1: Experimentet *WebGL Globe* från *Chrome Experiments*. Stommen är en jordglob, kodad i WebGL, som man låter användare tillämpa med sin egen data (Chrome Experiments 2013).

5.2.7. Web Map Service - WMS

Fram till nu har de nämnda standarderna varit generella för nästan all form av webbutveckling. WMS är däremot en standard framtagen av OGC som används i stor utsträckning i samband med kartapplikationer på webben. Dess specifikation definierar ett HTTP-protokoll för hur klient och server skall kommunicera med varandra när kartbilder anropas. Returprodukten är en rasterbild som tillåter användaren panorera, zooma och visa viss attributinformation (*GetFeatureInfo*).

Ingående lager kan vara både raster- och vektordata, vad som finns att tillgå tas fram genom ett s.k. *GetCapabilities*-anrop. Klienten får då reda på all nödvändig information som krävs i ett anrop till just den servern. I exempel 5.6 nedan syns hur ett sådant anrop kan se ut (Michaels & Ames 2008).

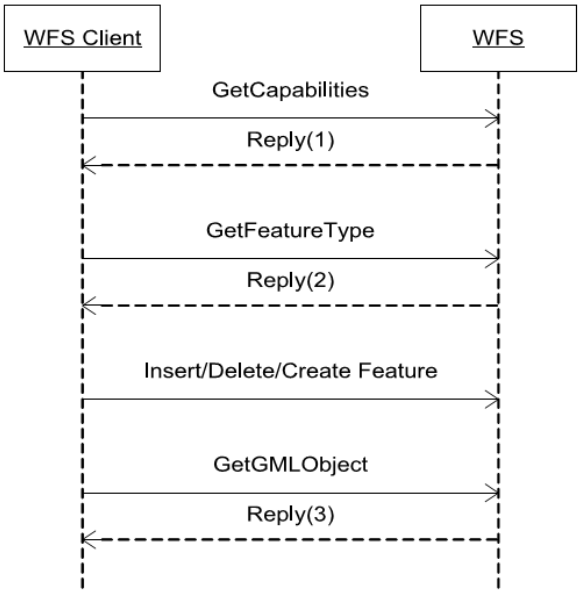
Exempel på ett *GetMap*-anrop i en WMS. Högerspalten utgör själva anropet och vänsterspalten är en kort beskrivning om de olika parametrarna. Nedan anges endast obligatoriska parametrar.

Adressen där kartan finns	http://localhost:8080/geoserver/wms?
Anropet är av *GetMap*-karaktär	request=GetMap
Protokollet som används	&service=WMS
Protokollets version	&version=1.1.1
De lager som begärs	&layers=topp:states
De stilar som begärs	&styles=
Referenssystem	&srs=EPSG:4326
Inhängning av begärt område	&bbox=-145.15104058007,21.731919794922, 57.154894212888,58.961058642578&
Kartans bredd	&width=780
Kartans höjd	&height=330
Vilket bildformat kartan presenteras i	&format=image/png

Exempel 5.6: Ett WMS-anrop där klienten begär en karta från servern, s.k. *GetMap*-anrop (GeoServer 2013).

5.2.8. Web Feature Service - WFS

Vad som tidigare nämnts om WMS gäller även i stora drag om WFS, men det finns vissa skillnader. WMS-protokollet är uteslutande konstruerat så att en klient kan titta på en, av servern, färdigproducerad kartbild. I en WFS är det däremot inte en bild som tas emot av klienten utan snarare geografiska vektordata (*features*), vilka senare används för att tillverka kartan på klientsidan. Vad innebär detta? Jo, snarare än att vara en titthålskarta kan man i en WFS även få tillgång till, ta bort samt uppdatera rådata. Det är således möjligt att göra fler anrop till en WFS än en WMS och kommunikationen mellan klient och server är därför även mer komplicerad. I figur 5.2 syns en överblick om hur händelseförloppet kan se ut. Slutprodukten i ett WFS-anrop kan t.ex. vara data i GML format, vilket då nödvändigtvis inte behöver komponeras till en karta hos klienten utan kan även sparas ned för senare användning.



Figur 5.2: Händelseförloppet i kommunikationen mellan klient och WFS-server (Sinha 2008 s.1258).

5.2.9. Web 3D Service – W3DS

W3DS är ett protokoll för avbildning av tredimensionell geodata så som stads- och landskapsmodeller. Det är ännu inte en OGC-standard utan finns idag som ett *OGC Discussion Paper*, vilket innebär att tekniken fortfarande granskas och är inofficiell.

Med tredimensionell avbildning menas att 3D-objekten i sig inte visas utan snarare visningselement som motsvarar geometrin för de geografiska objekten. Dessa ingår i s.k. scener, som i sig är optimerade för rendering i realtid (se figur 5.3). Det finns här likheter med WMS i att en avbildning representeras istället för ursprunglig vektordata samt att ytterligare attributinformation kan nås genom ett *GetFeatureInfo*-anrop. Som indataformat föreslås X3D (ISO-standard) och KML (OGC-standard) eftersom de är anpassade för realtidsvisualisering och kan hantera kartografiska egenskaper i 3D (W3DS 2010).

Figur 5.3: En W3DS över en stadsmodell (W3DS 2010).

5.3. Utbytesformat

För att två individer ska kunna kommunicera måste de först enas om ett format men även kunna prata samma språk, annars är det omöjligt att utbyta information med varandra. Detsamma gäller för datorer och datorprogram. Protokollet HTTP och märkningsspråket HTML utgör exempelvis tillsammans en sådan konstellation men problemet är att de är för oprecisa: de är inte lämpade för att beskriva olika enheters datasammansättning (Salminen & Tompa 2011). Vare sig det är data eller grafiska komponenter som utbyts finns det behov

av standardiserade format som på ett rimligt sätt kan paketera och distribuera informationen. Ett isolerat utbytesformat medför att applikationer som talar flera språk kan kommunicera med varandra och att kvalitén på datautbytet kan stärkas genom utveckling av utbytesformatet i sig.

5.3.1. eXtensible Markup Language - XML

XML är en W3C-rekomendation och är ett märkningsspråk som idag används i väldigt stor utsträckning. HTML, som ju också är ett märkningsspråk, har funnits långt innan XML men har alltså inte kunnat fylla de behov som uppstår i samband med filutbyten. Notera här att språken har olika syften: HTML för att visa data och XML för att distribuera och lagra data, men klart står att XML kompletterar HTML:s brister (W3S 2013c). En stor skillnad är att i XML tillåts användare definiera egna element. På så vis ges en större frihet att bygga upp en struktur precis på det sätt som funkar bäst för den aktuella tillämpningen (se exempel 5.7 nedan). Denna faktor är det som bidragit till att det idag finns hundratals olika dialekter av XML. (Salminen & Tompa 2011).

En XML-fil som beskriver en CD-katalog innehållandes två CD-skivor.

```
<CATALOG>
        <CD>
                <TITLE>Empire Burlesque</TITLE>
                <ARTIST>Bob Dylan</ARTIST>
                <COUNTRY>USA</COUNTRY>
                <COMPANY>Columbia</COMPANY>
                <PRICE>10.90</PRICE>
                <YEAR>1985</YEAR>
        </CD>
        <CD>
                <TITLE>Hide your heart</TITLE>
                <ARTIST>Bonnie Tyler</ARTIST>
                <COUNTRY>UK</COUNTRY>
                <COMPANY>CBS Records</COMPANY>
                <PRICE>9.90</PRICE>
                <YEAR>1988</YEAR>
        </CD>
</CATALOG>
```

Exempel 5.7: En enkel XML-fil. (modifierad från W3S 2013d).

5.3.2. Geography Markup Language - GML

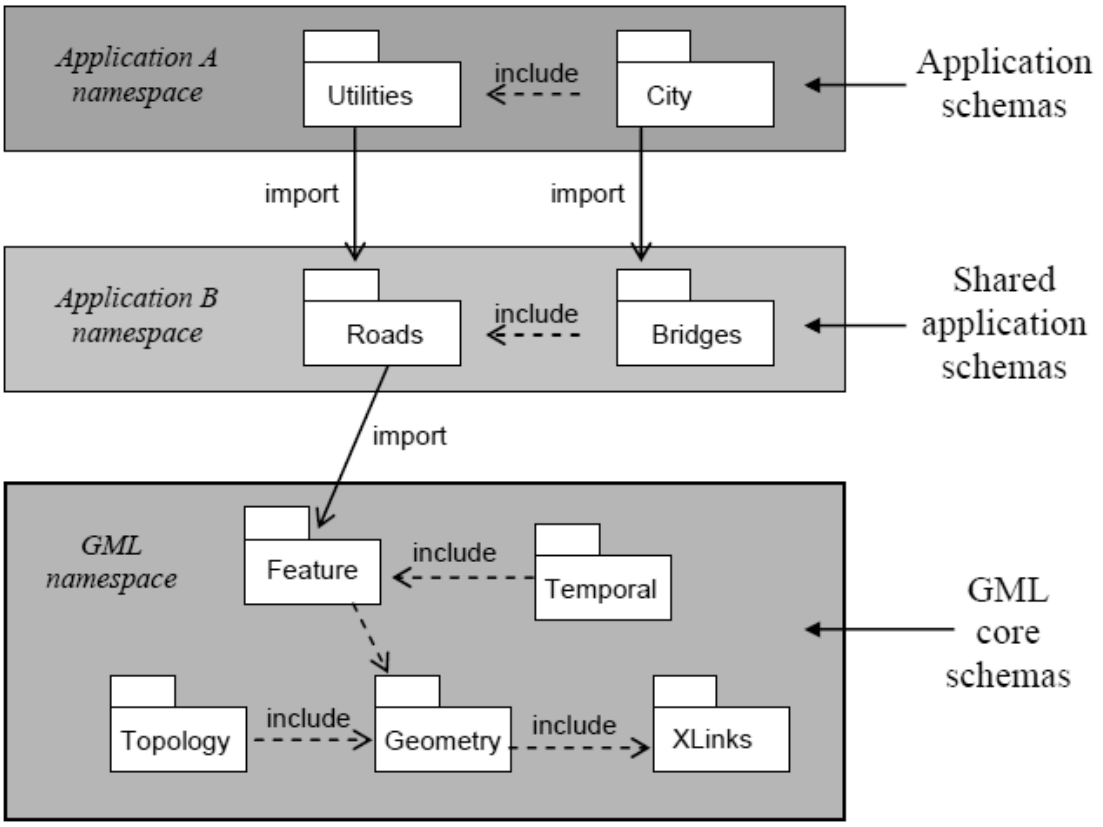
Inom Webb-GIS är GML ett av de mest centrala utbytesformaten och används exempelvis som svarsprotokoll av WFS-servrar. Märkningsspråket är en OGC-standard och en dialekt av XML för att standardisera geografisk data. Ett GML-objekt, en s.k. *Feature*, beskriver en instans i form av en konkret (t.ex. en väg eller bro, se exempel 5.8 nedan) eller en konceptuell händelse (t.ex. mobiltäckningskarta eller politiska gränser). Ett objekt beskrivs av sammansättningen dess egenskaper utgör, som i sin tur t.ex. kan vara av geometrisk-, topologisk- och temporal karaktär (Burggraf 2006).

En GML-fil som beskriver ett vägsegment.

```
<app:RoadSegment>
        <app:name>Fredsgatan</app:name>
        <gml:centerLine>
                <gml:LineString srsName="EPSG:4326">
                        <gml:coordinates>1,2 2,3 3,4 4,0</gml:coordinates>
                </gml:LineString>
        </gml:centerLine>
</app:RoadSegment>
```

Exempel 5.8: En enkel GML-fil. Taggarna som börjar på ”app:” kommer från ett tillämpat applikationsschema medan taggarna som börjar på ”gml:” är kärnobjekt i GML-schemat (modifierat från Burggraf 2006).

Uppbyggnaden av en GML-fil kan ske med applikationsspecifika objekt som i sin tur baseras på standardelement. Dessa särskiljs genom att taggens namn börjar på ”*app*” respektive ”*gml*”. Olika organisationer kan tillämpa GML på olika sätt, viktigt är då att det finns ett tillhörande schema där de applikationsspecifika elementen redogörs för. Exempel på sådana tillämpningar är CityGML för 3D-stadsmodeller samt GeoSciML för geologiska data. Dessa klassificeras dock som generella applikationsscheman, medan möjlighet finns att bygga på dessa med ännu mer applikationsspecifika scheman. I figur 5.4 nedan beskrivs de olika nivåer ett GML-schema kan bestå av.



Figur 5.4: Olika nivåer av GML-scheman. Kärnan är alltid schemat för GML (längst ner), därefter finns vedertagna påbyggnader så som CityGML och GeoSciML (mitten). Slutligen kan man även bygga vidare med egna specifika applikationsscheman (Burggraf 2006 s.179).

5.3.3. GeoScience Markup Language - GeoSciML

GeoSciML är ett GML-baserat utbytesformat för geologiska data. I dess grundutförande täcks vanliga företeelser på geologiska kartor men det är möjligt med applicering på provborrnings- och analysdata. Formatet började utvecklas 2003 och används idag bland annat av tidigare nämnda *OneGeology Portal* samt i *INSPIRE*-direktivet. En stark drivkraft bakom utvecklingen har varit att geologiska data de facto är oberoende av nation- och regiongränser (CGI 2013). I dessa yrkesutföranden är man således ofta tvungen att samla data från flera, av varandra, oberoende källor. Resultatet kan då bli att data har olika format och resurser behöver således läggas på dataförädling (en aspekt inom GIS-världen som är kostsam och ”onödig” om det går att undvika). Målet har därför hela tiden varit att standardisera hur geologiska data sprids. Ett exempel med borrhål syns nedan.

```
<gml:position>
    <gml:Point     gml:id="bh.121911.start"     srsName="epsg:7405">
                        <gml:pos>468410     0437120     10</gml:pos>
    </gml:Point>
</gml:position>
                        .
                        .
                        .
                <xmml:MultiInterval srsName="#bh.121911.shape">
                    <xmml:intervalMembers>
                        <xmml:Interval>
                            <gml:pos>0</gml:pos>
                            <gml:pos>4.57</gml:pos>
                        </xmml:Interval>
                        <xmml:Interval>
                            <gml:pos>4.57</gml:pos>
                            <gml:pos>6.09</gml:pos>
                        </xmml:Interval>

                    </xmml:intervalMembers>
                </xmml:MultiInterval>
```

Exempel 5.9: Delar från ett GeoSciML-exempel med en provborrning. Första stycket definierar position i horisontell led och andra stycket delar upp borrhålet i vertikal led (modifierad från Sen & Duffy 2004 s.1099).

5.3.4. JavaScript Object Notation och GeoJSON

JSON är ett textbaserat utbytesformat för representation av enkla objekt. Från början var formatet en delmängd av JavaScript men har genom tiden fått spridning till många andra språk. Dess nytta och användning är ganska snarligt den för XML, men det finns några små skillnader: JSON är enklare för människor att läsa och förstå, dess filer tar ofta mindre utrymme samt att XML-värden inte har specifika datatyper (IETF 2006).

GeoJSON är en påbyggnad för att representera enkla geografiska objekt och dess icke-geografiska attribut. Med enkla geografiska objekt menas här tvådimensionell data som är definierat i standarder hos både OGC och ISO (punkter, linjer, polygoner etc.). Notera här att tvådimensionell syftar till objektens utbredning i rummet, en punkt kan således ändå ha tredimensionell positionsangivelse (ISO 2004 & GeoJSON 2008).

JSON-formatet drivs inte av OGC men har ändå stöd av många välkända GIS-produkter, bland annat OpenLayers, Leaflet, GeoServer och PostGIS. Dess enkelhet och spridning är ett argument för vidare användning som utbytesformat för de geotekniska provborrningarna i detta projekt. Nedan syns ett exempel på hur en sådan punkt skulle kunna se ut uttryckt i GeoJSON.

```
{ type: 'FeatureCollection',
        features: [{
                type: 'Feature',
                geometry: {
                        type: 'Point',
                        coordinates: [-77, 37.9]},
                properties: {
                        title: 'Punkt 1123',
                        description: 'Punktbeskrivning, objektlänkar
                        etc'
                        'marker-color': '#CC0033'}
        },{
                type: 'Feature',
                geometry: {
                        type: 'Point',
                        coordinates: [-78, 36.5]},
                properties: {
                title: 'Punkt 1211',
                        description: 'Punktbeskrivning, objektlänkar
                        etc',
                        'marker-color': '#CC0033'}
        }]
}
```

Exempel 5.10: Två punkter uttryckta i GeoJSON-format. Observera att om inget annat anges är standardkoordinatsystemet uttryckt i decimala longitud och latitud baserat i referenssystem WGS 84 datum. (GeoJSON 2008).

5.3.5. Extensible 3D – X3D

X3D är en XML-dialekt till för att uttrycka geometrier tre dimensioner. Det är en ISO-standard och baseras på den tidigare tekniken VRML (*Virtual Reality Modeling Language*).

En X3D-fil byggs upp av komponenter i en trädstruktur där varje objekt utgör en nod som har en förälder (förutom roten). Hierarkin ligger sedan till grund för hur saker visas och animeras (Brutzman & Daly 2007).

