Datavetenskap

Andreas Bäck, Mikael Mogren

Location Based Positioning Service

Examensarbete C-nivå

2005:08

Location Based Positioning Service

Andreas Bäck, Mikael Mogren

Denna rapport är skriven som en del av det arbete som krävs för att erhålla en kandidatexamen i datavetenskap. Allt material i denna rapport, vilket inte är vårt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Andreas Bäck

Mikael Mogren

Godkänd, 20050602

Handledare: Christer Andersson

Examinator: Donald Ross

Sammanfattning

Detta examensarbete beskriver utvecklingen av en webbaserad kart/positioneringstjänst för ett brett spektrum av klienter, vilka använder sig av en GPS-mottagare för bestämning av den egna positionen. Applikationen möjliggör att en användare kan, med ett minimum av egen hård- och mjukvara, visualisera sin egen position på en karta över sitt närområde.

I uppgiften läggs stor vikt vid två faktorer; att hålla en hög grad av kompatibilitet för att kunna stödja ett stort antal klienttyper, och att klientens krav på utrustning skall minimeras.

Uppsatsen utgår ifrån en inledande studie av relevanta teknologier, för att sedan utifrån dessa beskriva konstruktionen av denna applikation.

Abstract

This thesis describes the development of a web based positioning service, suitable for a wide range of clients, each utilizing a GPS receiver to determine its current position. The application enables a user, with minimal requirements regarding hardware as well as software, to visualize his/her position on a map of his/her surroundings.

The solution emphasizes two aspects; to achieve a high degree of platform independence to support a wide range of clients, and to minimize the requirements regarding equipment for the client.

This thesis begins with a introductory study of involved technologies, and then later describes the design and implementation of the application.

Innehåll

Figurer

Kapitel 1

Introduktion

1.1 Syfte och målsättning

Syftet med arbetet är att producera en grundläggande positions-baserad karttjänst åt Karlstads universitet. Tjänsten skall sedan stå till grund för forskning angående personlig integritet i positions-baserade system. Implementationen skall genomföras med Java-teknologier.

1.2 Genomförande

Arbetet inleddes med att i samband med uppdragsgivaren närmare bestämma detaljer för implementationen, såsom protokoll för kommunikation mellan klient och server. Efter att detta bestämts undersöktes applikationens design närmare, tillsammans med relevanta teknologier.

1.3 Begränsningar

Tid

Examensarbetet har utförts under en tidsperiod motsvarande 10 högskolepoäng under vårterminen 2005.

Mjukvara

Emacs och Eclipse SDK version 3 har använts vid framtagandet av denna applikation. Uppsatsen är skriven i LaTeX.

Hårdvara

En av uppdragsgivaren erhållen GPS-mottagare, Holux GR-212 för utveckling och testning av applikationen.

1.4 Uppsatsens upplägg

- **Kapitel 2** beskriver bakgrund till förutsättningar för detta arbete.
- **Kapitel 3** ger den nödvändiga tekniska bakgrunden.
- **Kapitel 4** beskriver alla delar i denna applikation.
- **Kapitel 5** beskriver vilka utökningar och fortsatta arbeten som är aktuella.
- **Kapitel 6** presenterar resultatet och ger en avslutande diskussion.

Kapitel 2

Bakgrund

Syftet med denna uppgift har varit att ta fram en applikation för klienter som med hjälp av en mottagare avsedd för det globala positioneringssystemet (på engelska GPS, där GPS står för Global Positioning System), och någon form av Internetanslutning erhålla möjligheter för olika positionsrelaterade tjänster.

I dagsläget kan klienter erbjudas en stor mängd möjligheter; beroende på vilka tjänster som önskas erbjuds bland annat möjligheter som positionsangivelser på kartbilder, hjälp med navigering samt lokalisering av enheter. Lokaliseringen utav enheterna kan ske på en rad olika sätt, och vidare kan användare med operatörens hjälp lokalisera varandra. Beroende på vilken typ av tjänst som önskas kan användare, med eller utan en operatörs hjälp få information om sin egna position.

De olika typer av positionsbaserade tjänster som idag erbjuds kan erhållas på olika sätt:

- En enhet med ansluten eller inbyggd GPS-mottagare (t.ex. handdator eller laptop) skickar sina koordinater till en server som utför de beräkningar tjänsten kräver. Servern skickar därefter tillbaka resultatet. Vidare kan koordinaterna behandlas lokalt hos klienten, för att sedan t.ex. visas genom en kartbild på användarens bildskärm. Detta är ett område som är på stark frammarsch. De lösningar som existerar här idag innebär ofta att någon form av programvara och/eller kartpaket installeras hos

klienten. Detta ställer ytterligare krav på klienten genom uppdateringar med nya kartor och eventuell installation av programvara. Denna typ av lösning kan resultera i en relativt hög kostnad med avseende på användarvänlighet och systemresurser.