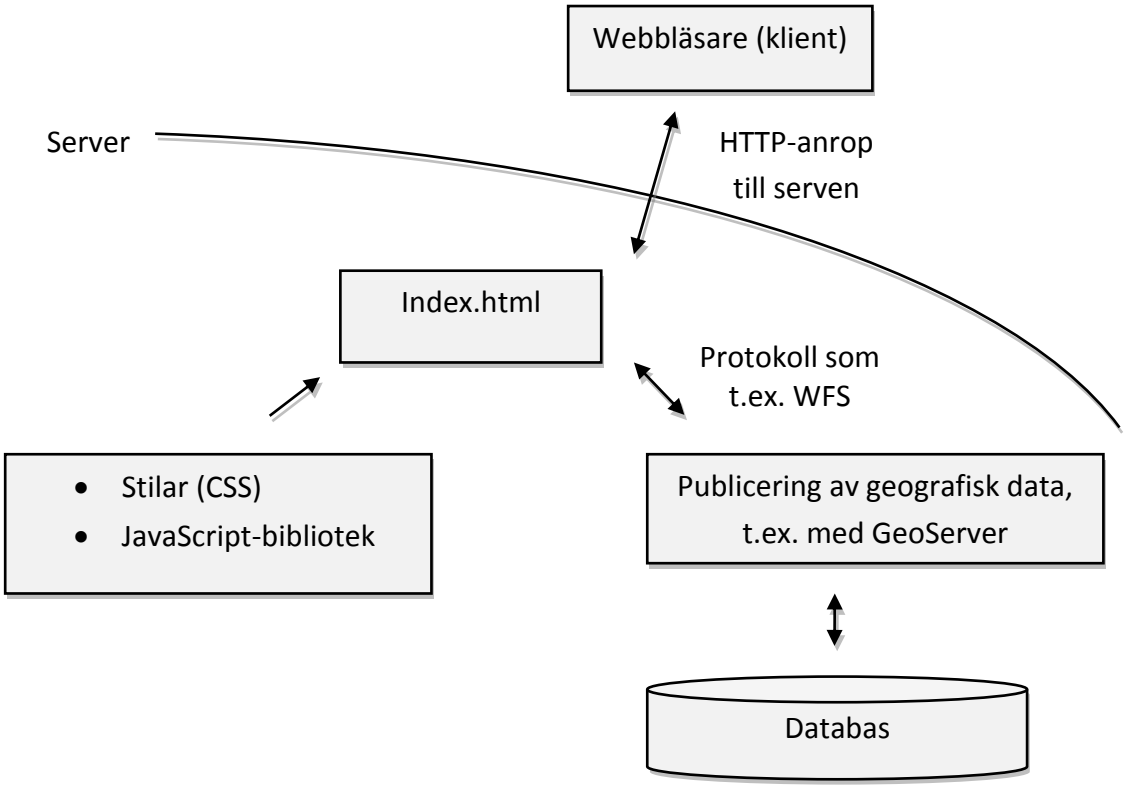
X3D-formatet behöver i dagsläget insticksmoduler för att köras i webbläsaren. Problemet löses med införandet av X3DOM: en modell som möjliggör direkt hantering av X3D som HTML5/DOM-element (X3DOM 2013).

5.4. Systemarkitektur Webbapplikation

En webbapplikations arkitektur måste anpassas så att kravspecifikationen kan stödjas i största möjliga grad. För att lösa detta finns det, grovt uppdelat, två olika strukturalternativ: en applikationsserver med webbprotokoll som t.ex. WFS eller lagring i filstrukturen på en server där t.ex. en JSON-fil finns på samma plats som hemsidan. Nedan redogörs för de två olika alternativen med för- och nackdelar.

5.4.1. Applikationsserver och webbprotokoll

Med webbprotokoll menas här de OGC-standarder som tas upp i avsnitt 5.2.7.– 5.2.9., nämligen WMS, WFS och W3DS. Internt skiljer sig dessa alternativ något men applikationens arkitektur är snarlik för samtliga alternativ. I grunden behövs någon form av databas (se avsnitt 6.1.) följt av en kartpubliceringstjänst som t.ex. GeoServer (se avsnitt 6.2.). En klient, skriven i HTML/JavaScript, framställer sedan kartan genom anrop till servern. Beroende på valt protokoll kan man sedan se, redigera eller lägga till företeelser och attribut i databasen. Arkitekturen kan se ut enligt figur 5.5 nedan.



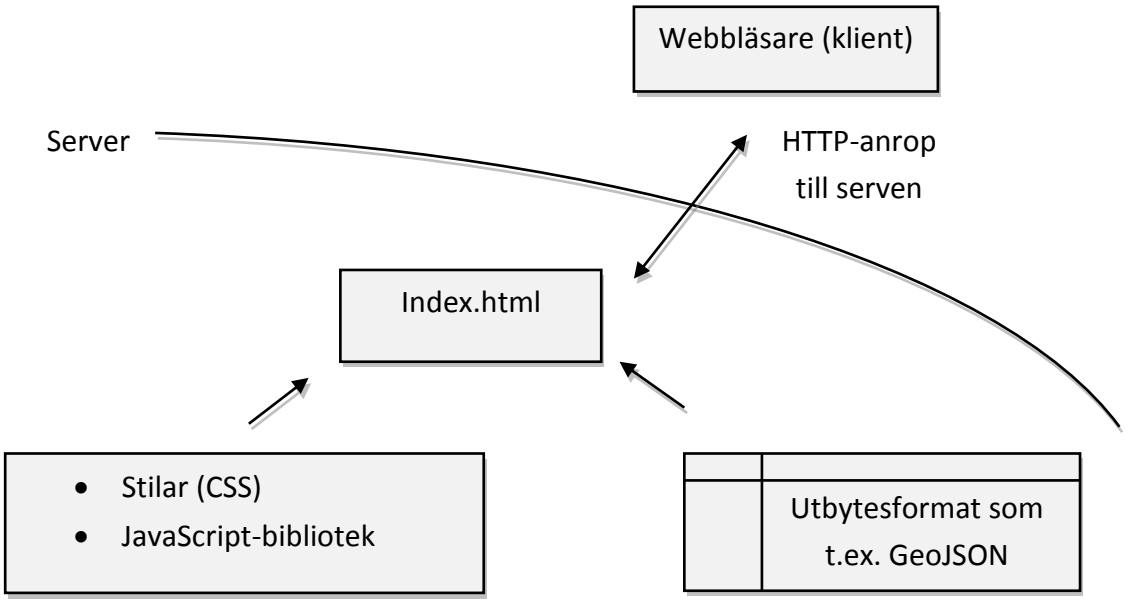
Figur 5.5: Arkitektur om en tjänst byggs med OGC-protokoll.

En fördel med detta förfarande är att serververktyg, så som GeoServer, har stöd för bricklager (engelska: tiled layer) från start. Stor mängd företeelser kan då laddas in i kartan utan någon större tidsåtgång. En annan fördel (åtminstone i fallet med WFS) är att datasetet

kan modifieras. Företeelser kan läggas till, redigeras och tas bort vilket i vissa fall är en viktig funktionalitet.

5.4.2. Filstruktur i en webbserver

En lokal lagringsstruktur är en något enklare metod att implementera. Kärnfilen i HTML ingår då i ett paket som innehåller all JavaScript-kod tillsammans med filer som innehåller företeelser. Dessa lagras sedan i utbytesformat som GeoJSON eller XML. Datasetet kan inte uppdateras av en användare utan inslag av programmering på serversidan (t.ex. PHP), vilket är en nackdel i vissa fall. Däremot finns möjligheter i form av tidsinställda script-körningar som hämtar data från en databas och uppdaterar utbytesfilerna (förfarandet beskrivs i avsnitt 8.3.5.). Denna metod är inte tillämpbar i alla fall utan fungerar endast i exempel där data som mest kräver daglig ajourhållning.



Figur 5.6: Arkitektur om en tjänst byggs med en lokal GeoJSON-fil.

6. Produkter

6.1. Databaser

6.1.1. SQLite och SpatiaLite

SQLite är en enkel databashanterare som bygger på en relationsdatabas. Med enkel menas här flera olika aspekter, till exempel att hela systemet ryms i cirka 500 kb C-kod. Enkelheten har alltid varit ett mål under dess utveckling och en följd av detta är att SQLite är serverlös (d.v.s. måste köras lokalt) och behöver därför ingen installation eller nätverkskonfiguration. Snarare än en egen process integreras databashanteraren istället i det program som

använder den. Alla program som kan nå en hårddisk har dessutom möjlighet att använda SQLite (SQLite 2013a).

Utöver dess enkelhet är SQLite även pålitligt. Källkoden består av ca 30 000 rader och är byggd som ett API så att anpassning till egna behov är möjlig utan att äventyra säkerhetsaspekter. SQLite är gratis att använda och är helt fri från kopieringsskydd, vilket innebär ungefär att dess grundare (och framtida användare och utvecklare) avsagt det kommersiella intresset med projektet. Allmänheten har gjort stora bidrag till projektet när det gäller dess utveckling. Ett konkret exempel är att det finns fler rader testkod än det finns programkod, vilket kanske kan ge en indikation på dess pålitlighet (Owens 2006).

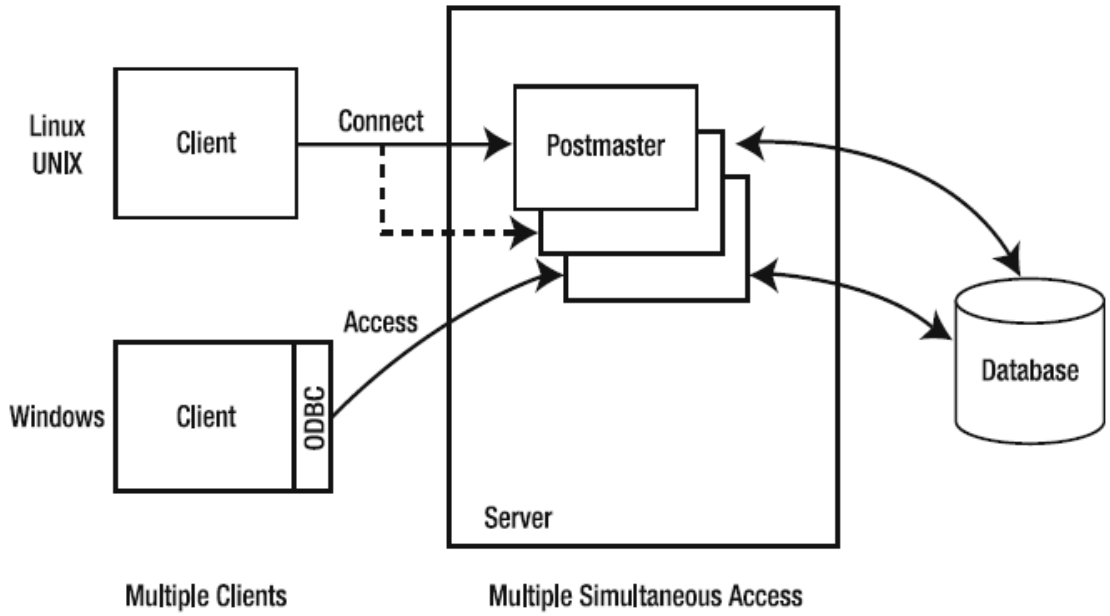
SQLite har sedan version 3.6.2 R-träd implementering, vilket är en vanlig indexeringsmetod i spatiala databaser. Dokumentationen är i skrivande stund knapphändig men det finns exempel där man använt SQLite med punktdata (SQLite 2013b).

Med tilläggsmodulen SpatiaLite möjliggörs hantering av geografiska data i SQLite. Målsättningen är, likt sin förgrund, att tillhandhålla ett enkelt och lagom avancerat databashanteringssystem. Mer än så är det inte: om det tänkta användningsområdet ska ha stöd för alla sorters geografiska funktioner och flera användare samtidigt bör andra alternativ användas (SpatiaLite 2013).

Funktionerna i SpatiaLite följer OGC och ISO:s *Simple Feature Access*-standarder. Dessa definierar hur relationsdatabaser i SQL ska hantera geografisk data, vilket möjliggör användning tillsammans med andra vedertagna GIS-produkter (ISO 2004).

6.1.2. PostgreSQL och PostGIS

PostgreSQL är den vanligaste och mest avancerade databashanteraren av öppen källkod-typ. Projektet har mer än 15 års aktiv utveckling som bakgrund och är kompatibelt med alla stora operativsystem (PostgreSQL 2013). Precis som många andra kommersiella alternativ kan man använda PostgreSQL i en klient/server-miljö. Hjärtat är en serverprocess och all åtkomst till databasen går via den, även vid begäran från lokala klienter. Uppdelningen på detta vis tillåter databasen att t.ex. ligga på en Linux-server medan klientprogrammen körs på Windows (Matthew & Stones 2005). En illustration över hur arkitekturen ser ut syns i figur 6.1 nedan.



Figur 6.1: Arkitekturen i PostgreSQL (Matthew & Stones 2005 s.14).

Källkoden är under s.k. PostgreSQL-licens vilket tillåter fri användning, spridning och modifiering. Olikt vissa öppen källkod-licenser, som t.ex. *General Public License*, är det här däremot tillåtet att ha stängd licens på en egen modifiering. Detta medför att PostgreSQL både används som färdig produkt men även som en utvecklingsplattform för vidare nischer.

Idag finns många avancerade funktioner inbäddade, en del av dem helt unika för PostgreSQL. Ett sådant exempel är GiST (*Generalized Search Tree*) som är ett gränssnitt där man kan definiera egna datatyper och bestämma vilka lagringsstrukturer de ska anta. GiST har sedan kommit att ligga till grund för en rad projekt, däribland även PostGIS (PostgreSQL 2013).

Databasmodellen i PostgreSQL är objekt-relationell och för att hantera geografisk data krävs införande av nya datatyper. Detta är något som tillägget PostGIS gör, inte helt olikt fallet ovan med SQLite och tillägget SpatiaLite. Utöver definiering av nya datatyper tillhandhåller PostGIS även spatiala funktioner och indexeringsförbättringar. Även PostGIS har öppen källkod och är en av de mest använda spatiala databashanterarna. Generellt har det fler utmatningsformat än konkurrerande kommersiella alternativ och är i många fall även snabbare (Obe & Hsu 2011).

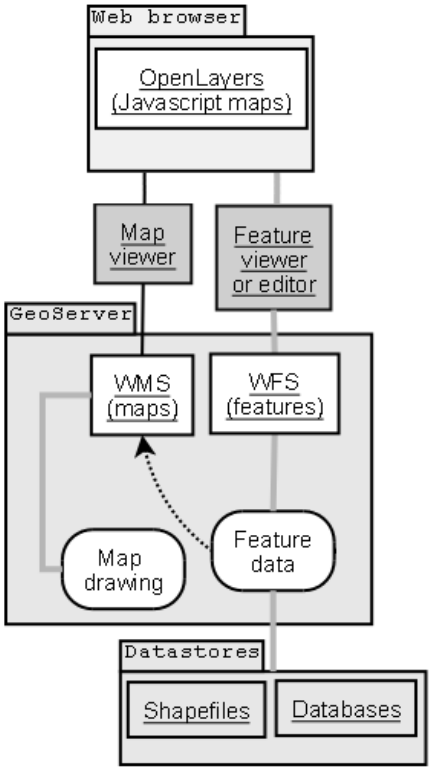
PostGIS är väl lämpat för både små och stora projekt. Dess vida kompabilitet gör det enkelt att applicera databashanteraren i alla projekt. Ett storskaligt exempel är lantmäteriet i Frankrike, *Institut Géographique National*, där man använt PostGIS för att lagra en grundkarta över landet (PostGIS 2012). Systemet, som heter BDUni, har över 100 miljoner företeelser och underhålls av ca 100 fältarbetare som dagligen tillför nya objekt.

6.2. Kartpubliceringsverktyg

6.2.1. GeoServer

GeoServer är ett verktyg för att distribuera geografisk information på webben. Det är gratis, innefattas av öppen källkod-licens och använder sig av öppna standarder från OGC (så som WMS och WFS). Ett användarvänligt webbgränssnitt finns som tillåter hantering av kartprodukter i en interaktiv miljö. Där finns även stöd för enkel förhandsvisning av data med hjälp av OpenLayers (GeoServer 2009).

Av sina medtävlare är GeoServer det mest använda alternativet och kan beskrivas som ett nav för GIS-produkter i öppen källkod (se figur 6.2).



Figur 6.2: En karta över olika OSGeo-projekt där GeoServer ofta används som ett nav (modifierad från Wikipedia 2010).

6.2.2. MapServer

Ett annat OSGeo-projekt är MapServer vars syfte är densamma som för GeoServer. Vissa skillnader föreligger dock, MapServer har exempelvis inte ett grafiskt gränssnitt utan ingående data anges istället i textfiler, s.k. map-filer. Programspråket är C och applikationen är generellt mer avskalad än GeoServer, i MapServer finns exempelvis ingen förhandsvisning av den slutgiltiga kartprodukten (MapServer 2013).

6.3. Bakgrundskartor

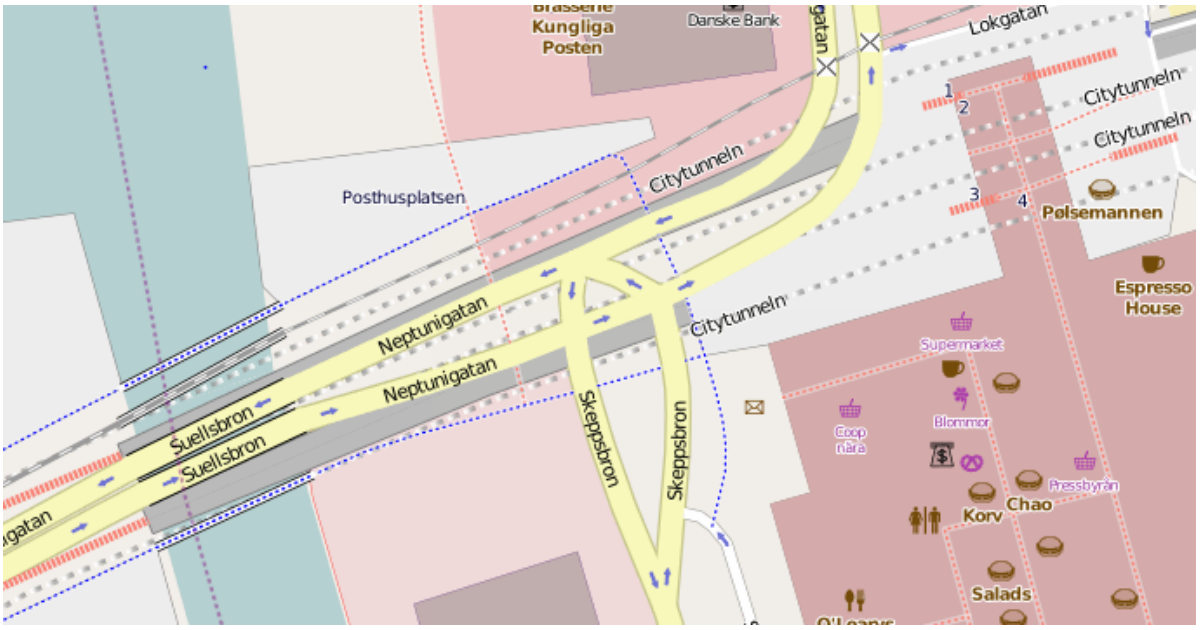
Det finns flera olika bakgrundskartor att välja på med varierade utseenden, informationsmängd och kostnad. Valet styrs av faktorer som t.ex. visuell hierarki som syftar till att styra användarens uppmärksamhet mot den informationen som är viktigast, vilket exempelvis uppnås med linjestorlekar och utmärkande färger. En karta kan ha som syfte att framhäva specifik information, som t.ex. en geologisk karta, och faller då under kategorin tematiska kartor. Nedan redogörs för några kartalternativ som skulle kunna vara aktuella (OSM 2013a).

6.3.1. Open Street Map

Open Street Map (OSM) är en karta som är helt gratis att använda. Innehållet är insamlat av egna användare (engelska*: crowd sourcing*) och idén växte ur en frustration över det dyra priset på statliga organisationers geografiska data. OSM lanserades först i Storbritannien

men fick snabbt spridning, idag har man över 1 miljon användare som hjälper till att förbättra kartan.

Informationsmängden i kartan är relativ hög. Utseendemässigt är kartan från början ljus med lila färger för administrativa gränser och namn. När man rör sig mot lägre skalnivåer ändras färgschemat till en mer varierad sådan. På små skalnivåer syns även cykelstråk såväl som butiker och restauranger (se figur 6.3).



Figur 6.3: Sista skalnivån i *Open Street Map* innehåller mycket information. Kartan visar delar av Malmö Centralstation med omkringliggande restauranger m.m. (OSM 2013b).

6.3.2. Google Maps

Google tillhandhåller en bakgrundskarta som inte är helt olik OSM. Dess kartografiska egenskaper är av hög kvalité och informationsmängden är lagom i alla skalnivåer. Googles kartor används idag av många för privat bruk. Tidigare var kartorna helt gratis men sedan 2011 tar man betalt för användning över en viss gräns från kommersiella sidor. Gratisversionen får dessutom endast användas om värdsidan är publik och öppen för allmänheten.

Figur 6.4: Google Maps bakgrundskarta (Google Maps 2013).

6.3.3. Stamen Maps

Stamen Maps Toner är, ur kartografiskt avseende, en något udda bakgrundskarta. Den är avskalad och färgkonventionen är anpassad till ett minimalistiskt och stilrent utseende. Kartan är gratis att använda och har tydliga tillämpningar där bakgrundskartan ska ha så lite fokus som möjligt.

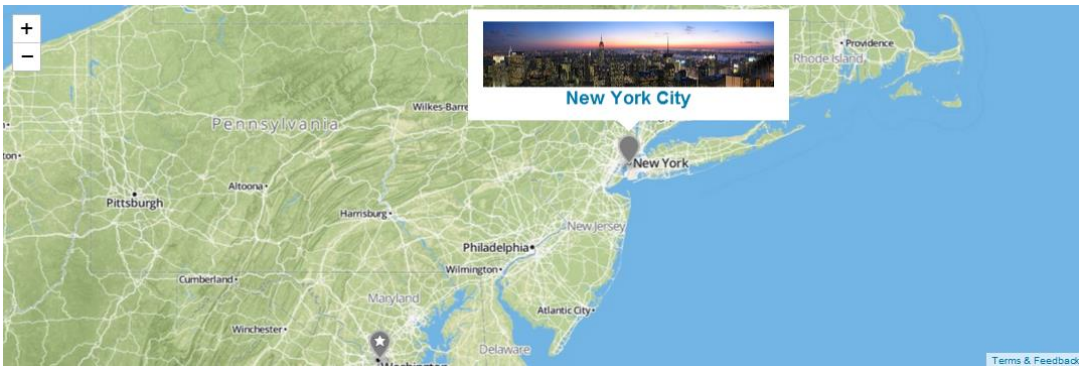
Figur 6.5: *Stamen Maps Toner*-bakgrundskarta (Stamen 2013).

6.4. Exempel från applikationsgränssnitt och färdiga tjänster

Detta projekt berör kartapplikationer med inslag av 3D och punktlager i form av geotekniska provborrningar. Visualiseringsverktyg för detta har studerats och det finns flera bra alternativ, både öppen källkod och kommersiella. Generellt har HTML5:s intåg förbättrat webbaserade kartors estetik och användarvänlighet. Det finns numera flera bibliotek där utvecklaren får en verktygslåda och kan med enkla medel komponera ihop sin personliga applikation. Nedan beskrivs de bitar från olika produkter som bedöms vara mest intressanta för denna studie.

6.4.1. MapBox

MapBox är en betaltjänst där användare på ett enkelt sätt kan skapa snygga och anpassade kartor. Dess JavaScript API innehåller ett 60-tal tilläggsmoduler, i figur 6.6 demonstreras ett av dessa. Den skapar ett poppuppfönster som innehåller text och bild, vilket i ett tillämpat fall skulle kunna motsvara rapporter och länkar till referensdata.



Figur 6.6: Demonstration av insticksmodulen *Costum Marker Tooltip* i kartbiblioteket MapBox (MapBox 2013a).

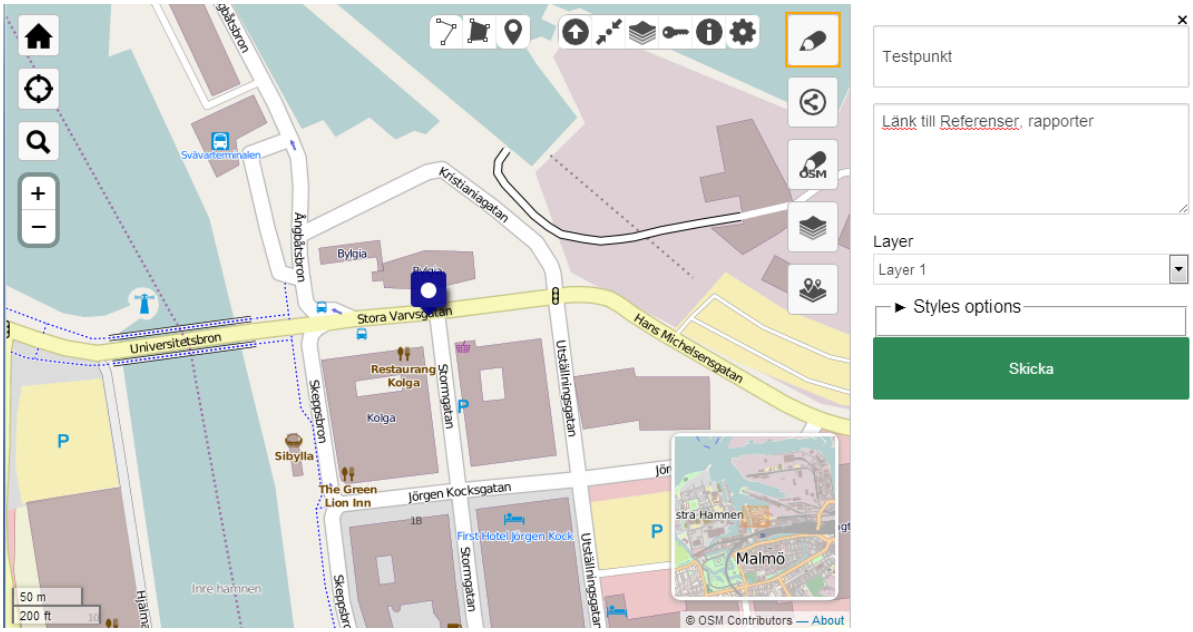
Ett annat potentiellt användbart tillägg är *Linking to External* Data. I detta exempel ändras flaggan uppe till höger när muspekaren svävar över olika länder (se figur 6.7). Det intressanta här är länkningen till extern data och i ett anpassat fall skulle flaggan kunna motsvara bilder på stratigrafiska lagerföljder när användaren klickar på ett punktobjekt.

Figur 6.7: Demonstration av insticksmodulen *Linking to External Data* i kartbiblioteket MapBox (MapBox 2013b).

6.4.2. uMap

uMap är en färdig tjänst som går att börja använda utan någon inlärningströskel. Medan föregående exempel var ett API så är uMap mer som en webbtjänst. Ett användarkonto skapas och sedan finns möjlighet för egen anpassning utifrån de funktioner som finns implementerade. En (vanlig) användare kan tända och släcka viss funktionalitet men inget utrymme finns för vidare anpassning. Som utvecklare finns större möjligheter. Eftersom uMap är helt i öppen källkod är det fritt att driftsätta en egen, anpassad, version av tjänsten.

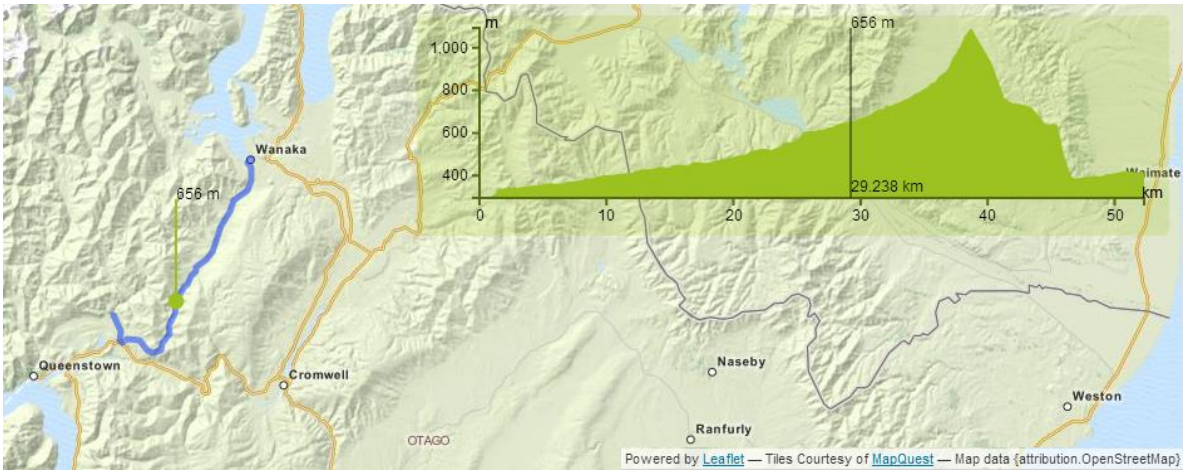
En fördel med uMap är att det finns färdiga verktyg för dataimport dels i form av frihands-redigering (se figur 6.8) men även uppladdningsmöjligheter för kommaseparerade textfiler (CSV), KML (Märkningsspråk för Google Earth) och GeoJSON. Med ganska enkla medel skulle man här kunna importera borrhål med tillhörande attribut och länkar till externa rapporter.



Figur 6.8: Addering av ett punktobjekt i öppen källkod-verktyget uMap (uMap 2013).

6.4.3. Leaflet

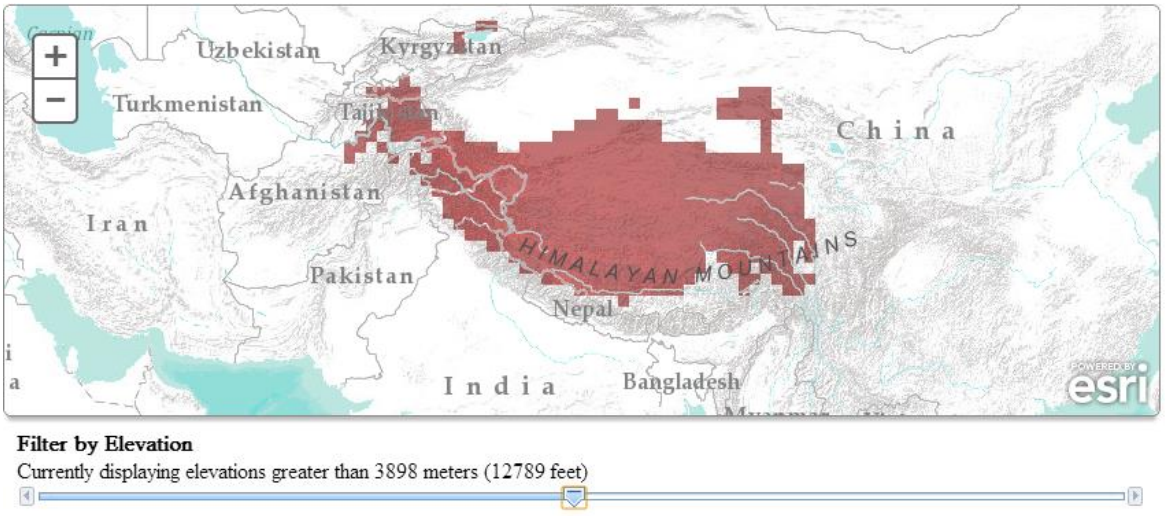
Leaflet är en annan öppen källkod-verktygslåda baserat på HTML5 och CSS3. Om MapBox i någon grad är riktad till användare så är Leaflet snarare helt riktad mot utvecklare. Dess API innehåller i stort sett liknande moduler men ett intressant tillägg är här modulen *Elevation* (se figur 6.9) som skulle kunna vara användbar för att visualisera jordlager. En höjdmodell skapas utefter en definierad sträcka i kartan. Vid panorering i höjdmodell-fönstret (grönt diagram) visas även vilken position angiven höjd har i kartan (grön spalt vid den blåa linjedragningen). Vid tillämpning skulle flera sådana diagram kunna läggas på varandra och skapa en lämplig tvärsnittsvy över hur jordlagren ligger i förhållande till varandra.



Figur 6.9: Insticksmodulen *Elevation* till Leaflet, ett öppen källkod-API (LeafLet 2013a).

6.4.4. ArcGIS

Även ArcGIS har ett rikt utvecklar-API. Innehållet är mer uttömmande än exemplen ovan och intrycket är att flexibiliteten är större. Det finns bra verktyg för visning och redigering av data men inget unikt i 3D-väg. Det enda nämnvärda är ett höjdmodellfilter som använder sig av *Canvas*. Verktyget tillåter användaren att se vilka områden som har större höjdvärde än det som anges i reglaget (se figur 6.10). Detta skulle eventuellt kunna appliceras på jordlager om man har en liknande höjdmodell över t.ex. sandnivåer och urberg. En annan tillämpning vore om man kunde använda två reglage, ett i horisontell led och ett i vertikal led för att kunna röra sig in och ut ur en 3D-modell. Vägen till att göra det ändamålsenligt användbart är nog dessvärre långt.



Figur 6.10: Höjdnivåfilter i ArcGIS API (ArcGIS 2013). Utifrån reglagets höjdvärde läggs ett rött rasterlager ovanpå kartan. De röda färgerna motsvarar alltså höjdvärden över 3898 meter.

6.4.5. OpenLayers

Det kanske mest vedertagna applikationsbiblioteket för geografisk information är OpenLayers. Projektet startade i mitten på 2000-talet och målet var att lösa ett, för tiden, stort problem. De stora karttjänsterna (Google Maps, Virtual Earth, Yahoo etc.) hade då egna proprietära API:n för att utnyttja deras data. Detta innebar problem för utvecklare, exempelvis i form av att man inte kunde växla mellan olika tjänster utan att skriva om all kod. OpenLayers ändrade på detta och möjliggjorde användning av olika karttjänster i ett enda API och dessutom även i en och samma karta. Utöver detta stödjer OpenLayers även OGC-standarder så som WFS och WMS som publiceras med verktyg så som MapServer och GeoServer (Obe & Hsu 2011).

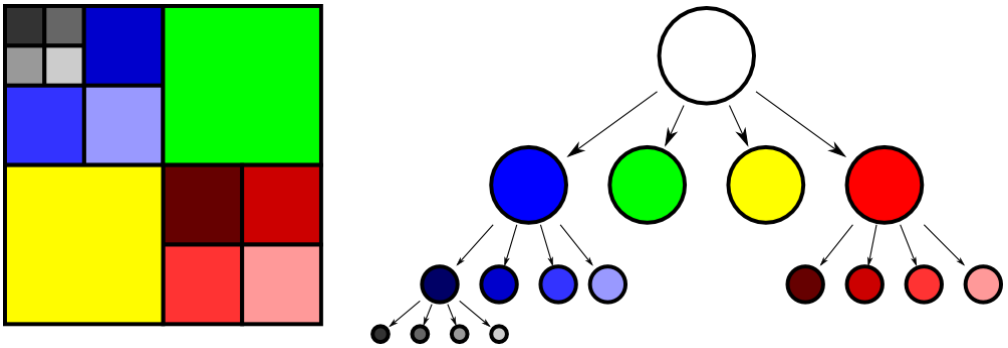
I skrivande stund finns ett begränsat antal exempel med OpenLayers och HTML5, Canvas och WebGL. Kommande version, OpenLayers 3, ska ha fullt stöd för dessa verktyg men ligger i dagsläget (2013) i ett alfa-stadie.

7. Visualisering av kvantitativa punktdata

Visualisering av kvantitativ data är en svår uppgift som i olika fall lösts med varierad kvalité. För att få inspiration om de problem som kan uppstå är det viktigt att titta på befintliga exempel och studera för- och nackdelar med dem. En generell uppfattning i dagsläget är att tekniskt avancemang inom detta område krävs. Problematiken ligger i att utföra för kostsamma beräkningar på klientsidan. Vid stora datamängder och ineffektiva generaliseringsalgoritmer finns det risk att stänga ute användargrupper som t.ex. mobiltelefonanvändare från applikationen. Nedan redogörs för ett par sätt att lösa detta problem på följt av exempel med kvantitativa punktdata från befintliga applikationer.

7.1. Quadtree

Några olika realtidsalgoritmer presenteras i Bereuter och Weibel (2013) där man använt indexeringsmetoden *quadtree* (ungefär: fyrling-träd) som utgångspunkt. Principen är att bygga upp ett nodträd där varje nod har fyra barn som representerar en spatial uppdelning i kvadranter (se figur 7.1). Tekniken har styrkor i bl.a. snabba rumsliga sökningar samt approximation av punktdensitet inom givna områden.



Figur 7.1: Principen för indexeringsmetoden *quadtree* (Wikipedia 2008).

Quadtree kombineras sedan med generaliseringsmetoder som urval, utjämning, aggregering eller förskjutning av punkter. Förfarandet och tillämpningarna skiljer sig något för de olika alternativen. De har alla däremot ungefär samma relation till trädstrukturen och idén är att generaliseringar ska ske beroende på vilken detaljnivå användaren befinner sig i. Med figur 7.1 som exempel innebär detta att när låg detaljrikedom krävs kan en generaliseringsmetod appliceras på de blåa, gröna, gula och röda kvadraternas punktmoln var för sig (Bereuter och Weibel 2013). När användaren rör sig mot högre detaljrikedom bildas nya uppdelningar exempelvis i form av kvadranter inom det röda området. Med t.ex. generaliseringsmetoden urval skulle då punktmolnet kunna representeras av den punkt vars placering är närmast punktmolnets mitt. Således förenklar man visningen från en ohanterlig mängd punkter till några få representativa punkter.