- En enhet (t.ex. mobiltelefon) ansluten till någon form av trådlöst nätverk (t.ex. GSM) kan erhålla sin position genom att den *trianguleras*. Triangulering är en känd teknik som med viss framgång kan användas för att ta reda på var en mobil terminal befinner sig. Grundtanken är att utnyttja information om avståndet från minst tre basstationer och på det sättet räkna ut mobilens troliga position. Principen för triangulering diskuteras mer utförligt i avsnitt 3.2. klienten har med denna typ av positionering liten eller ingen kontroll på över hur dess position avslöjas då leverantören av tjänsten står för mätningen av dess position.

2.1 Förutsättningar

Lösningen på den positioneringstjänst denna uppsats är baserad på kommer att beskrivas mer i detalj i kapitel 4. Övergripande har denna lösning några skillnader gentemot de lösningar som existerar idag (se avsnitt 2.2 för andra lösningar). En enhet med ansluten GPS-mottagare (t.ex. handdator, mobiltelefon eller laptop) skall utan någon speciell förinstallerad programvara kunna ansluta till en webbsida och erhålla sin position på markerad på en tvådimensionell kartbild. Det krävs dock att enheten har någon form av webbläsare (vilket även mobiltelefoner har idag). Lösningen är anpassad för att fungera på ett så brett spektrum av klienttyper som möjligt. Detta innebär att nästan all logik ligger hos servern. En klient har den enda uppgiften att läsa in koordinater från GPS-mottagaren och vidarebefordra dessa till en webbserver. Webbservern returnerar sedan en bild med klientens position markerad. Inläsningen av koordinater från GPS-mottagaren sker normalt per automatik genom en *signerad java-applet* (se avsnitt 4.3.2). Med vissa mindre klienter, där inläsning av koordinater med hjälp av en applet inte möjligt, erbjuds möjligheten att

manuellt skriva in koordinater.

När det gäller uppbyggnaden av kartdatabasen sköts det utav användare med administratörsrättigheter. Dessa kan när som helst ladda upp kartpaket till databasen. Vilka kartformat som används och vilka ytor som täcks är initialt ospecificerade.

2.2 Andra lösningar

I dagsläget erbjuds en mängd positionsrelaterade tjänster. Den gemensamma länken som sammanknyter dessa tjänster är att de alla levererar någon form av positionering. I vilken form respektive tjänst kommer varierar en hel del. Tillvägagångssättet för att tillhandahålla dessa tjänster kan variera beroende på tjänst. Hur tjänsten är uppbygd beskrivs i mer i avsnitt 3.4. Detta avsnitt kommer dock endast ta upp några utav de positionsrelaterade tjänster som finns och vad som erbjuds till privatpersoner idag.

2.2.1 Webbaserade tjänster

Det finns ett antal rent webbaserade tjänster att tillgå idag. Positionering med hjälp från dessa tjänster brukar erhållas genom att koordinater eller lägen (t.ex. adresser) specifieras av användaren. Frågan vidarebefodras sedan till en server vilken returnerar resultatet till användaren i någon form, vanligtvis som text eller som en HTML-sida med en bild. En server-klient-lösning är ett vanligt förekommande sätt att lösa uppgifter av denna typ. Vanligtvis ligger någon form av kart- eller positionsdata lokalt på en server. Klienter får sedan ansluta till servern (behöver inte vara själva kart-servern) för att erhålla positionsdatan.

Eniro Sverige AB [1] erbjuder en kartjänst i vilken användare bland annat erbjuds söka positioner genom att manuellt ange adresser. Begärd position visas sedan upp i webbläsaren i form av en kartbild över det aktuella området. Användaren har sedan möjligheten att zooma och panorera i själva kartbilden. Eniro låter även användare plotta ut en vägbeskrivning mellan olika positioner genom att ange en start- och slut-adress. Denna

tjänst återfinns på http://kartor.eniro.se/. En liknande tjänst går finna hos Google Maps på http://maps.google.com/. Båda dessa är för närvarande gratis för allmänheten.

Det som krävs för att få denna tjänst fungera är en enhet med kompatibel webbläsare och Internetuppkoppling. Tjänsten kräver dock viss förkunskap angående vilken position som önskas från användaren eftersom positionering inte sker automatiskt.

2.2.2 Positionering via mobilnätet

Positionering av mobila enheter (vanligtvis mobiltelefoner) med hjälp av mobilnätet är den i dagsläget vanligast förekommande metoden för att lokalisera användare. Avsnitt 3.4 beskriver hur tekniken är uppbygd och fungerar mer i detalj medan 3.2 tar beskriver de olika teknologier som används vid GSM[1]-positionering. Detta avsnitt tar dock endast upp några av de tjänster och lösningar som existerar på marknaden idag.

Friendfinder är en tjänst som erbjuds av TeliaSonera AB [2]. Den erbjuder privatpersoner att för en mindre kostnad erhålla positionsdata angående deras egna eller andras enheter (exempelvis vänner och bekanta). För att tjänsten skall kunna utnytjas måste användaren ha TeliaSonera som operatör samt acceptera de villkor vilka gäller för tjänsten. Användaren kan via ett webbgränssnitt eller via sms erhålla data angående sin egen eller någon annans position. För att någon annan än klienten själv skall kunna erhålla data angående enhetens position, krävs att denne har accepterat att den skall kunna bli uppsökt. Tjänsten går att hitta på http://friendfinder.telia.se/.