7.2. Multiplicatively Weighted Voronoi Diagram (MWVD)

Som tidigare nämnt är realtidsalgoritmer för punktgeneraliseringar ett område som fortfarande är i behov av utveckling. Ovan exempel är ett av få tillgängliga. En annan algoritm, som dessvärre inte går under kategorin realtidsalgoritmer, presenteras i Yan & Li (2013) och kalls för *Multiplicatively Weighted Voronoi Diagram* (MWVD). Generaliseringsmetoden som används är urval, d.v.s. man väljer att dölja en delmängd av ursprungspunkterna. Algoritmen använder sig av Voronoi-diagram och Delaunay-triangulering för att sätta upp ett topologiskt granneförhållande mellan punkter. Följande förutsättningar råder för algoritmen:

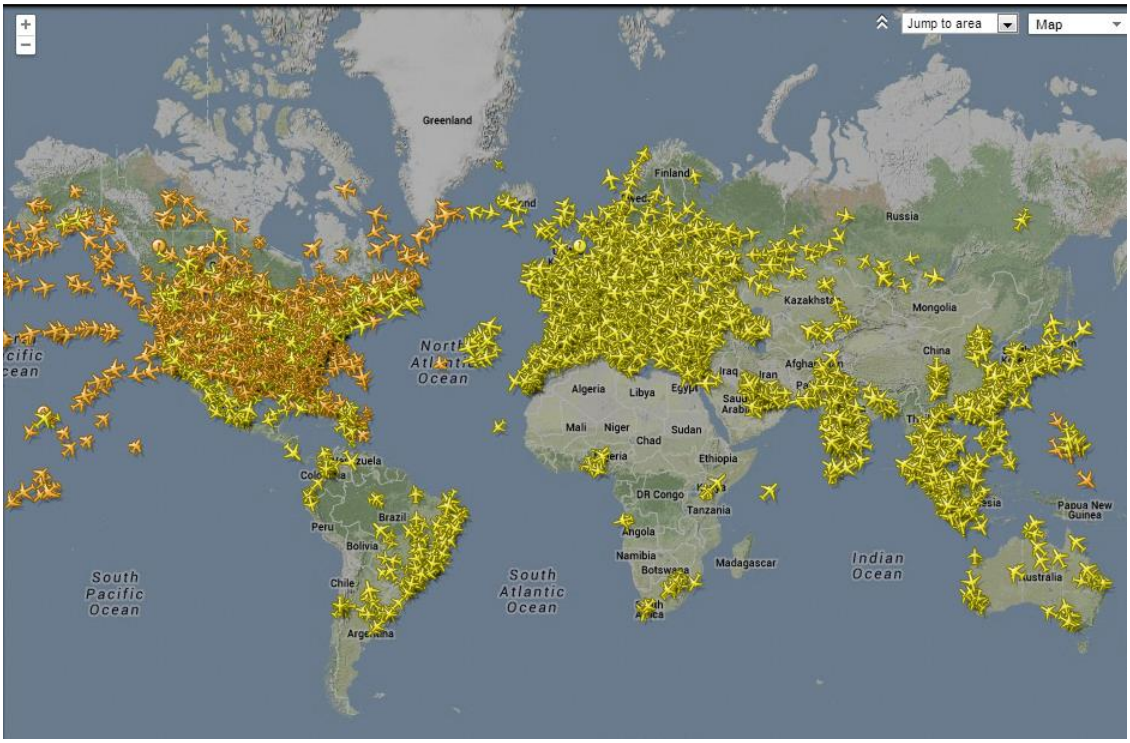
- Varje punkt kan ha tre olika tillstånd: borttagen, ledig eller fixerad.
- En punkt som har ett fixerat tillstånd kan ej byta tillstånd innan algoritmen är färdig.
- En punkt som är borttagen kan ej byta tillstånd, inte ens i ett senare skede.
- När algoritmen påbörjas är alla punkter i ett ledigt tillstånd.

Algoritmen för MWVD är enligt följande och återupptas för olika skalnivåer, d.v.s. olika generaliseringsgrader av punktmolnet:

1. Samtliga kvarvarande punkters sannolikhet att tas bort rankas och lagras i en vektor (sannolikheten är linjär mot dess polygonarea).
2. Börja med den punkt som har störst sannolikhet att tas bort, d.v.s. den med minst polygonarea, och ta bort den om följande villkor uppfylls:
    a. Den har ett ledigt tillstånd.
    b. Den har störst sannolikhet att tas bort av de punkter som är lediga.
    c. Ingen av dess Voronoi-grannar är borttaggna.
3. Om en punkt har tagits bort: Ge dess Voronoi-grannar ett fixerat tillstånd.
4. Gå till steg 2 och fortsätt tills ingen mer punkt kan tas bort.

7.3. Flightradar24

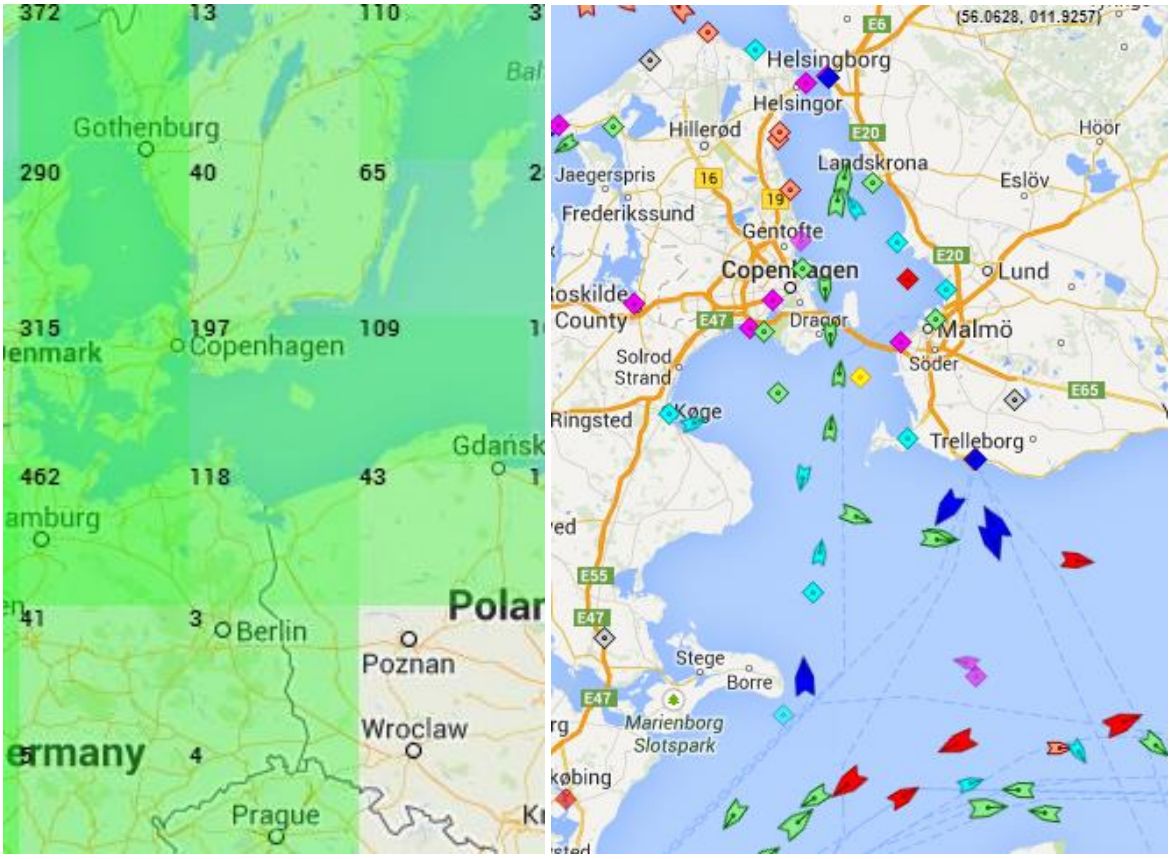
Flightradar24 är en hemsida som visar ett punktmoln i form av alla flygplan som för närvarande befinner sig i luften. Den kartografiska kvalitén är acceptabel på stora skalnivåer men vid en mer överblickande skalnivå blir företeelserna snabbt röriga och sidan blir framförallt långsam (se figur 7.2). Kartan innehåller ”endast” cirka 7500 företeelser som dessutom är utspridda över hela världen. Trots detta är kartan ohanterlig på mer än hälften av skalnivåerna. En sak som bör noteras är att kartan uppdateras i realtid vilket möjligen kan vara en förklaring till den enkla presentationen. Som bakgrundskarta används Google Maps som är något nedtonad vilket fungerar bra i och med att företeelserna då får mer uppmärksamhet.



Figur 7.2: Flightradar24 är en karta som uppdateras i realtid och visar samtliga flygplan som befinner sig i luften (Flightradar24 2013).

7.4. Marine Traffic

Nästa exempel är en karta som håller reda på fartyg i realtid. Den här kartan är dock mer intressant ur ett kartografiskt avseende jämfört med föregående exempel. På mindre skalnivåer skapas klusterområden i ett rutnät (se figur 7.3) som beskriver hur många fartyg som finns i området. Olikt förra exemplet upplevs detta inte alls som en rörig framställning och lösningen medför även att kartan flyter på bra. I mindre skalnivåer upphör klusterbildningen och symboler för fartygen visas istället. Symbolerna är olika för olika typer av fartyg och i allmänhet är mängden företeelser på en bra nivå. Som bakgrundskarta används, likt föregående exempel, Google Maps. Denna gång är den dock inte nedtonad vilket ändå hade varit en fördel även i detta fall.

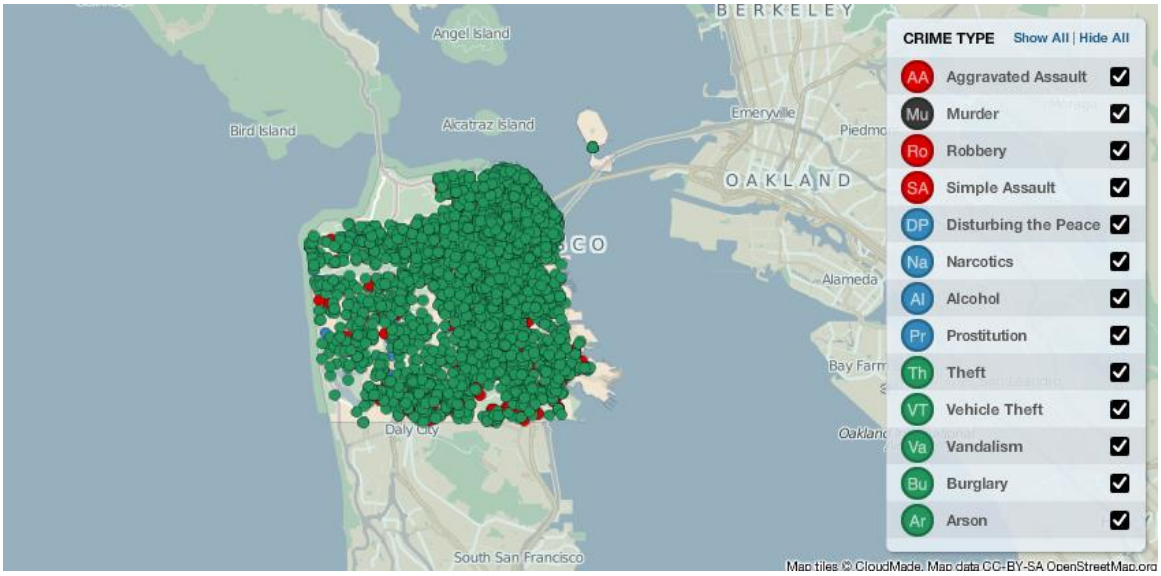


Figur 7.3: MarineTraffic är en karta som uppdateras i realtid och visar uppkopplade fartyg. Till vänster syns klusterbildningar som uppstår i mindre skala. Till höger syns en större skala där symboler för fartygen istället visas (MarineTraffic 2013).

7.5. San Francisco Crimespotting

San Francisco Crimespotting (på svenska ungefär: bevittnande av brott) är en interaktiv karta där olika brott placeras ut på en karta. Brotten är indelade i fyra kategoriska färger: svart för mord, rött för misshandel och rån, blått för narkotika m.m. och grönt för materiella skador. Kartan har olika reglage för vilka tidsperioder och brott som ska visas. Motsvarande punkter visas sedan i kartan och det är här kartans stora kartografiska brist finns: alla punkter

överlagrar varandra. I figur 7.4 nedan ser en stor del av brotten ut att vara materiella (gröna). Faktum är att upplevelsen ändras med skalnivåerna och helt plötsligt kan punktmolnet vara övervägande blått efter att användaren zoomat in en bit.



Figur 7.4: En karta över brott utförda i San Francisco (SFC 2013).

8. Fallstudie – Sweco GeoAtlas

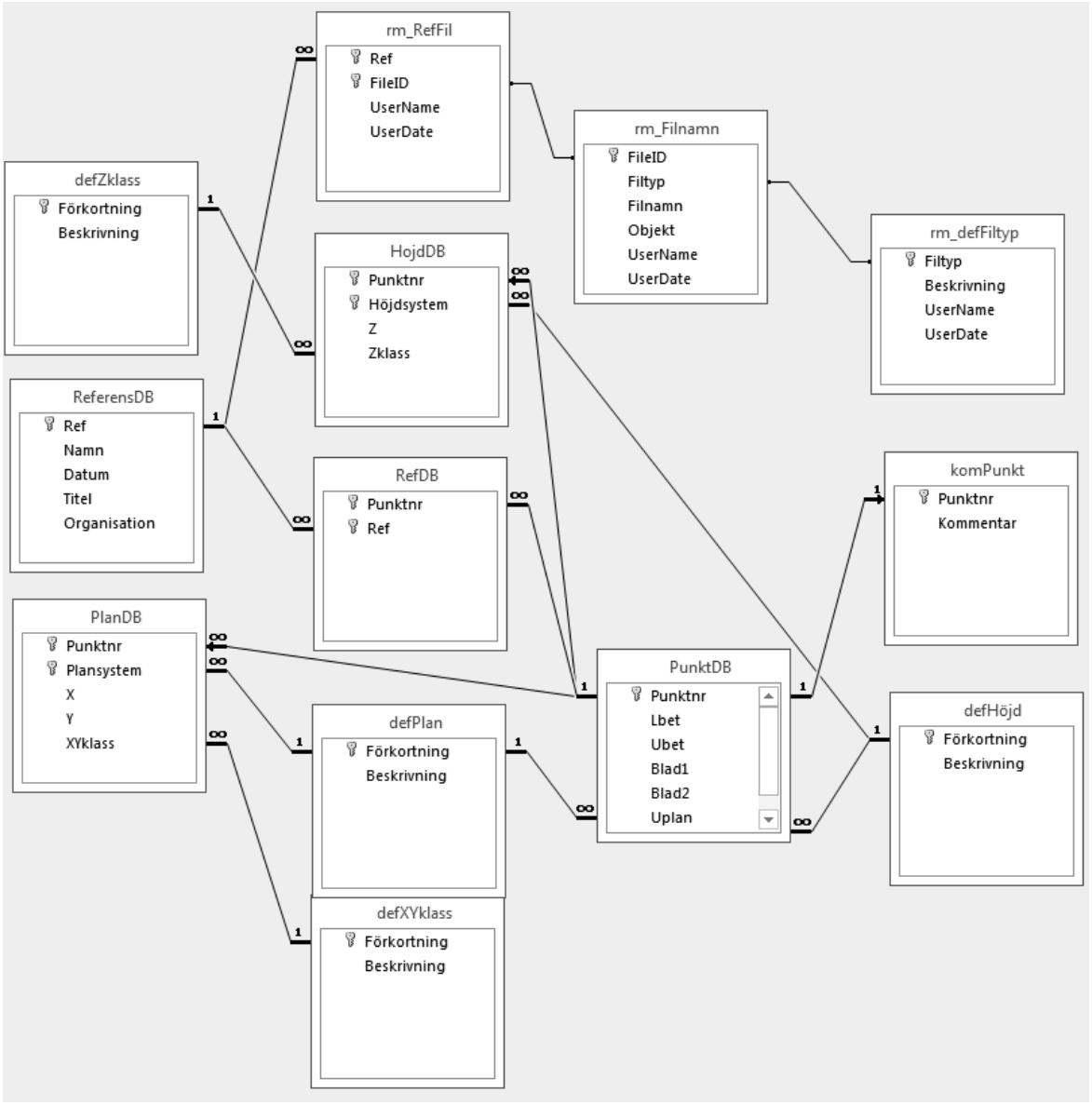
8.1. GeoAtlas idag

GeoAtlas är ett GIS och är bl.a. till för hantering av geotekniska och geologiska data, utförande av grundvattenundersökningar samt markanalyser. GeoAtlas existerar i två förgreningar: Sweco och Malmö. Projektet började som Malmö GeoAtlas i slutet på 90-talet och beställdes då av Malmö stad med Sweco Position som utförare. Därefter har man på Sweco fortsatt utveckla en version för intern användning. Detta exjobb baseras på Sweco-förgreningen men går även att tillämpa på Malmö-förgreningen.

8.1.1. Arkitektur

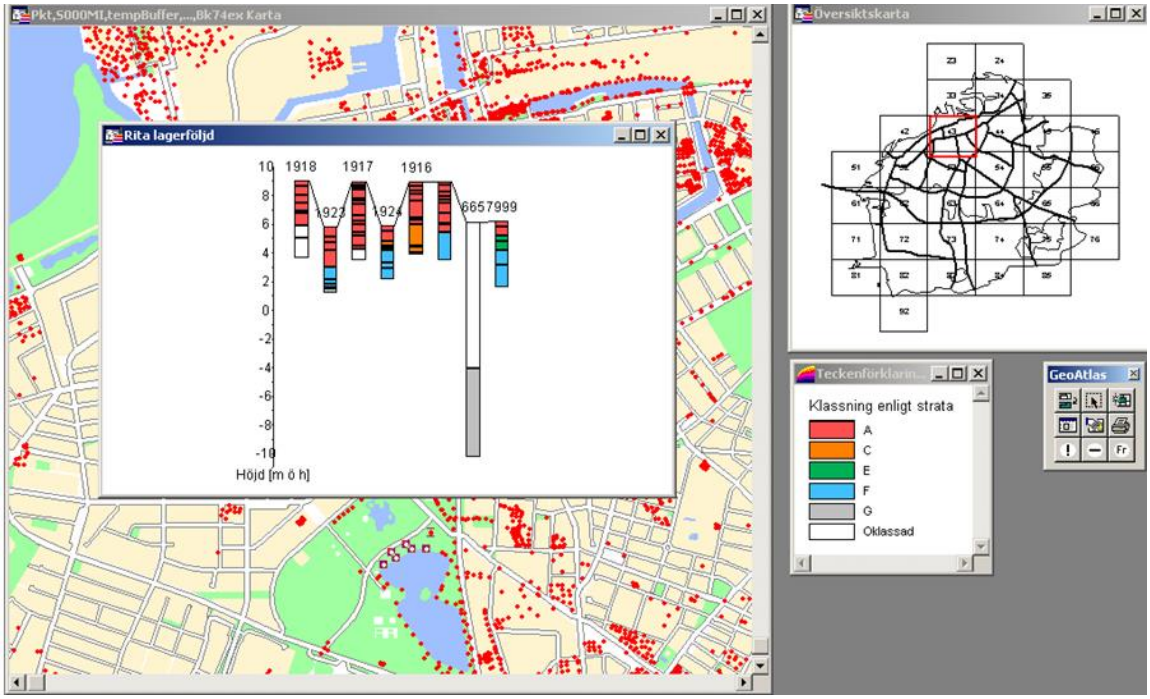
I Sweco GeoAtlas lagras punktdata i en Microsoft Access-databas. Dess struktur är relativt komplicerad med flera tabeller med få attribut (se figur 8.1). Det finns en huvudtabell, *PunktDB*, som innehåller information i form av punktnummer och tidigare beteckningar samt en annan, *ReferensDB*, som innehåller information om referenser. Förhållandet mellan dem är 1-N (en referens kan innehålla flera punkter). Punkttabellen i sig innehåller inte någon information om punkternas läge utan den är sparad i en annan tabell, *PlanDB*. Även detta är ett 1-N förhållande då *PlanDB* innehåller koordinater i flera koordinatsystem för ett punktnummer. Så fortsätter det och vissa tabeller innehåller endast information om ett attribut. Ett exempel är tabellen *komPunkt* som endast innehåller ett kommentarfält om vissa punkter.

En delmängd från databasens struktur syns i figur 8.1. Tabellerna i figuren är (ungefär) de som funnits att tillgå i detta projekt. Hela databasen består av cirka 170 tabeller och hantering av samtliga är både tidsödande och ointressant för denna rapports ändamål.



Figur 8.1: Ett utdrag ur Sweco GeoAtlas databasstruktur.

Microsoft Access-databasen kopplas sedan till GIS-programmet MapInfo som agerar som kartdatabas. Exempel på lagrade kartor är fastighetskartor, historiska kartor och grundvattenkartor. I MapInfo kan man även utföra flera olika analyser samt tolka fram tredimensionella geologiska kartor. En av funktionaliteterna som är tänkt att lyfta ut till webbgränssnittet är visualisering av s.k. borrhålsloggar, vilket är en profilskiss på vilka stratigrafiska lager som finns i ett borrhål. I figur 8.2 syns ett exempel på hur det ser ut i MapInfo när profildata visas.



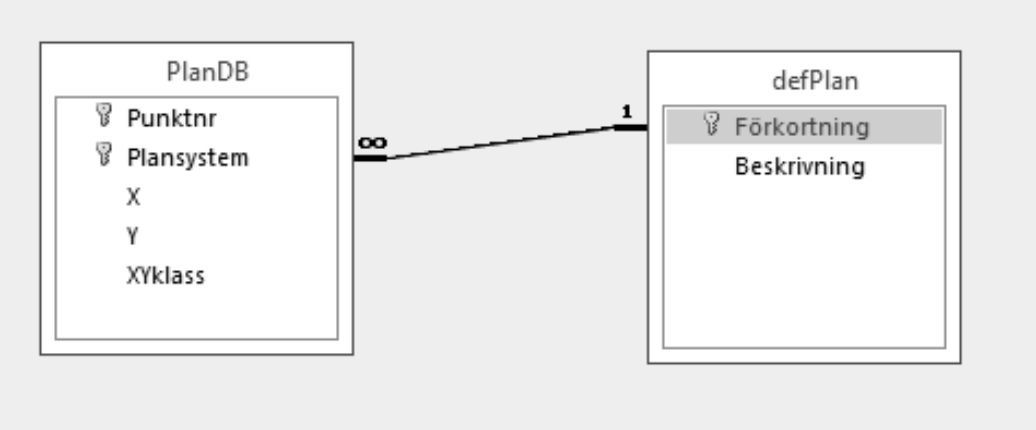
Figur 8.2: Stratigrafisk profilbild för några punkter, utritat i MapInfo.

8.1.2. Metadata

Metadata betyder data om data och beskrivas som datas egenskaper i termer om vem, vad, när, var, varför, och hur. Sådan information lagras ofta separat i nära anknytning till ett datasett. Metadata inom GIS kan vara saker som referenssystem, inmätningskvalité, distributionsvillkor och attributbeskrivningar. En standard för geografisk metadata är *ISO 19115: Geographic information – Metadata* (Semerjian 2008).

Ibland är uppdelningen mellan data och metadata tydlig. I t.ex. fallet med ArcGIS förekommer data i en tabell (DBase-format) medan metadata lagras i XML-filer och i MapInfo:s grundutförande (d.v.s. utan koppling till GeoAtlas) används liknande förfarande. I GeoAtlas finns dock allt lagrat i en Microsoft Access-databas och där är avgränsningen mellan data och metadata något svårgreppad.

Kopplat till en punkt i GeoAtlas finns en stor mängd information. Saker som koordinatsystem, inmätningskvalité i plan och höjd samt ursprung är några exempel. Dessa finns ofta i tabeller som länkas med en siffernyckel till huvudtabeller (se exempel i figur 8.3). I en del tillämpningar kan de tolkas som metadata men i GeoAtlas finns dessa värden specifikt för varje punkt.



Figur 8.3: Relationen mellan den centrala tabellen *PlanDB* och tabellen *defPlan* som innehåller information om inmätningskvalité i planled. *PlanDB.XYklass* har siffervärden, 1-5, och *defPlan.Beskrivning* ger förklaringar till dessa siffror. En sådan förklaring kan t.ex. vara ”Lägesangivelse i plan med +/- 2,5 m felmarginal”.

I GeoAtlas finns fem tabeller som innehåller olika sökbara klassificeringar av data. Dessa har en hierarkisk struktur som möjliggör stegvis specificering av ett dataurval. En användare kan t.ex. söka på ett jordprov för att visa samtliga punkter med ett visst innehåll. Därefter kan specificering dessutom ske så att endast punkter visas där en viss typ av provtagning gjorts. Dessa variabler blir då generella för data som visas och i den enskilda sökningen skulle jordlager och typ av provtagning kunna tolkas metadata.

8.2. Kravspecifikation

8.2.1. Målgrupp

GeoAtlas är en mjukvara som måste installeras och köras lokalt på användarens dator. Mjukvaran är något komplicerad och vanlig användning kräver att man är väl insatt i systemet. I praktiken innebär detta att på ett kontor med cirka 40 anställda så är kanske fem personer experter eller tillräckligt insatta för att använda sig av programmet. Syftet med en webbapplikation är att så många som möjligt ska kunna använda sig av geoteknisk data utan någon inlärningströskel. Målgruppen är således icke-experter och vid utformning av användargränssnittet bör detta tas i beaktande.

8.2.2. Funktionalitet

Det finns många sätt att använda GeoAtlas på och att lyfta ut all funktionalitet till ett webbgränssnitt är inte aktuellt i denna studie. För att en webbapplikation ska vara till någon nytta finns dock vissa saker som måste finnas med. Nedan följer de krav som ställs från Sweco Position på en webbaserad miljö för visning av borrhålsinformation.

- Funktioner i kartan
    - zooma in och ut
    - panorera
    - (mäta längd).

- Funktion i gränssnittet
    - Sökning på punktnamn. Selekterar punkten i kartan och visar attributet i listan.
    - Sökning på attribut. Resultatet listar de punkter som innehåller det sökta attributet.
- Selektering av punkt i kartan
    - visa attribut
    - visa stratigrafiska data som uppritad profil
    - visa referensdata. Klickbar länk för visning av referensuppgift som PDF.
- Lägga till nya punkter till datasetet
    - genom att trycka på en position i kartan
    - genom att mata in koordinater.
- Exportverktyg för punktdata
    - enstaka punkter
    - för flera punkter inhägnade av en polygon.
- Funktionalitet kring geologisk 3D-modell
    - visa stratigrafiskt tvärsnitt
    - generera fiktiva borrkärnor.
- Utskrift
    - möjlighet att skriva ut kartfönstret tillsammans med attribut för selekterad punkt.
- Karta ska innehålla följande
    - bakgrundskarta i färg som ger en god möjlighet att orientera sig för att hitta valt borrhål i terrängen
    - norrpil
    - skalstreck.

8.2.3. Användargränssnitt

Utöver de konkreta kraven som ställs på funktionaliteten finns även andra saker att ta hänsyn till. Allt som läggs till i kartan måste vägas mot kostnaden i utrymme och graden av oordning det medför. Exempelvis är norrpil något som är viktigt i en papperskarta medan det i en tvådimensionell webbkarta antas vara så att norr är uppåt och väster är till vänster. På mindre skärmar blir alternativkostnaden för hög och således finns liten nytta med att införa en sådan. Komponenter som däremot alltid behövs är skalstreck, zoom-funktionalitet och även någon form av översiktskarta. Idén med det sistnämnda är hämtad från gränssnittet i Sweco GeoAtlas (se figur 8.2). Behovet av en översiktskarta finns eftersom man som användare ofta arbetar i stor skala. Det blir då enklare att navigera sig om man, på en mer översiktlig nivå, ser kartfönstrets position. En följd av det sistnämnda är även att behovet av en ”hoppa till ursprunglig vy”-knapp finns.

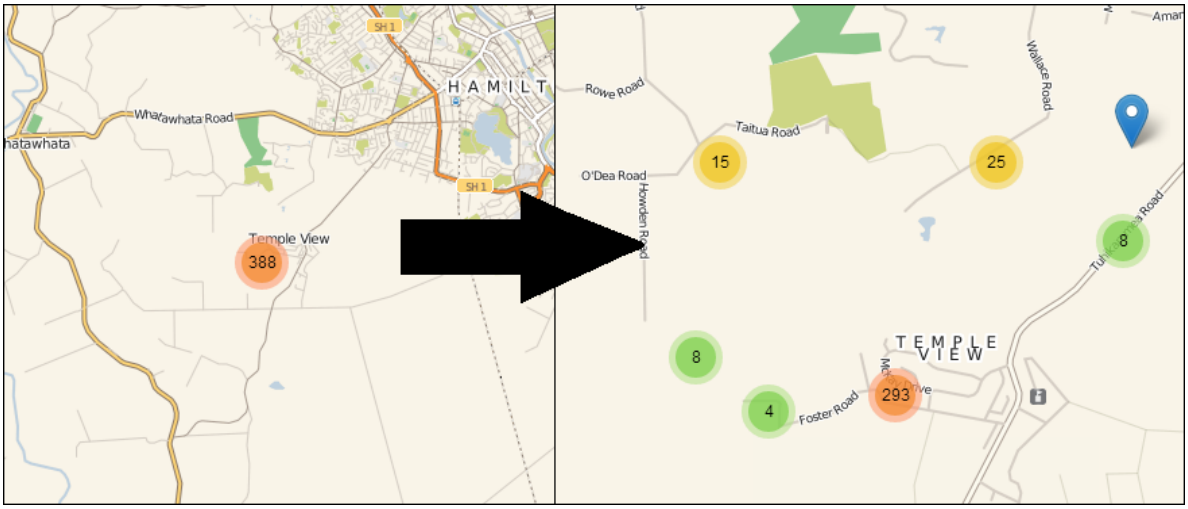
8.3. Teknikval

8.3.1. Klientteknik

Det visade sig tidigt att HTML5 skall ligga till grund för fallstudien. Den största anledningen är att hålla dörrar öppna: en eventuell 3D-implementering skulle i så fall innebära införande av Canvas och framförallt WebGL eller W3DS. Större delen av applikationens grundutförande är dock skrivet i JavaScript (som f.ö. WebGL också baseras på) vilket är en nödvändighet vid implementering av dynamiska hemsidor. En viktig egenskap är att man, med JavaScript, kan påverka objekt i dokumentet utan att uppdatera hela sidan. För en kartapplikation innebär detta att delar av sidan kan ändra utseende utan att kartan nollställs, d.v.s. genereras på nytt och centreras vid en hårdkodad koordinat med en förutbestämd zoom-nivå.

8.3.2. JavaScript-bibliotek

Vid teknikval har initialt en grundfaktor prioriterats mer än andra, nämligen det ingående punktlagrets densitet. Många tusen företeelser ska trängas på en liten geografisk yta och utgångsläget för en prototyp bör därför vara ett generaliseringsverktyg som underlättar den kartografiska framställningen. Denna insikt är en följd av tidigare studerade kartprodukter (se kapitel 7.). Exempel på klusterfunktioner finns bl.a. som insticksmodul till kartverktyget Leaflet (se figur 8.4) och OpenLayers.



Figur 8.4: Insticksmodul som möjliggör klusterbildning i en Leaflet-karta. Till höger syns en in-zoomad vy av klustret till vänster (Modifierad från Leaflet 2013b).

Det finns en långt gången diskussion på webben kring huruvida Leaflet eller OpenLayers är det bästa API:t för kartframställning. OpenLayers bibliotek är betydligt mognare (funnits i ca 7 år jämfört med Leaflets 3 år) och har gott om dokumentation. Leaflets styrka är å andra sidan enkelheten och prestandan, exempelvis är API:t 118 kb jämfört med 753 kb för OpenLayers. I dagsläget är utvecklingsprocessen för nästkommande version av OpenLayers (version 3) i ett alfa-stadie och dess införande kommer innebära större kompabilitet med

HTML5-tekniker så som WebGL, vilka är intressanta för detta projekt. Om OpenLayers 3 hade funnits som stabil version hade kanske valet blivit annorlunda men i dagsläget är Leaflet vad som anses vara bäst lämpat för detta projekt med avseende på funktionalitet och estetik.

8.3.3. Bakgrundskarta

De bakgrundskartor som finns att tillgå har sina för- och nackdelar. *Stamen Toner* (se avsnitt 6.3.3.) är avskalad och passar väl till tillämpningar som innehåller ett stort antal företeelser i förgrunden. I detta fall begränsas dock användarens navigeringsförmåga för mycket. Som geolog behöver man kunna fastställa vart en punkt på kartan finns i verkligheten, vilket kan vara svårt i en allt för enkel karta. Google-kartan (se avsnitt 6.3.2.) är ett bättre alternativ i dessa avseenden och väl lämpad för ändamålet. Den är i stort sett gratis för privat bruk men i kommersiella syften tillkommer licenskostnader.

En karta som inte diskuterats tidigare är Fastighetskartan. Fastighetsinformation är något som hade förbättrat applikationen och användarens navigeringsförmåga ute i fält. Sådan information finns inte i andra kartprodukter och Fastighetskartan i sig är dyr. Tills vidare får fastighetsinformation istället ses som ett önskvärt tillägg till applikationen.

Open Street Map (se avsnitt 6.3.1.) är helt gratis att använda och ganska snarlik Google Maps, dock med vissa estetiska olikheter. På stora skalnivåer finns information som kan tänkas vara överflödig för ändamålet, som t.ex. namn på olika restauranger. Visningsstilen är dock diskret och faktum är att sådan informations existens hjälper geologer att hitta brunnars faktiska placering (i verkligheten) mycket enklare. Därmed är även Open Street Map ett bra alternativ och skillnaden i pris jämfört med Google-kartan gör att den används i fallstudien.

8.3.4. 3D-funktionalitet

För att kunna införa stöd för tvärsnittsvyer, fiktiva borrhål o.s.v. måste någon form av 3D-funktionalitet implementeras. I avsnitt 5.2. tas WebGL upp samt W3DS. Det finns en stor skillnad mellan dessa i att W3DS i sig är ett protokoll för inhämtande av företeelser. Det innebär att hela kartan i sig blir tredimensionell (se figur 5.3). I denna fallstudie passar det bättre med en ”vanlig” tvådimensionell karta som bas så att grundläggande funktionalitet i kravspecifikationen inte äventyras. För 3D-funktionalitet är då WebGL ett bättre alternativ eftersom den kan fungera som en fristående modul med koppling till kartan, d.v.s. att 3D-modulen blir följsam med användarens panoreringar. När sedan användaren trycker i den vanliga kartan skickas koordinaterna till 3D-modulen, varpå data kan visas från en fördefinierad 3D-modell.

8.3.5. Arkitektur

I avsnitt 5.4. tas två olika alternativ upp för applikationens uppbyggnad. Båda har för- och nackdelar men i denna fallstudie har enkelheten prioriterats. Målsättningen är att ha en applikation att faktiskt kunna visa upp och därför var det nära till hands att implementera en karta med en lokal lagringsfil som GeoJSON. Valet föll på denna metod eftersom utbytesformatet i en WFS-tjänst även kan vara GeoJSON. Således bör steget till en fullt fungerande WFS-tjänst inte vara allt för stort om applikationen vidareutvecklas i framtiden.

Vid små datamängder är det möjligt att en GeoJSON-lösning har lägst tidskomplexitet men i fallstudien uppgår antalet punkter till cirka 35 000. Applikationen blir då kostsam att starta upp, ett fenomen som inte uppstår i WFS-tjänster eftersom databasen läses in på serversidan och endast ett urval av företeelser skickas till klienten.

Utan en PHP/AJAX-lösning i samband med GeoJSON-alternativet måste ett script köras för att uppdatera data i applikationen. I dataförädlingsfasen användes FME för att konvertera data från Microsoft Access-databasen till en GeoJSON-fil. Samma arbetsyta i FME skulle kunna användas i en tidsinställd scriptkörning på servern för att ajourhålla GeoJSON-filen. De data som behandlas i fallstudien är inte av sådan karaktär att den behöver ajourhållas minutiöst. En daglig uppdateringsfrekvens är fullt tillräcklig vilket innebär att metoden med lokal lagring är ett rimligt alternativ.