Tjänsten är inte bunden till någon speciell typ av enhet. Tyvärr erbjuder tjänsten ingen större precision. Avvikelser från den faktiska positionen kan variera flera kilometer på de ytor som har dålig täckning av operatörens basstationer. I de områden där det är tätare mellan basstationerna uppnås en precision på cirka 200 meter.

[1]Global System for Mobile Communications. Den mest använda standarden för kommunikation mellan mobiltelefoner.

2.2.3 Globala Positionerings Systemet

Positionering av enheter med det Globala Positionerings Systemet är en metod som blivit alltmer populär bland allmänheten. GPS diskuteras mer i detalj i avsnitt 3.1. Själva positioneringstjänsten är gratis att använda sig av. Den tillhandahåller dock endast positionsdata för en given position. Önskas någon extra funktionalitet, till exempel en position grafiskt utritad på en kartbild, kräver detta någon extra form av programvara (d.v.s. en applikation), samt kartbilder.

TomTom BV [3] är ett företag som erbjuder två liknande positioneringstjänster. Dessa tjänster har de största likheterna vi hittat jämfört med applikationen som beskrivs i denna uppsats. Tjänsterna är på samma sätt anpassade för små enheter med ansluten GPS-mottagare. Den största skillnaden mellan vår lösning och TomToms lösning är att TomToms kräver installation av extra programvara samt kartpaket hos enheten. Detta krävs inte i den lösning vilken har legat som grund till denna uppsats.

TomTom MOBILE 5 är framtagen för att fungera med vissa "smarta" mobiltelefoner. Den använder en av TomTom framtagen separat GPS-mottagare för att erhålla geografisk data. Några av de krav tjänsten ställer är att telefonen är utrustad med trådlös överföringsteknik (bluetooth) samt att den har stöd för minneskort. Kommunikationen mellan telefon och GPS sker via bluetooth. Minneskort krävs för att installera den programvaran som behövs, samt att ge plats åt kartpaket för de ytor som skall täckas. Kartorna kan laddas ner till telefonen via GPRS-nätet (General Packet Radio Service), eller från en dator. Dock är vissa kartpaket ganska stora. Rekommendationen är därför att föra över dessa till telefonen via en dator då överföringen via GPRS tar lång tid, eller blir för kostsam.

TomTom NAVIGATOR har stora likheter gentemot MOBILE 5. Det som skiljer dem åt är att TomTom NAVIGATOR är anpassad för handdatorer. Anslutningen till GPS kan ske via bluetooth eller kabel. Figur 2.1 ger en enkel illustration över TomToms lösning.

Om användaren har de kartpaket denne önskar tillkommer ingen GPRS-anslutning

vid positionering. Önskas realtidsinformation krävs dock GPRS-anslutning. Detta har sina fördelar då GPRS-anslutningar till andra nät kan vara ekonomiskt kostsamma.

Mer information om dessa tjänster finns på http://www.tomtom.com/.

Figur 2.1: TomTom lösning med GPS.

Kapitel 3

Litteraturstudie

Här diskuteras översiktligen olika möjligheter att mäta en klients position vid användning av positionsbaserade tjänster. Då detta arbete använder sig av GPS-systemet för positionering, är detta system det som beskrivs mest ingående.

3.1 GPS

GPS är ett avgiftsfritt system för satellitnavigering, vilket tillhandahåller precis positionsbestämning samt exakt tidsbestämning över hela jordens yta.

Systemet utvecklades av det amerikanska försvarsdepartementet för användning inom det Amerikanska försvaret. Systemet driftsattes 1978, då den första satelliten sköts upp.

Tidigare infördes en precisionsförsämrande störning i systemet, vilken kunde kringgås av USA:s militär. Denna störning togs dock bort år 2000, enligt beslut av president Clinton. USA har dock fortfarande möjlighet att applicera en precisionsförsämrande störning om situationen anses kräva det. Detta är en anledning till att alternativa system tagits fram av EU respektive Ryssland. Inget av dessa system är dock ännu ett praktiskt alternativ, då det ryska systemet Global Orbiting Navigation Satellite System (GLONASS) [4] i skrivande stund endast utgörs av ett fåtal satelliter och EU:s Galileo-system [5] fortfarande befinner

sig på ett tidigt stadium.

3.1.1 Teknologi

GPS-systemet kan anses bestå av tre delar, rymd-, kontroll- och användardel. Rymddelen består av en satellitkonstellation[1] bestående av minst 24 satelliter. Kontrolldelen består av ett flertal markstationer runtom i världen, vilka ansvarar för övervakning av satelliterna, synkronisering av satelliternas atomur, och uppladdning av data till satelliterna. Användardelen består av en mängd GPS-mottagare, både för militärt och civilt bruk.