8.3.6. Avstämning mot kravspecifikation

Av de teknikval som är gjorda kan JavaScript och HTML5 användas till att implementera större delen av punkterna i kravspecifikationen. Undantagen är modifiering av data, exportverktyg samt 3D-funktionalitet.

Nya punkter kan med rådande teknikval endast läggas till i GeoAtlas och data i webbapplikationen kan därefter t.ex. uppdateras med hjälp av FME (se avsnitt 8.3.5.).

Ett exportverktyg i form av en utskriftsvänlig sida kräver införandet av ett Canvas-element. Canvas är visserligen en del av HTML5 och för att hämta eller modifiera innehållet i ett Canvas-element används JavaScript (se diskussion om utskriftsvänlig sida i avsnitt 9.7.).

Förutsättningarna ovan kan även tillämpas på 3D-funktionaliteten. Inslag av WebGL (som är en del av HTML5) krävs och JavaScript används då som programmeringsspråk.

En prototyp har således alla förutsättningar att uppfylla kravspecifikationen i sin helhet. Eventuella avvikelser från kravspecifikationen kommer diskuteras senare i rapporten.

8.3.7. Licenser

De teknikval som är gjorda är hämtade från olika håll. Ett problem som kan uppstå är att de innefattas av olika licenser vilket kan ha en inverkan på hur slutprodukten får användas. Vissa licenser innebär t.ex. att slutprodukten i sig måste vara fri att kopiera. Nedan redogörs för de villkor som gäller för teknikvalen (där det är applicerbart).

För bakgrundskartan *Open Street Map* gäller licensen *Creative Commons Attribution-ShareAlike 2.0* (CC BY-SA). Bakgrundskartan är fri att kopiera utan undantag så länge korrekt referens till upphovsrättsinnehavaren och licensvillkoren ges (CC 2013).

JavaScript-biblioteket Leaflet ges under en egenkomponerad öppen källkod-licens. Källkoden är fri att återanvända så länge referens ges till upphovsrättsinnehavaren tillsammans med en s.k. *disclaimer*. En *disclaimer* är ett sätt att friskriva skaparen från juridiskt ansvar gentemot en användare (Leaflet 2013c).

Insticksmodulen till Leaflet som hanterar klusterbildningen går under en MIT-licens som innebär att modulen är helt fri att använda och distribuera (Leaflet 2013a).

Ovan nämnda komponenter medför inga licensrestriktioner i den slutprodukt som de ingår i. Så länge referenser ges enligt ovan finns det inga hinder att tillverka en stängd prototyp med de teknikval som gjorts.

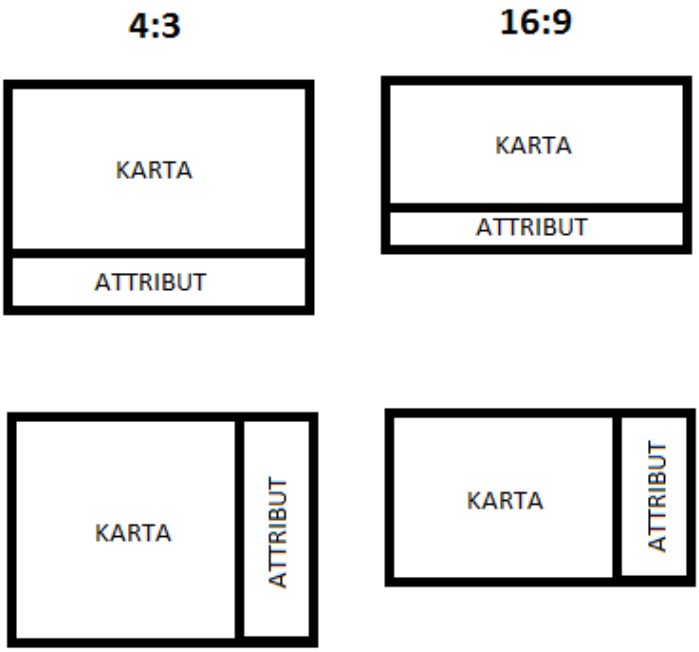
8.4. Placering av information i skärmen

The Geology of Britain viewer har tidigare uttryckts som ett idealexempel för visning av geotekniska data. Skillnaden mot fallstudien är att större vikt kommer läggas på öppen källkod-alternativ. Att ha ett bra exempel att utgå från är givetvis en stor fördel men en eventuell implementering kommer därmed garanterat ha stora avvikelser i jämförelse.

Till varje punkt finns en rad olika attribut som behöver presenteras vid interaktion. Eftersom kartfönstret kommer innehålla mycket information är det fördelaktigt att visa punkternas attribut i ett avskilt område utanför kartan. I andra exempel (se kap 7.) har poppuppfönster används men när det rör sig om många punkter är det lätt att en sådan ruta endast är i vägen. Ett annat alternativ är att attributen visas i ett nytt fönster, men en sådan lösning är inte användarvänlig.

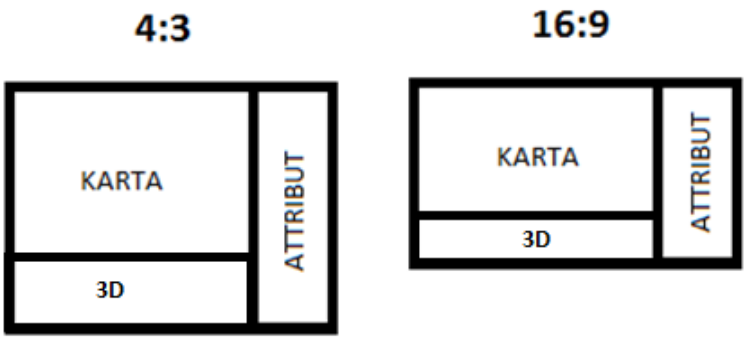
Själva kartbilden skall utgöra så stor del som möjligt av sidan. Detta leder till att punkternas attribut måste visas i sidans utkant, antingen som en remsa som är horisontell eller vertikal bredvid kartan. Valet av sida är direkt beroende av hur främst datorskärmar ser ut idag. Äldre skärmar har vanligtvis bildförhållandet 4:3 (bredd:höjd) medan nyare skärmar använder förhållandet 16:9. Vid breda bildförhållanden bidrar en horisontell attributremsa

till ett smalt och avlångt kartfönster (se överst i figur 8.5), medan en vertikal istället bidrar till en mer kvadratisk, behaglig, form på kartfönstret.



Figur 8.5: Skiss på ett kartfönster i bildförhållandet 4:3 och 16:9. Attributremsan utgör 25 % av fönstret bredd/höjd.

Om inte själva kartan är i 3D som i fallet med W3DS-tjänsten så är det troligt att 3D-funktionalitet istället uppnås med WebGL-teknik i en extern modul. Ett tänkbart scenario är att visa ett tvärsnitt över de stratigrafiska lagren. En sådan funktion skulle i så fall ha horisontell orientering. Dess placering i fönstret kan tänkas vara enligt figur 8.6 nedan. Om större utrymme behövs skulle attributfönstret kunna minimeras för att ge större 3D-yta.



Figur 8.6: Ett tidigt förslag på hur en 3D-funktionalitet kan placeras.

8.5. Prototyp

Baserat på de teknikval som gjorts har en prototyp tillverkats. Inom projektets ram har det endast funnits möjlighet att skapa en enkel prototyp. För att kompensera detta tas sedan brister och påbyggnadsmöjligheter upp för diskussion.

8.5.1. Avstämning mot kravspecifikation

Nedan följer en lista som stämmer av prototypens funktionalitet mot de krav som fanns i kravspecifikationen. Av 18 krav har 6 uppfyllts och i listan markeras de avklarade kraven med symbolen ✓.

- Funktioner i kartan
    - ✓ zooma in och ut
    - ✓ panorera
    - (mäta längd).
- Funktion i gränssnittet
    - ✓ Sökning på punktnamn. Selekterar punkten i kartan och visar attributet i listan.
    - Sökning på attribut. Resultatet listar de punkter som innehåller det sökta attributet.
- Selektering av punkt i kartan
    - ✓ visa attribut
    - ✓ visa stratigrafisk data som uppritad profil
    - ✓ visa referensdata. Klickbar länk för visning av referensuppgift som PDF.
- Lägga till nya punkter till datasetet
    - genom att trycka på en position i kartan
    - genom att mata in koordinater.
- Exportverktyg för punktdata
    - enstaka punkter
    - för flera punkter inhägnade av en polygon.
- Funktionalitet kring geologisk 3D-modell
    - visa stratigrafiskt tvärsnitt
    - generera fiktiva borrkärnor.
- Utskrift
    - möjlighet att skriva ut kartfönstret tillsammans med attribut för selekterad punkt.
- Karta ska innehålla följande
    - ✓ bakgrundskarta i färg som ger en god möjlighet att orientera sig för att hitta valt borrhål i terrängen

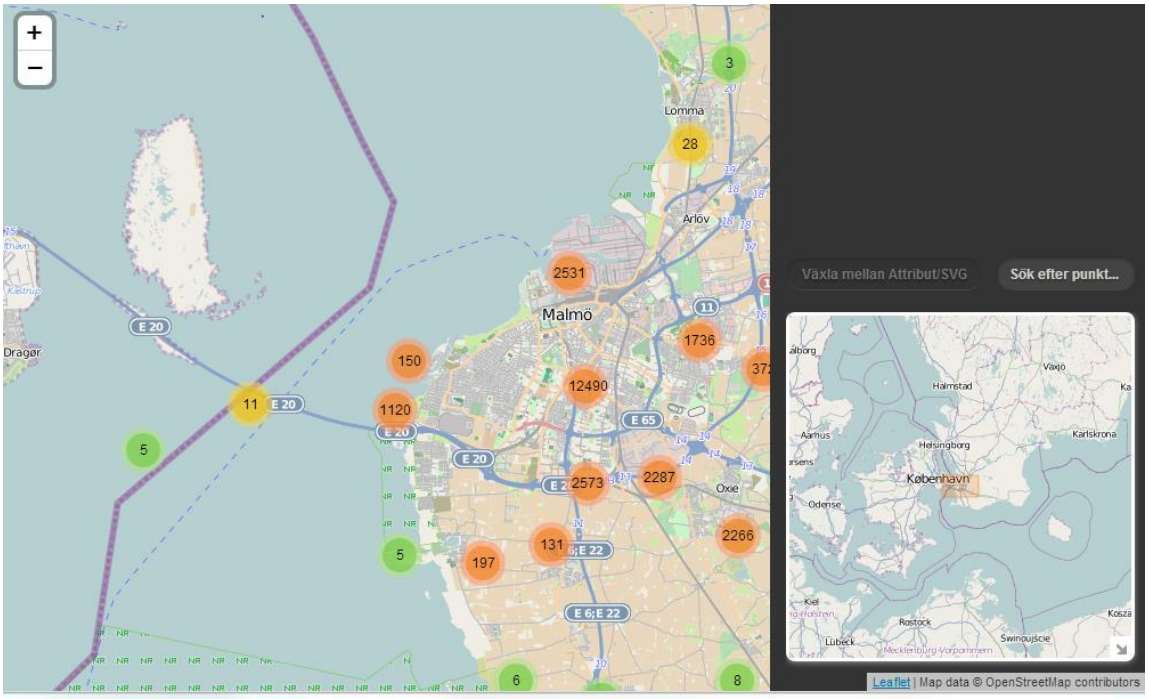
- norrpil
- ✓ skalstreck.

I samband med vissa av kraven ovan har tekniska motgångar stötts på. Följden har blivit att lösningen på dessa krav ligger utom räckhåll inom ramen för denna studie. Kraven som främst berörs är de om tillägg av nya punkter i kartan samt funktionalitet kring geologisk 3D-modell. I grund och botten är det en tidsfråga i båda fall. Beträffande punktinteraktion var utgångsläget från början att lägga upp en WFS-tjänst, så blev dock inte fallet när valet av arkitektur granskades närmare.

3D-funktionalitet är något som under tidens gång fallit ur fokus mer och mer. En stor anledning är att någon geologisk 3D-modell inte fanns tillgänglig över Malmö när projektet begav sig. Därefter tillkommer även tidsfaktorn och den stora tröskel som finns i att programmera i WebGL-miljö. Sammantaget gör dessa faktorer att 3D-funktionalitet endast får ses om en möjlig påbyggnad och inte som en inkluderad del i detta projekt.

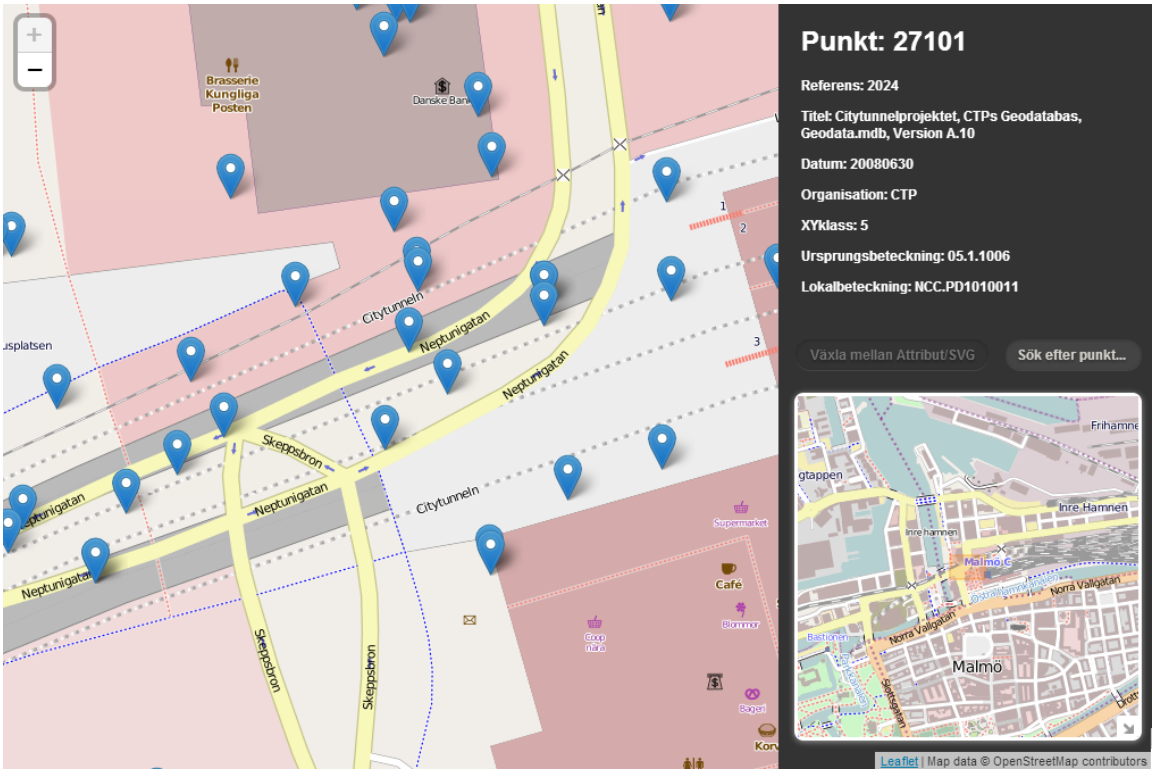
7.5.2 Genomgång av prototyp

Sidan startas från en webbläsare och det första som händer är att användaren får vänta på att alla resurser laddas. Detta tar normalt några sekunder och mest tid går till att ladda listan med alla punkter. När allt är färdigladdat ser den startade applikationen ut enligt figur 8.7 nedan. Färdigt att användas finns en huvudkarta, en översiktskarta och ett sökfält. Härifrån kan man antingen zooma/panorera sig vidare eller mata in ett punktnummer i sökfältet. Kartan består av många tusen punkter men endast ett fåtal syns, dessa är kluster innehållandes andra punkter med en stil som antyder hur många punkter som ”gömmer sig”.



Figur 8.7: Startvyn i applikationen.

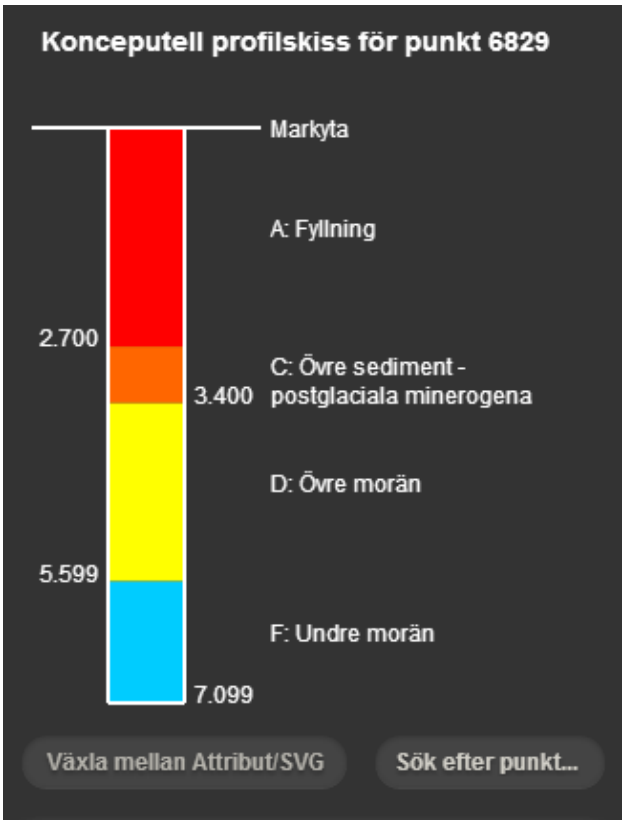
Därefter jobbar man sig genom skalnivåerna och vid dom två sista ersätts samtliga kluster med markörer för varje punkt. Om man klickar på en sådan dyker attributen för punkten upp till höger i bild, se figur 8.8 nedan.



Figur 8.8: Attributen visas för punkt 27101 i applikationen.

För vissa punkter finns stratigrafisk profildata inlagt och i GeoAtlas redovisas denna information enligt figur 8.2. Inspiration har hämtats från detta exempel (och *The Geology of Britain Viewer*) och resultatet i prototypen blev enligt figur 8.9. I de fall stratigrafisk profildata finns att tillgå genereras en SVG-bild. Lagren är skalriktiga och har färger i enighet med färgschemat från GeoAtlas. Förutom lagerbeskrivning anges siffror som anger vilket djup lagret slutar på (markyta = 0 m).

Knappen ”Växla mellan Attribut/SVG” är endast aktiverat när en SVG går att generera. Om användaren, efter att ha tagit upp en SVG-bild klickar på kartan avaktiveras knappen och SVG-filen försvinner.