Satelliterna i GPS-systemet färdas i medelhög omloppsbana (20200 km över havet) med en omloppstid på 12 timmar. Satelliterna är positionerade i sex plan med 60 graders mellanrun, med fyra satelliter i varje plan[2]. Genom detta system är alltid mellan fem och åtta satelliter synliga från en godtycklig punkt på Jorden.

Positionsbestämning utförs med hjälp av *trilateration* (se 3.1.2), vilket redogörs för senare i detta avsnitt. Avståndet till satelliterna från mottagaren avgörs genom att mäta den tid det tar för signalen att färdas från sändare till mottagare.

Satelliternas tidsmätning säkerställs genom att varje satellit har ett atomur, vilket mäter den exakta tid i sekunder som förflutit sedan midnatt den 6:e Januari 1980. Detta tidssystem tar inte hänsyn till att en extra sekund ibland läggs till[3] för att korrigera för dagarnas förlängning. Detta leder till att GPS-tid ligger något före Greenwich-tid, vilket korrigeras av GPS-mottagarna. Även om klockornas atomur är mycket exakta uppstår en viss klockdrift[4], vilken mäts av markstationerna vilka sänder korrektionsdata till satelliterna vid ett flertal tillfällen per dygn. Dessutom påverkas satelliternas klockor av relativitetsrelaterade effekter, vilket medför att de går långsammare än om de hade befunnit sig på jorden. Denna felaktighet på ca 38 nanosekunder per dag korrigeras av elektronik på

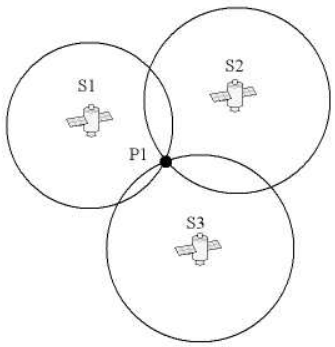
[1] Ett flertal satelliter som samarbetar på ett koordinerat sätt.
[2] Ett geometriskt plan vilket skär en planets mittpunkt. I detta plan ligger satelliterna i omloppsbana.
[3] Vanligen var 18:e månad.
[4] Ca 1 nanosekund per dygn.

satelliterna.

3.1.2 Trilateration

Detta är den teknik som idag används i GPS-systemet för att bestämma en GPS-mottagares position. Principen för detta system illustreras i figur 3.1, och bygger på bestämning av skärningspunkter mellan cirklar/sfärer. Genom att bestämma den tid en signal färdas från en satellit kan avståndet till satelliten bestämmas, och då det kan visas att tre givna cirklar endast skär varandra i en punkt kan avstånden till tre satelliter användas för att bestämma en mottagares position exakt i två dimensioner (longitud och latitud).

Att finna en positon i tre dimensioner (longitud, latitud och höjd över havet) kräver att en skärningspunkt mellan sfärer hittas. Det kan visas att det minsta antal sfärer som krävs för detta är fyra, och sålunda krävs kontakt med fyra satelliter för en korrekt positionsbestämning i tre dimensioner.



Figur 3.1: Trilateration.

3.1.3 Koordinatsystem

För att kunna utföra positionsbestämning krävs någon form av referenssystem. Att uppskatta Jordens form som en jämn sfär är inte lämplig här, då planeten inte är var sig jämn,

eller helt sfärisk till sin form. En lämplig metod är att uppskatta planetens form som en ellipsoid, utifrån tillgänglig geodata.

Fram till 1950-talet fanns ingen global modell för planetens form, vilket krävdes av flera anledningar, däribland kraven ställda av en ökad rymdforskning och behov av exakta globala kartor.

Av dessa anledningar togs World Geodesic System (WGS) [6] fram av USA:s försvarsdepartement. Förbättrad instrumentation har möjliggjort stadiga förbättringar av systemets precision. Den nu gällande modellen WGS84, vilken används av GPS-systemet, anses vara giltig till år 2010.

3.1.4 Felkällor

Som tidigare har nämnts är korrekt tidsbestämning av största vikt i GPS-systemet. Ett tidsfel på exempelvis en nanosekund medför ett positioneringsfel på 30 cm.

Positioneringsfel rörande tidsbestämning härstammar från ett antal felkällor. Olika effekter som påverkar tidsmätningen beskrivs nedan.

- Det går inte att garantera att mottagaren har en helt korrekt klocka. Dock kan mottagarens tidsfel beräknas och därmed elimineras, förutsatt att minst fyra satelliter finns tillgängliga. Dessutom kan eventuella extra satelliter användas för att minska positioneringsfelet.

- På grund av att ljusets hastighet genom atmosfäriska förhållanden varierar i jonosfären[5] kan fel på upp till 10 meter uppstå. Dessa fel korrigeras genom att en signal sänds på ett annat frekvensband. Genom att mäta fasdifferensen mellan signalerna kan felet beräknas och korrigeras.

- GPS-mottagaren kan, förutom de korrekta positioneringssignalerna, även ta emot

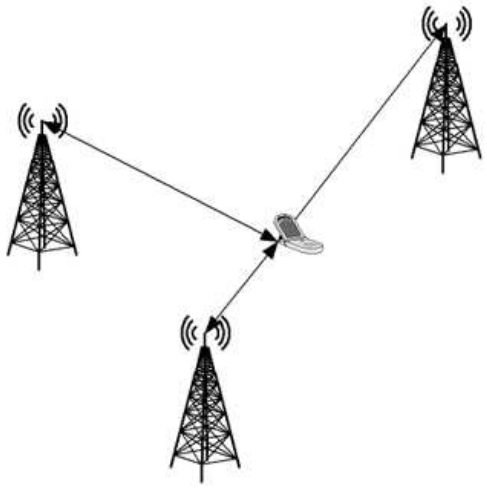
[5]Den delen av atmosfären som är på 50-500 km höjd över havet, där atmosfären joniseras av solens strålning.

störande signaler som studsar mot marken eller mot närliggande byggnader. Ett flertal tekniker för att filtrera bort reflektioner från avlägsna källor finns. Reflektioner från närliggande föremål kan dock fortfarande orsaka precisionsförsämringar.

3.2 Mobiloperatörsbaserade system

GSM-baserade positioneringssystem är i dagsläget mycket vanliga, då de ofta kräver minimala förändringar på operatörs- och kundsidan. Dessa system är dock underlägsna GPS-baserade system i fråga om precision i positioneringen, men har för närvarande ett ekonomiskt övertag.

Denna rapport kommer ej att gå in i detalj på dessa systems funktionalitet. Däremot är det lämpligt att här ge en översikt över de vanligaste teknologierna för GSM-baserad positionering. Principen för denna positioneringsmetod illustreras i figur 3.2. Grundtanken är att utnyttja information om avståndet från minst tre basstationer och på det sättet räkna ut mobilens troliga position



Figur 3.2: Triangulering av en mobiltelefon.

Det finns ett antal olika metoder för bestämning av en enhets posistion. En kortfattad

beskrivning av dessa ges nedan.

- *Cell Of Origin (COO):* Detta system lokaliserar användare utifrån aktuell basstation. Noggranheten beror på hur tätt stationerna är placerade, vilket medför att realtivt god precision (omkring 150 meter) kan uppnås i tätbebyggda områden.

- *Time Of Arrival (TOA):* Positionsbestämning utförs genom mätningar av den tid det tar för signaler att färdas från användarens mobiltelefon till flera basstationer. En viss precisionsförbättring erhålls jämfört med COO, vilken dock ej kan anses stå i proportion till den relativt höga kostnad som denna lösning medför, då en betydande mängd extrautrustning måste läggas till i basstationerna.

- *Angle Of Arrival (AOA):* Detta system påminner om TOA, dock utförs positionsbestämningen med vinkelmätning i stället för tidsmätning. Denna teknologi lämpar sig dåligt i tätbebyggda områden då den lätt kan störas av byggnader.

- *Enhanced Observed Time Difference (E-OTD):* Positionering utförs på liknande sätt som TOA, dock utförs tid- och positionsberäkningarna hos klienten, vilket medför en något dyrare mobiltelefon. Med denna teknologi kan god precision uppnås, ned till ca 50 meter, vilket gör den användbar för positionering av personer som har råkat ut för olyckor.

3.3 NMEA 0183

National Marine Electronics Association 0183 (vanligen benämnt NMEA) är en standard ursprungligen avsedd för kommunikation med instrument för marin navigering, vilken även har blivit en de facto-standard för kommunikation med GPS-utrustning.

3.3.1 Kommunikation

NMEA 0183 baseras på kommunikation mellan en sändande enhet som kommunicerar med en eller flera lyssnande enheter via seriell överföring. De specifika inställningarna för kommunikationen är beskrivna i tabell 3.1.

Överföringshastighet	4800 bps
Databitar	8
Stoppbitar	1 eller 2
Paritet	ingen
Handskakning	ingen

Tabell 3.1: Kommunikationsinställningar för NMEA 0183.

NMEA-meddelanden utgörs av textsträngar, vilka är max 82 tecken långa, inklusive vagnretur och radmatning, vilka terminerar meddelandet. De tecken som kan förekomma i ett NMEA-meddelande är alla ASCII-tecken som kan visas, samt radmatning och vagnretur.

Ett NMEA-meddelande inleds med ett '$'-tecken, vilket efterföljs av ett flertal kommaseparerade datafält. Om ett eller flera datafält skulle utelämnas, resulterar detta i två eller flera på varandra följande komma-tecken.

Ett exempel på ett fullständigt NMEA-meddelande kan ses nedan.

$$\$GPGLL^{1}, xxxx.xx^{2}, a^{3}, yyyy.yy^{4}, a^{5}, hhmmss.ss^{6}, a^{7}, *cc^{8}CRLF$$

1. *Addressfält:* De första två tecknen identifierar den sändande enheten, "GP" betyder här att enheten är en GPS-mottagare. De efterföljande tre tecknen identifierar meddelandetypen.
2. *Latitud:* Mottagarens latitud, där de första två siffrorna är gradantalet, och de efterföljande siffrorna är minutdelen av koordinaten.
3. *N eller S:* Anger om ovanstående latitud avser en position norr eller söder om ekvatorn.