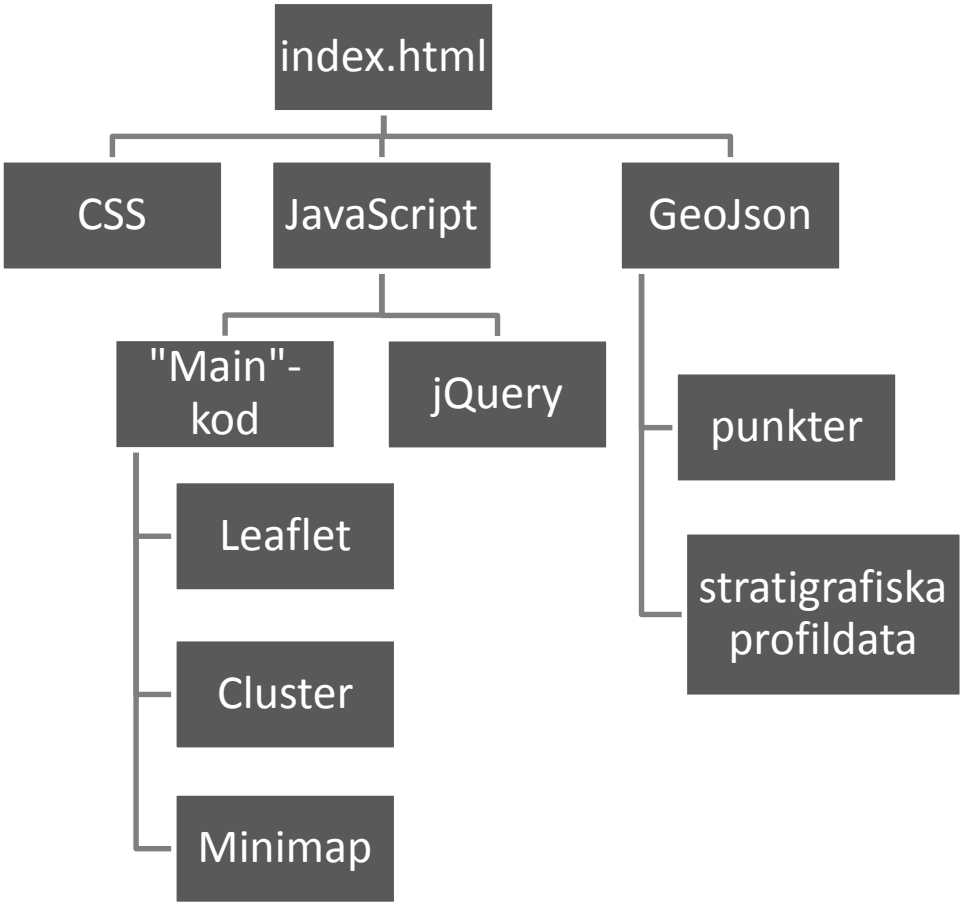
Figur 8.9: Stratigrafiska lagerföljden i kartapplikationen består av en genererad SVG-bild.

Den vilande texten i sökfältet är alltid ”Sök efter punkt…”. När ett värde skrivs in, och användaren trycker på *enter*-knappen på tangentborden, utförs en linjärsökning i listan med punkter. Vid en träff returneras positionen för denna och kartan centreras över markören, samt att en röd cirkel ritas ut i ett hårdkodat antal sekunder. Om ingen träff hittas ändras texten i sökfältet till ”Ingen träff” i några sekunder, och byts sedan tillbaka till ”Sök efter punkt…”.

8.5.2. Filstruktur

Rotfilen i prototypen är index.html, den innehåller inte mycket kod utan fungerar snarare som en länk till övrig kod (se figur 8.10). Stilar, JavaScript-bibliotek och företeelser i GeoJSON-format kallas alla från *index.html*-filen. Förutom detta finns även SVG-figurens stomme lagrad i rotfilen. Innan applikationen kör igång består figuren endast av sträck för markytan och ytterkanterna för borrhålet.

JavaScript-filen *main.js* är platsen för en stor mängd av den egenkomponerade koden (se Appendix A). Här skapas exempelvis kartan, översiktskartan och klustren. All funktionalitet kring knappar finns uttryckta här tillsammans med hantering av företeelser i kartan. Exempel på sådan hantering är saker som ska hända med fönstret när användaren trycker på en markör.



Figur 8.10: Filstrukturen i prototypen. Rotfilen index.html kallar på filer i filsystemet av typen CSS, JavaScript och GeoJson.

8.5.3. Dataförädling

Vid användning av en arkitektur som i avsnitt 5.4.1 måste en databas finnas i grunden. Prototypen i fallstudien kräver dock lagring av data i ett utbytesformat som JSON eller XML. Viss handpåläggning behöver ske eftersom testdata är i en Microsoft Access-databas med en stor mängd tabeller. Som verktyg för att lösa detta har FME (Feature Manipulation Engine) använts. I FME kan tabeller slås ihop (motsvarande JOIN i SQL) på ett enkelt sätt och från start till slut har cirka 10 olika sammanslagningar skett där de, för applikationen, viktigaste attributen bevaras. Förutom detta har även en geometrikolumn skapats utifrån koordinaterna som var lagrade i Access-databasen.

Vid framtida användning skulle detta steg kunna automatiseras för att ajourhålla data, med eller utan FME. FME är inte gratis men används i stor utsträckning bland verksamhetsutövare inom GIS. Dessutom finns gratisversioner (som t.ex. GeoKettle) som kan fungera som alternativ. Metodiken skulle då kunna vara en tidsinställd batch-fil som startar upp mjukvaran och ersätter GeoJSON-filen på servern.

9. Diskussion

9.1. HTML5

När projektet började var tanken att HTML5 skulle vara kärnan för all problemlösning. Det visade sig att HTML5 blev sekundärt medan nästan all programmering skedde i JavaScript. Införandet av HTML5, CSS3, WebGL och Canvas har möjliggjort en helt ny dimension inom webbdesign. Inom ramen för den efterforskning som gjorts är uppfattningen att kartor med större 3D-funktionalitet bör vara på ingång. Dessvärre saknades kunskap och tid till att på riktigt sätta sig in och implementera en sådan funktionalitet i detta projekt. Istället får diskussion föras kring hur en sådan karta hade kunnat se ut och fungera, vilket givetvis inte har samma bärkraft jämfört med en faktisk implementation.

9.2. Leaflet/OpenLayers

I avsnitt 8.3.2. har en diskussion förts kring huruvida Leaflet eller OpenLayers skall tjäna som karttjänst i fallstudien. Valet föll på Leaflet och motiverades med att dess klusterfunktion har bättre funktionalitet och estetik. Kortare resonemang fördes även kring kompabilitet med HTML5. Där har OpenLayers 2 har svagt stöd medan Leaflet är något bättre.

Vid exjobbets början var OpenLayers 3 i ett alfa-stadie medan det i slutskedet ligger i ett beta-stadie. OpenLayers 3 håller på att presentera sig på allvar och ska uttryckligen ha stöd för Canvas och WebGL redan från början i utvecklingsarbetet. På deras hemsida talar man om en konvergens mellan 2D och 3D, vilket i största grad skulle vara intressant i den egna fallstudien. Om beta-stadiet varit förutsättningen vid exjobbets början hade kanske teknikvalen sett annorlunda ut.

9.3. Kluster

9.3.1. Server/klient

Som tidigare nämnt är klusterfunktionalitet en sak som prioriterats i prototypens utformande. En fungerande sådan har tagits fram men det finns plats till förbättring. Kluster-modulen löser ett problem i att visning av kvantitativ data blir enklare. Den skapar även ett problem i att applikationen blir långsammare. Detta beror på att klusterbildningen sker i realtid på klientsidan. Motsatsen är att sköta sådana beräkningar på serversidan men vilket som är bäst är inte helt självklart. På serversidan skulle kluster kunna tillverkas på begäran och skickas till klienter. Det skulle göra klienterna mindre beroende av att ha hög processorkraft, men samtidigt belastas servern istället och risken är att nettoeffekten inte blir positiv. Detta beror helt på hur många användare applikationen har samtidigt. Beräkningar på serversidan skulle dock även innebära införande av nya tekniker så som PHP-programmering, vilket gör att alternativet är uteslutet i just denna fallstudie.

Om klusterbildning sker på klientsidan kommer klienten vänta på att företeelser skickas från servern för att sedan behandla dessa till kluster innan visning sker. Processen kan vara kostsam vid högt antal företeelser men en stor fördel är att klienten kan arbeta på olika skalnivåer utan att göra ytterligare anrop till servern. Efter att företeelser är laddade av klienten är det upp till klientens processorkraft hur pass bra applikationen flyter på, d.v.s. hur smidigt det går att skapa nya kluster i varje skalnivå.

9.3.2. Andra alternativ

Att skapa kluster i realtid är en kostsam åtgärd oavsett om den sker på server eller klientsidan. Ett alternativ är att istället skapa kluster på förhand och spara dessa i egna ingående lager i applikationen. Dessa kan sedan tändas/släckas beroende på vilken skalnivå användaren befinner sig i. Detta är en relativt enkel lösning som eliminerar en stor mängd realtidsberäkningar som sker i dagsläget. Nackdelen är att uppstartsfasen skulle bli marginellt långsammare och att den animering som finns i dagsläget upphör fungera. Tester har inte genomförts men en sådan metodik skulle troligtvis kunna innebära en märkbar prestandaförbättring i applikationens löpande användning.

9.4. Brister och buggar

Prototypen får i dagsläget ses som påbörjad. Många centrala funktionaliteter är färdigimplementerade men generellt måste många detaljer fixas till. Ett exempel är utskrivningen av en punkts attribut. I dagsläget är koden skriven så att attributnamnet hämtas från en av GeoJSON-filerna. Sedan skrivs attributnamnet ut tillsammans med attributens värde för varje företeelse. Detta är dock intetsägande i vissa fall, exempelvis attributet ”XYklass” som kan ha värdet ”1”. Det är tidigare uttryckt att målgruppen för applikationen är icke-experter. För att uppnå detta måste attributnamnen ändras till något som fungerar bättre i löpande text, exempelvis: ”Lägesangivelse i plan med +/- 2,5 m felmarginal” (vilket motsvarar XYklass=4).

Det finns olika typer av punkter i datasetet och alla innehåller inte stratigrafiska profildata. Punkttyperna är i dagsläget inte ett ingående metadata i testdata och därför är det svårt att särskilja vad som är vad. De som innehåller eller inte innehåller stratigrafiska profildata går dock att skilja på (med linjär tidskomplexitet), vilket även bör göras. I ett enkelt fall skulle färgen på markören vara skillnaden men i ett fulländat skede bör även symbolerna vara något mer förklarande än vad de är idag.

9.5. Tidskomplexitet

Punkterna laddas in i kartan direkt via kluster-modulen vars källkod är lång (cirka 2000 rader) och komplicerad. En effektivisering av koden i den änden skulle vara för tidskrävande och diskuteras därför inte vidare. Istället läggs fokus på den egenproducerade koden.

I övriga delar av applikationen sker inga större beräkningar förutom när användaren trycker på en markör eller använder sig av sökrutan. Av dessa är det förstnämnda den ”dyraste” åtgärden. En funktion står och lyssnar på en markörklickning och när det inträffar händer följande:

1. Hämta alla attribut för den klickade markören (punktnummer, referensnummer o.s.v.) och uppdatera attributfönstret.
2. Kontrollera om stratigrafiska data finns och uppdatera i så fall SVG-figuren.

Steg 1 är inte kostsam eftersom den klickade markören redan har en referens till ett objekt i punktlistan. Steg 2 däremot kräver att punktnumret i fråga jämförs mot alla rader i den lista som innehåller stratigrafisk data. Följande förutsättningar råder:

- Tabellen innehåller 40840 rader.
- Varje punktnummer kan finnas ungefär 0-5 gånger i tabellen.
- Om flera stratigrafiska lager finns är det inte en garanti att de är i höjdordning i tabellen.

Detta medför att en kostsam algoritm påbörjas enligt följande:

- Utför linjärsökning i listan.
    - För alla träffar: Lägg till träffen i en undansparad vektor.
- Sortera vektorn efter fallande höjd.
- Aktivera ”Växla till SVG”-knappen och rita ut SVG-figuren.

Ett värsta-fall för första punkten är att hela listan måste sökas igenom utan träff till en kostnad av O(*n*) där *n* är antal punkter i tabellen med stratigrafisk data. Övriga punkter i algoritmen ovan utförs endast på ett begränsat antal punkter, max cirka 5 stycken, och är därför obetydliga för tidskomplexiteten.

I fallet ovan samt i fallet med sökrutan sker linjärsökningar vilket inte är den mest effektiva sökmetoden. I många fall går det att ersätta linjärsökningar med binärsöknigar. Ett sådant exempel är sök-algoritmen. I befintlig algoritm skriver användaren in ett punktnummer och sedan jämförs det värdet med samtliga punktnummer i listan i stigande ordning. Detta medför att sökning efter punkt 1 går snabbare än en sökning efter punkt 23400 (förutsatt att punk 1 existerar, annars inträffar *worst case*: O(*n*)). I fallet med binärsökning krävs att listan är sorterad. Sökalgoritmen börjar i mitten på listan och jämför med det sökta värdet. Om det sökta värdet t.ex. är större fortsätter sedan sökningen rekursivt i listans övre halva. Med binärsökning är tidskomplexiteten O(log *n*) i värsta fall.

9.6. Metadatavisning

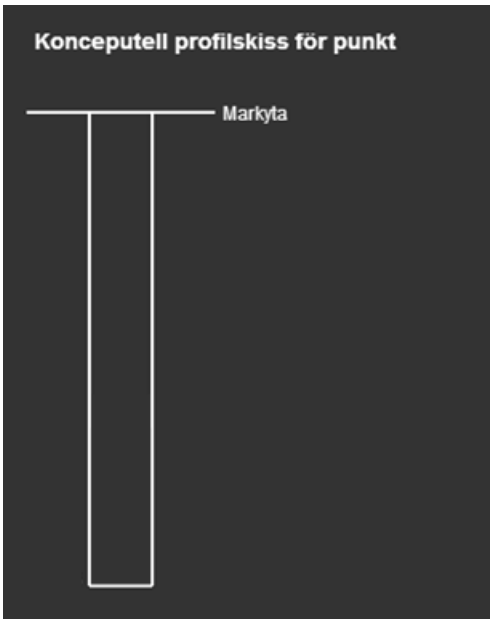
I prototypens befintliga form finns inget behov för separat visning av metadata. Tolkningen kring vad som faktiskt är metadata i GeoAtlas har varit något oklar. Ett sätt att tolka det på är att metadata är beskrivande data som är gemensam för hela datasetet. I sådana fall blir inte en diskussion om metadatavisning aktuell innan en utökad sökfunktion implementeras. Den skulle kunna tillåta användaren filtrera punkter utifrån sådan data som beskrivs sist i avsnitt 8.1.2. Implementering av en sådan funktion är inte orimlig om prototypen används i framtiden. Ett tänkbart utseende skulle i så fall kunna vara analogt med attribut/SVG-knappen. I sidans svarta fält visas då, istället för attribut eller SVG-figur, en sökfunktion med t.ex. rullistor för de olika sökalternativ som finns. Kartfönstret skulle sedan anpassas till de gjorda valen och endast visa t.ex. borrningar som är utförda senast år 2000 och innehåller attributet ”undermorän”.

9.7. Utskriftsvänlig sida med Canvas

En av punkterna i kravspecifikationen var att kunna skriva ut (egentligen exportera) data från applikationen. Sidans standardutformning är inte utskriftsvänlig eftersom attribut och stratigrafiska profiler inte kan visas samtidigt. Ett exportverktyg behöver dels kunna visa all punktrelaterad information och även spara en bild från kartfönstret. Att programmera en sådan funktion från grunden skulle ta lång tid men det finns JavaScript-bibliotek som är färdiga att använda. Ett exempel, *HTML2Canvas,* använder sig av HTML5, Canvas och JavaScript. Grundidén är att JavaScript kan läsa DOM (Document Object Model) och utifrån det generera en snarlik version av sidan med Canvas. Med generera menas här att en faktiskt skärmdump inte tas utan informationen i DOM:en snarare används till att återbygga sidan i ett Canvas-objekt. Ett Canvas kan sedan i sin tur sparas i ett bildformat hos klienten med hjälp av ytterligare JavaScript-bibliotek. Denna lösning sker helt och hållet på klientsidan och är således ett bra alternativ i prototypen.