4. *Longitud:* Detta fält följer samma format som latitudfältet.
5. *E eller W:* Anger om longituden är östlig eller västlig.
6. *Tidsangivelse:* Anger vid vilken tidpunkt positioneringen utfördes.
7. *Status:* A om positionen i meddelandet är giltig, V om den inte är pålitlig, exempelvis som en följd av för lågt antal synliga satelliter.
8. *Kontrollsumma:* En kontrollsumma, vilken inleds med '*', och utgörs av XOR av alla tecken i meddelandet mellan '$' och '*'. Detta fält är ej obligatoriskt.

3.4 Positionsbaserade tjänster

3.4.1 Översikt

Området inom positionsbaserade tjänster (på engelska LBS, där LBS står för Location Based Services) är i skrivande stund under stor utveckling. Med positions baserade tjänster menas att information rörande en användares (rättare sagt, en användares enhets) position på något sätt blir meddelad till leverantören av tjänsten som sedan utför de uppgifter tjänsten kräver. Positionen kan bestämmas genom att enheten själv använder speciell hårdvara, som till exempel en GPS-mottagare (GPS-systemet diskuterades mer ingående i avsnitt 3.1) för att ange sin position, eller av mobiloperatören inom operatörens nät (se avsnitt 3.2 för ytterligare information angående mobiloperatörsbaserade system). Vissa enheter (t.ex handdatorer) levereras med integrerad GPS-mottagare medan andra erbjuder möjligheten för någon typ av anslutning. GPS-motagarna har de senare åren krympt i storlek och sjunkit i pris, för att idag ha blivit ett vanligt förekommande hjälpmedel bland privatpersoner och företag för att erhålla data angående sin position.

De positionsbaserade tjänsterna kan delas in i två kategorier:

- Användaren "drar" informationen från LBS-leverantören. Detta sker på användarens

begäran genom att denne sänder data angående sin position. Denna tjänst benämns på engelska som en *pull*-tjänst. Detta är ett vanligt förfarande vid positionerings- och navigationstjänster.

- Informationen "trycks" på användaren. Detta innebär att användaren blir kontinuerligt positionerad av LBS-leverantören. Denna tjänst benämns på engelska som en *push*-tjänst. Detta är ett vanligt förfarande vid mobila dating och upplysningstjänster

I båda av dessa fall är det leverantören som ansvarar för inmätning av en användarens position.

Med hänsyn tagen till den personliga integriteten kan tekniken där mobiloperatören ansvarar för mätningen utav enheternas position i vissa fall anses kränkande. Då detta sköts av operatören har användaren liten eller ingen kontroll över hur och när data angående enhetens position skickas. Det enda säkra sättet att kontrollera huruvida data angående en enhets position erhålls är att avbryta uppkopplingen till mobiloperatörens nät (t.ex genom att stänga av sin mobiltelefon).

I de fall där användaren själv ansvarar för mätningen av sin egna position får denne ett mer kontrollerat förfarande. Användaren kan t.ex. med hjälp från en GPS-mottagare själv ange sin position. Detta ger användaren en klar uppfattning om när data angående dennes position röjs. Förfarandet vid användandet av en GPS-mottagare behöver inte nödvändigtvis innebära att någon ytterligare part är inblandad vid positioneringstjänsten då tjänsten kan utföras lokalt hos användaren.

3.4.2 Roller i positionsbaserade tjänster

En LBS-tjänst inkluderar flera viktiga roller:

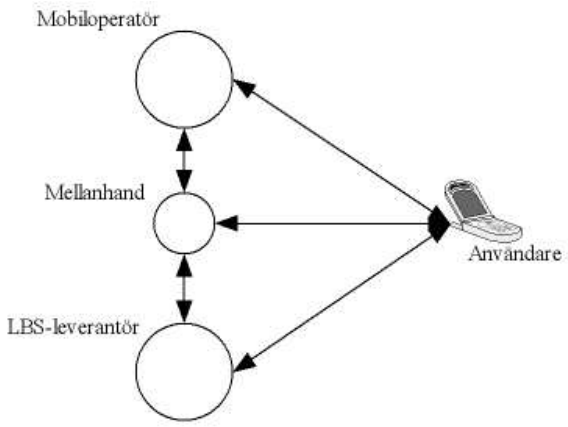
Användare: En användare villig att avslöja information angående dess position i utbyte mot en viss tjänst. Om enheten själv ansvarar för inhämtning av positioneringsdata agerar denne även som en *positionsleverantör*

Positionsleverantör: Detta är källan till informationen angående användarens position. Ofta agerar mobiloperatörerna som positionsleverantörer. Men i de fall där användarenheterna själva har möjlighet att ange sin position, agerar de som positionsleverantör.

Mobiloperatör: En mobiloperatörs ansvarsområden innefattar bland annat ansvaret för den trådlösa infrastrukturen (t.ex. GSM) över vilken kommunikationen mellan användare och LBS-leverantör tar plats.