9.8. Stratigrafisk profilskiss – algoritm och buggar

Den stratigrafiska profilskissen består av en SVG-figur med två delar: en statisk del som ritas upp (och göms) vid applikationens uppstart samt en dynamisk del. Innehållet i den statiska delen syns i figur 9.1 till höger. Allt annat genereras dynamiskt när en användare trycker på en markör. Den statiska delen finns hårdkodad i index-filen och består av ett SVG-

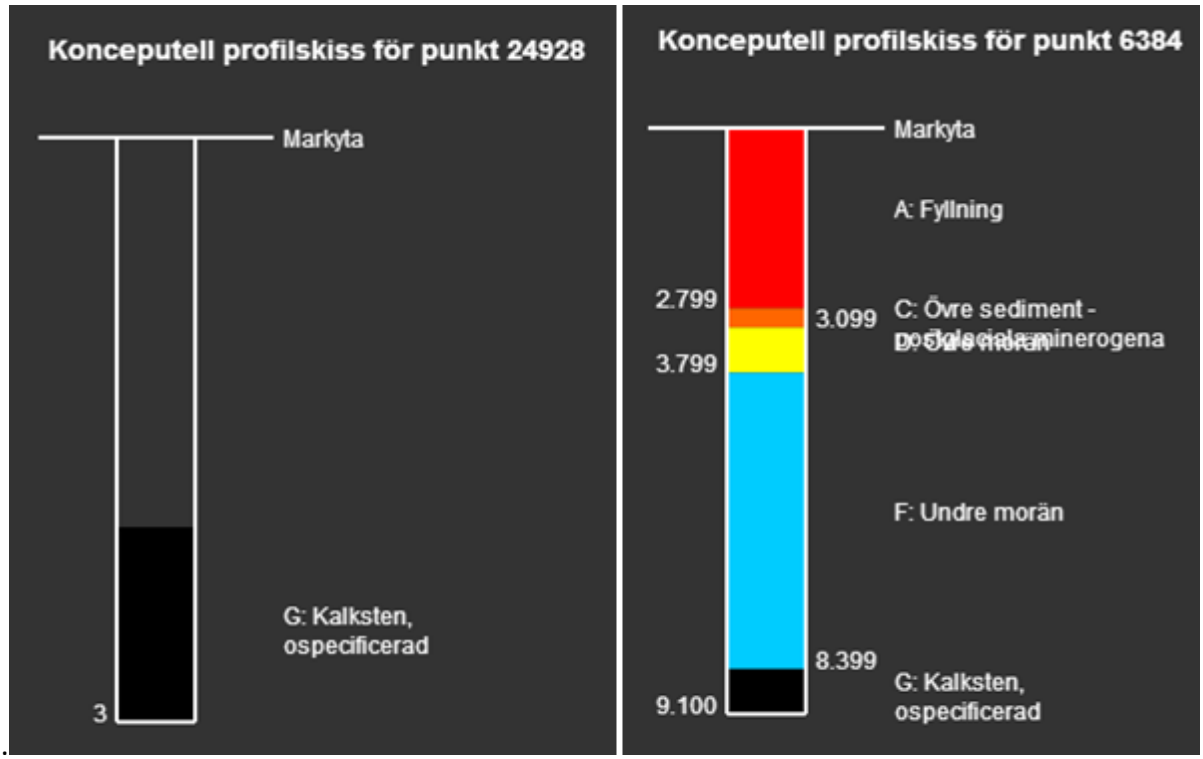


Figur 9.1: Den statiska delen i SVG-figuren.

rotelement med efterföljande *line*- och *text*-element. Dessa element påminner om övriga HTML-element i sin uppbyggnad. Faktum är att de ingår i dokumentets objektmodell (DOM) vilket möjliggör dynamisk ändring med hjälp av JavaScript. Algoritmen när en användare trycker på en markör är enligt följande:

- Töm allt i SVG-figuren förutom de statiska elementen.
- För varje stratigrafiskt lager
    - Skapa en rektangel med bestämd bredd och med en höjd som baseras på lagrets verkliga djup.
    - Färglägg rektangeln beroende på vad det stratigrafiska lagret heter.
    - Skriv ut lagertexten i halva lagrets höjd (se figur 8.9).
    - Skriv ut slutmeter för lagret.

SVG-utritningen fungerar tillfredsställande med reservation för ett par buggar. Det första problemet är en följd av inkonsekvent rådata. I några enstaka fall har stratigrafiska lagerföljder upptäckts som inte börjar på meter 0 (d.v.s. markytan). Varför sådana fall finns är oklart men eftersom GeoAtlas tillåter sådan representation bör även webbgrässnittet göra det. Resultatet blir som till vänster figur 9.2. Dessvärre skriver algoritmen i dagsläget ut slutmeter för varje lager, vilket i en standardpunkt fungerar bra, men i detta specialfall resulterar i att startmetern för lagret uteblir.



Figur 9.2: Buggar som kan uppstå i utritningen av stratigrafisk profildata. Till vänster: specialfall där lagerföljden inte börjar på noll meters djup. Till höger: Om flera smala lager ligger i följd så flyter texten ihop i vissa fall.

Ett annat specialfall är när två eller fler smala lager följer efter varandra. Problemet illustreras i figur 9.2 till höger och är en följd av att textplaceringen alltid sätts till lagrets mitt (se t.ex. ”F: Undre morän” i samma figur). Om ett av lagrens text är långt och skrivs ut på två rader kan det medföra överlappning med nästkommande lagertext. Detta skulle kunna lösas genom att textplaceringarna alltid sätts med lika avstånd till varandra och att linjer får peka från lagret ut till texten.

9.9. Mobilkompabilitet

Prototypens nuvarande utformning är inte speciellt mobilvänlig även om stor potential finns. Leaflet i sig är mobilanpassat och övriga teknikval som JavaScript, CSS3 och SVG har också stöd i de stora mobila webbläsarna (Can I Use 2013). Den stora problematiken är valet att utföra alla beräkningar på klientsidan. Detta omöjliggör mobil användning eftersom processeringskraften inte räcker till. Störst kraft går i dagsläget åt till klusterfunktionaliteten. I avsnitt 9.3. diskuteras alternativ till detta och vid införande av ett sådant kommer mobilkompabiliteten förbättras avsevärt.

9.10. Öppen källkod

Prototypen har med framgång byggts endast med hjälp av öppen källkod-alternativ. Denna aspekt är viktig för att projektet skall kunna återskapas utanför den egna fallstudien. Det enda fallet av proprietär mjukvara som förekommit är FME som användes i dataförädlingsfasen. Dataförädlingen anses dock inte vara en del av själva prototypen och är en sak som kan utföras på många olika sätt. Bland annat kan man åstadkomma liknande resultat i PostGIS, om än med något större tidsåtgång.

10. Slutsatser

Syftet med detta projekt var att undersöka hur en applikation för visualisering av data knutet till geotekniska provborrningar kan se ut och utifrån det skapa en prototyp. Med avseende på skärmupplösningar i dagens datorer och handhållna enheter har designval gjorts. Slutsatsen där är att det är svårt att utforma en applikation som passar alla plattformar, både tekniskt och estetiskt. Viktiga saker som har tagits i beaktning är mängden information på skärmen, dels i form av attribut och dels i form av företeelser. En klustermodul används i applikationen för att göra data så lättöverskådligt som möjligt. All informations placering på skärmen har motiverats och optimerats för att få en så stor kartyta som möjligt.

De teknikval som har gjorts möjliggör implementation av samtliga punkter i kravspecifikationen. För kartans uppbyggnad har Leaflet valts som API där stöd finns för många av de grundläggande krav som ställs på applikationen, som t.ex. zoom, panorering, skalstreck och bakgrundskarta. För lagring av företeelser har utbytesformatet GeoJSON valts som har nära koppling till JavaScript, vilket är det valda skriptspråket. Funktioner som grundat sig på dessa teknikval är bl.a. sökfunktion, dynamisk attributvisning och

referenshantering. Slutligen har SVG har valts som standard för dynamisk visning av stratigrafisk profildata.

Funktioner som inte implementerats är möjlighet att mäta längd i kartan, sökning på valfria attribut, redigering av rådata, exportverktyg, norrpil samt 3D-funktionalitet. Generellt beror detta på tidsbrist men i fallet med 3D-funktionalitet beror det även på avsaknad av data i form av geologisk 3D-modell.

Resultatet av detta arbete är en prototyp som påvisar att det går att flytta visning av geotekniska provborrningar från GeoAtlas till en webbmiljö. Därmed får rapportens syfte anses vara uppfyllt. Vidareutveckling kan ske så att de uteblivna funktionerna implementeras, men för att utföra analyser kommer fortfarande GeoAtlas vara en nödvändighet. Då målgruppen för webbapplikationen är icke-experter är det inte heller orimligt att ha den uppdelningen i funktionalitet mellan desktop-mjukvara och webbapplikation. Det sistnämnda fungerar då som en lättillgänglig datavisningsmiljö snarare än ett GIS i webbmiljö.

Referenser

Artiklar

Bereuter och Weibel 2013. Real-time generalization of point data in mobile and web mapping using quadtrees. *Cartography and Geographic Information Science* 40:4, s. 271-281. doi: 10.1080/15230406.2013.779779.

Burggraf 2006. Geography Markup Language. *Data Science Journal* 5, s.178-204. doi: http://dx.doi.org/10.2481/dsj.5.178.

Peng & Zhang 2004. The roles of GML, SVG and WFS specifications in the development of Internet GIS. *Journal of Geographical Systems* 6, s. 95–116. doi: 10.1007/s10109-004-0129-0.

Sen & Duffy 2004. GeoSciML: Development of a generic GeoScience Markup Language. *Computers & Geosciences* 31, s. 1095-1103. doi: 10.1016/j.cageo.2004.12.003.

Yan & Li 2013. An approach to simplifying point features on maps using the multiplicative weighted Voronoi diagram. *Journal of Spatial Science* 58:2, s. 291-304. doi: 10.1080/14498596.2013.815578.

Böcker

Brutzman & Daly 2007. *Extensible 3D Graphics for Web Authors*. San Francisco: Morgan Kaufmann Publishers.

Clark m.fl. 2012. *Beginning HTML5 and CSS3*. New York: Apress.

Harrie & Svensson 2013. Lagring av geografiska data. I *Geografisk informationsbehandling – teori, metoder och tillämpningar*, L. Harrie, sida 161-182. Studentlitteratur: Lund.

Hawkes 2011. *Foundation HTML5 Canvas*. New York: Friends of ED.

Hunt 2006. Geotechnical Investigation Methods: A Field Guide for Geotechnical Engineers. Boca Raton: Taylor & Francis Group.

Lee m.fl. 2012. *Foundation Website Creation with HTML5, CSS3, and JavaScrip.* New York: Friends of ED.

Matthew & Stones 2005. *Beginning Databases with PostgreSQL*. New York: Apress.

Obe & Hsu 2011. *PostGIS in Action*. Stamford: Manning Publications Co.

Owens 2006. *The Definitive Guide to SQLite*. New York: Apress.

Rob & Coronel 2009. *Database Systems: Design, Implementation and Management*, 8. Uppl. Boston: Course Technology.

Salminen & Tompa 2011. *Communicating with XML*. New York: Springer Sience.

Uppslagsverk

Michaels & Ames 2008. Web Feature Service and Web Map Service. I *Encyclopedia of GIS*. Shekhar & H. Xiong, sida 1259-1261. New York: Springer Sience.

Semerjian 2008. Metadata. I *Encyclopedia of GIS*. Shekhar & H. Xiong, sida 1259-1261. New York: Springer Sience.

Sinha 2008. Web Feature Service (WFS). I *Encyclopedia of GIS*. Shekhar & H. Xiong, sida 1256-1259. New York: Springer Sience.

Urban & Dietrich 2009. Object Data Models. I *Encyclopedia of Database Systems*, L. Liu & M Tamer Özsu, sida 1929-1935. New York: Springer Sience.

Webbsidor

ArcGIS 2013. ArcGIS API for JavaScript: Canvas with raster layer example. https://developers.arcgis.com/en/javascript/jssamples/exp_rasterlayer.html (Hämtad 2013-09-13).

BGS 2013. British Geological Survey: Geology of Britain viewer. http://mapapps.bgs.ac.uk/geologyofbritain/home.html (Hämtad 2013-09-20).

Can I Use 2013. Can I Use – Support tables for HTML5, CSS3 etc. http://caniuse.com (Hämtad 2013-11-25).

CC 2013. Creative Commons - Attribution-ShareAlike 2.0 Generic (CC BY-SA 2.0). http://creativecommons.org/licenses/by-sa/2.0/ (Hämtad 2013-12-12).

CGI 2013. Commission for the Management and Application of Geosience Information: GeoSciML. http://www.cgi-iugs.org/tech_collaboration/geosciml.html (Hämtad 2013-09-27).

Chrome Experiments 2013. Chrome Experiments: The WebGL Globe. http://www.chromeexperiments.com/globe (Hämtad den 2013-09-10).

ESRI 1998. ESRI Shapefile Technical Description. http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf (Hämtad 2013-10-04).

Flightradar24 2013. Flightradar 24 – Live Air Traffic. http://www.flightradar24.com (Hämtad 2013-11-21).

GeoJSON 2008. The GeoJSON format specification. http://geojson.org/geojson-spec.html (Hämtad 2013-09-17).

GeoServer 2009. What is GeoServer. http://geoserver.org/display/GEOS/What+is+Geoserver (Hämtad 2013-10-07).

GeoServer 2013. GeoServer: WMS reference. http://docs.geoserver.org/stable/en/user/services/wms/reference.html (Hämtad 2013-09-25).

Google Maps 2013. Google Maps. http://maps.google.se (Hämtad 2013-11-12).

IETF 2006. The Internet Engineering Task Force Specifications: RFC4627 (JSON). http://tools.ietf.org/html/rfc4627 (Hämtad 2013-09-17).

ISO 2004. International Organization for Standardization: 19125-1:2004 Geographic information - Simple feature access. http://www.iso.org/iso/catalogue_detail.htm?csnumber=40114 (Hämtad 2013-09-17).

Leaflet 2013a. Leaflet Markercluster plugin. http://github.com/Leaflet/Leaflet.markercluster (Hämtad 2013-11-04).

LeafLet 2013b. LeafLet plugins: Elevation. http://leafletjs.com/plugins.html (Hämtad 2013-09-12).

Leaflet 2013c. Leaflet License. https://github.com/Leaflet/Leaflet/blob/master/LICENSE (Hämtad 2013-12-12).

MapBox 2013a. Mapbox JavaScript API v.1.3.1: Custom Marker Tooltips. http://www.mapbox.com/mapbox.js/example/v1.0.0/custom-marker-tooltip/ (Hämtad 2013-09-12).

MapBox 2013b. Mapbox JavaScript API v.1.3.1: Linking to external data. http://www.mapbox.com/mapbox.js/example/v1.0.0/linking-to-external-data/ (Hämtad 2013-09-12).

MapServer 2013. MapServer 6.4.0 Documentation. http://mapserver.org/documentation.html (Hämtad 2013-10-07).

MarineTraffic 2013. Live Ships Map. http://www.marinetraffic.com (Hämtad 2013-11-21).

OSM 2013a. Open Street Map – The Free Wiki World Map. http://www.openstreetmap.org (Hämtad 2013-11-12).

OSM 2013b. History of OpenStreetMap. http://wiki.openstreetmap.org/wiki/History_of_OpenStreetMap (Hämtad 2013-11-12).

Postgis 2012. PostGIS Usage: Institut Géographique National. http://postgis.net/2012/10/18/ign (Hämtad 2013-10-10).

PostgreSQL 2013. About PostgreSQL. http://www.postgresql.org/about/ (Hämtad 2013-10-01).

Respond 2013. Respond.js. https://github.com/scottjehl/Respond (Hämtad 2013-11-26).

SFC 2013. San Francisco Crimespotting. http://sanfrancisco.crimespotting.org (Hämtad 2013-11-21).

SGU 2013. Sveriges Geologiska Undersöknings Kartvisare, http://www.sgu.se/kartvisare/kartvisare-borrkarnor-sv.html (Hämtad 2013-10-02).

SpatiaLite 2013. Manual for SpatiaLite – Spatial extensions for SQLite. http://www.gaia-gis.it/gaia-sins/spatialite-manual-2.3.1.html (Hämtad 2013-09-30).

SQLite 2013a. About SQLite. http://www.sqlite.org/about.html (Hämtad 2013-09-30).

SQLite 2013b. SQLite R*Tree Module. http://www.sqlite.org/draft/rtree.html (Hämtad 2013-09-30).

Stamen 2013. Stamen Maps. http://maps.stamen.com (Hämtad 2013-11-12).

uMap 2013. uMap – Play with the demo. http://umap.fluv.io/en/ (Hämtad 2013-09-12).

W3C 2004. World Wide Web Consortium Specifications: SVG. http://www.w3.org/Graphics/SVG/About.html (Hämtad 2013-09-24).

W3DS 2010. The Scope of W3DS. http://www.w3ds.org/doku.php?id=scope (Hämtad 2013-10-08).

W3S 2013a. W3 Schools: HTML5 Canvas. http://www.w3schools.com/html/html5_canvas.asp (Hämtad 2013-09-24).

W3S 2013b. W3 Schools: SVG Tutorial. http://www.w3schools.com/svg/ (Hämtad 2013-09-30).

W3S 2013c. W3 Schools: Introduction to XML. http://www.w3schools.com/xml/xml_whatis.asp (Hämtad 2013-09-26).

W3S 2013d. W3 Schools: An XML example. http://www.w3schools.com/xml/cd_catalog.xml (Hämtad 2013-09-26).

Web Standards Project 2003. The Web Standards Project Welcomes New Members, Defines Initiatives for 2003. The Web Standards Project. http://www.webstandards.org/press/releases/2003-02-14/ (Hämtad 2013-09-20).

Wikipedia 2008. Quadtree. http://de.wikipedia.org/wiki/Quadtree (Hämtad 2013-12-10).

Wikipedia 2010. Open Source Geospatial Foundation.
http://en.wikipedia.org/wiki/Open_Source_Geospatial_Foundation (Hämtad 2013-10-07).

X3DOM 2013, About X3Dom. http://www.x3dom.org/?page_id=2 (Hämtad 2013-10-08).

Institutionen för naturgeografi och ekosystemvetenskap, Lunds Universitet.

Detta arbete ingår i en serie examensarbeten (Seminarieuppsatser) i geografisk informationsteknik. Uppsatserna finns tillgängliga på institutionens geobibliotek, Sölvegatan 12, 223 62 LUND. Serien startade 2010. Hela listan och själva uppsatserna är även tillgängliga på LUP student papers och via Geobiblioteket (www.geobib.lu.se)

1 *Patrik Carlsson och Ulrik Nilsson* (2010) Tredimensionella GIS vid fastighetsförvalting.

2 *Karin Ekman och Anna Felleson* (2010) Att välja grundläggande karttjänst – Utveckling av jämförelsemodell och testverktyg för utvärdering

3 *Jakob Mattsson* (2011) Synkronisering av vägdatabaser med KML och GeoRSS - En fallstudie i Trafikverkets verksamhet

4 *Patrik Andersson and Anders Jürisoo* (2011) Effective use of open source GIS in rural planning in South Africa

5 *Nariman Emamian och Martin Fredriksson* (2012)Visualiserig av bygglovsärenden med hjälp av Open Source-verktyg - En undersökning kring hur man kan effektivisera ärendehantering med hjälp av en webbapplikation

6 *Gustav Ekstedt and Torkel Endoff* (2012) Design and Development of a Mobile GIS Application for Municipal Field Work

7 *Karl Söderberg* (2012) Smartphones and 3D Augmented Reality for disaster management - A study of smartphones ability to visualise 3D objects in augmented reality to aid emergency workers in disaster management

8 *Viktoria Strömberg* (2012) Volymberäkning i samhällsbyggnadsprojekt

9 *Daniel Persson* (2013) Lagring och webbaserad visualisering av 3D-stadsmodeller - En pilotstudie i Kristianstad kommun

10 *Danebjer Lisette och Nyberg Magdalena* (2013) Utbyte av geodata - studie av leveransstrukturer enligt Sveriges kommuner och landstings objekttypskatalog

11 *Alexander Quist* (2013) Undersökning och utveckling av ett mobilt GIS-system för kommunal verksamhet

12 *Nariman Emamian* (2014) Visning av geotekniska provborrningar i en webbmiljö