LBS-leverantör: En LBS-leverantör levererar själva tjänsten till användarna. För att leverera en tänst åt en användare behöver LBS-leverantören i sin tur erhålla information angående användarens position från positionsleverantören.

Mellanhand: Mellanhanden är en icke obligatorisk entitet som kontrollerar hur detaljerad information angående en användares position skickas. Mellanhanden meddelar leverantören av LBS-tjänsten den data som skickats. Figur 3.3 illustrerar interaktionen med en mellanhand.



Figur 3.3: LBS-tjänst med en interagerande mellanhand.

En operatör och en LBS-leverantör behöver inte nödvändigtvis vara skilda organisationer. Ett problem rörande användarnas personliga integritet kan uppstå när mobiloper-

atörerna själva levererar LBS-tjänster i deras eget nät. Det innebär att de både äger och kontrollerar de olika LBS-applikationerna. Mobiloperatören är i ett läge där den har möjligheten att sammanställa vilken information som skickas till en användarna beroende på vilken position denne befinner sig på, för att därefter sammanställa detaljerade kundprofiler. Vidare är mobiloperatören i ett läge där den kan "låsa in" sina kunder att använda just deras "standarder" och tjänster.

LBS-applikationer kan delas in i enanvändar- och fleranvändar-applikationer (på engelska kallad *peer-to-peer*). I enanvändar-applikationer förekommer en direkt koppling mellan användarna och LBS-leverantören och vidare förekommer ingen direkt kommunikation mellan de olika användarna. I fleranvändar-applikationer kan användarna kommunicera direkt med varandra. LBS-leverantören huvuduppgift i detta sammanhang är att bistå användarna att upprätta anslutningen mellan varandra. Tabell 3.2 beskriver klassificieringen ev respektive LBS-applikation beroende på vilken tjänst som erbjuds.

	Pull LBS	**Push LBS**
Enanvändare	positioneringstjänster, navigeringstjänster.	mobil handel, upplysningstjänster.
Fleranvändare	-	lokalisering av användare, mobila datingtjänster

Tabell 3.2: Klassificering av LBS-applikationer

3.5 Model View Controller

Model-View-Controller [7] (MVC) är en mjukvaruarkitektur som separerar en applikations datamodell, användaregränssnitt och kontrollogik till tre distinkta komponenter; *modell*, *vy* och *kontroll*. En sådan indelning tydliggör en avgränsning mellan applikationens olika delar. Det erbjuder en mer lättläst kod, samt ett strukturerat sätt att bygga sina applikationer.

Vyn hanterar utdata till användaren. Den visar modellens tillstånd, vanligtvis genom ett för applikationen lämpligt användargränssnitt. Vyn erhåller förändringar av modellen. Dessa förändringar meddelas sedan till kontrollern. Vidare har vyn ett *observer*-mönster[6], vilket definierar ett 1-till-m beroende mellan objekten. Detta innebär att när ett objekt förändrar tillstånd blir alla intressenter varse och uppdateras automatiskt.

Kontrollern hanterar indata från användaren. Den bearbetar händelser (ofta genom kontroll av indata), och anropar modellen som reagerar därefter. Vyn och kontrollern kan i vissa fall kombineras. De är starkt kopplade och har ett 1-till-1 förhållande.

Modellen är själva "spindeln i nätet". Den representerar och stödjer själva logiken av problemet. Modellen meddelar sitt tillstånd till samtliga vyer. Kommunikationen sker med händelser i ett *observable*-mönster, vilket innebär att det är ett objekt som iaktas av en eller flera observatörer.

Flöde av händelser

MVC-modellen kommer i flera olika smaker. Flödet av händelser, vilket illustreras i figur 3.4 kan generellt beskrivas som följande:

1. Användare interagerar med någon typ av användagränssnitt (t.ex trycker på en knapp).

2. Kontrollern tar emot en notifikation angående användarens begäran från användargränssnittets objekt.

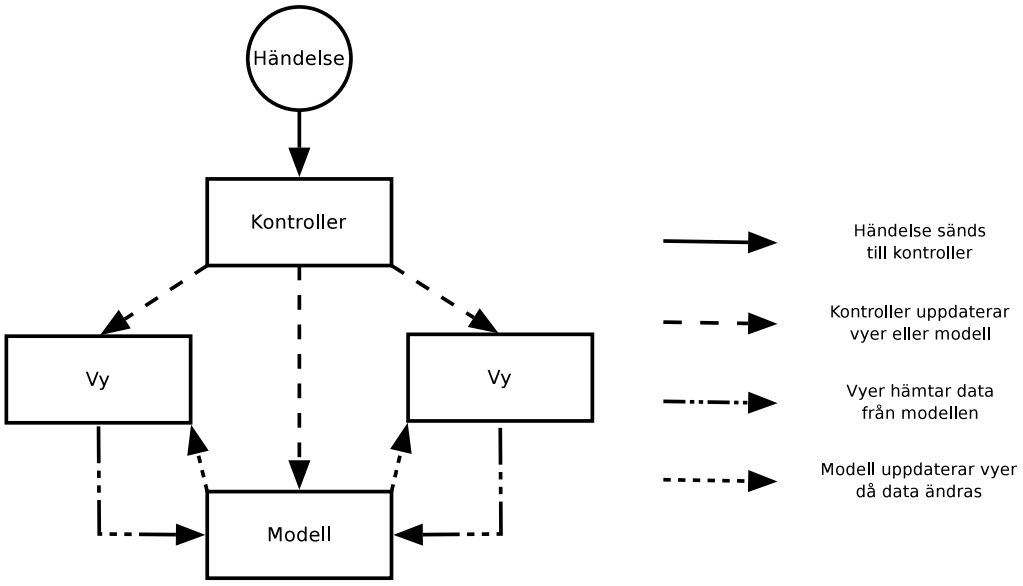
3. Kontrollern anropar modellen vilken uppdateras för att på ett lämpligt sätt uppfylla användarens begäran.

4. Modellen meddelar vyn med de ändringar som utförts. Detta kan i vissa fall gå via kontrollern eller på annat sätt.

[6]Ett designmönster som innebär att observera tillståndet av objekt i program.

5. Vyn använder modellen för att återge ett lämpligt gränssnitt. I vissa fall har vyn inte har direkt kommunikation med en extern modell. I dessa fall trycks datan på vyn av kontrollen från den externa modellen.

6. Användargränssnittet inväntar ytterligare instruktioner från användare, vilket inleder en ny cykel.



Figur 3.4: Flödet i en MVC-applikation.

3.5.1 Fördelar och problem

Genom att använda MVC kan flera olika användargränssnitt utformas till en och samma modell. Utvecklingen av modellen och användargränssnittet kan därför ske både parallellt eller separat. En annan fördel med är att MVC tillåter återanvändning av de olika delarna i systemet. Distributionen av en applikation förenklas då MVC kan ses som en klient/server-lösning (indirekt kommunikation mellan GUI och modell).

Det kan ibland uppstå problem vid särskiljning av de logiska eller tekniska rollerna. I vissa utvecklingsverktyg, exempelvis Qute (Qt) och Visual Basic (VB) är vy och kontroller samma objekt. Implementationen av MVC-modellen i mycket små projekt kan i vissa fall anses omständigt, då upplägget generar en hel del extra kod. Vissa problem kan uppstå, speciellt vid distribuerade tillämpningar, då mycket trafik genereras mellan vyn och modellen även vid små förändringar.

3.6 Bestämning av klientegenskaper

Under de senaste åren har en stor ökning av mängden Internet-anslutna enheter skett. Den största delen av dessa nyanslutna enheter utgörs av "små" enheter, såsom mobiltelefoner och handdatorer.

Dessa enheter tenderar att vara mer begränsade till sin karaktär än vanliga persondatorbaserade klienter, framförallt med tanke på skärmstorlek och dataformat som stöds. Dessa klienters egenskaper skiljer sig vitt, inte endast jämfört med traditionella klienter, utan även sinsemellan.

Detta scenario, där en mängd klienter med egenskaper som kraftigt skiljer sig åt kräver en metod för att kunna identifiera en klients egenskaper i syfte att kunna anpassa en tjänst på ett sådant sätt att den passar en given klients karaktäristik. Två kompatibla standarder för att representera egenskaperna hos en klient har tagits fram, Composite Capabilities / Preferences Profile (CC/PP) av World Wide Web Consortium (W3C), och User Agent Profile (UAProf) som har utvecklats av WAP[7] Forum. Båda dessa standarder är baserade på Resource Description Framework (RDF).

[7]Wireless Application Protocol, internationell standard för kommunikation mellan trådlösa enheter. T.ex. Internet-åtkomst via en mobiltelefon.

3.6.1 Resource Description Format

RDF [8] är en teknologi framtagen av W3C för att kunna representera metadata, d.v.s. information om information. Från början var denna standard avsedd att tillhandahålla bättre möjligheter för indexering av websidor. Den relativa enkelheten hos RDF-formatet har dock medfört att det har används inom andra områden, t.ex. representation av klienters egenskaper.

En vanlig representation av RDF-data är genom serialisering som XML, denna representation används t.ex. av CC/PP och UAProf.

Ett RDF-dokument representerar sin information i form av en samling utsagor om resurser, där varje utsaga består av *subjekt*, *predikat* och *objekt*. Varje sådan kombination av subjekt, predikat och objekt kallas för en trippel.

- *Subjekt:* Subjektet är resursen, det som beskrivs av trippeln.
- *Predikat:* Predikatet anger vad som representeras av relationen mellan subjekt och predikat, dvs. vilken egenskap hos subjektet som avses med beskrivningen.
- *Objekt:* Värdet för den egenskap som beskrivs.

3.6.2 Composite Capabilities / Preferences Profile

CC/PP [9] är en standard, framtagen av W3C, för att beskriva egenskaper och användarinställningar hos en klient, så att en server kan anpassa en tjänst till klienten. Inget protokoll för hur denna data skall överföras definieras dock i standarden.

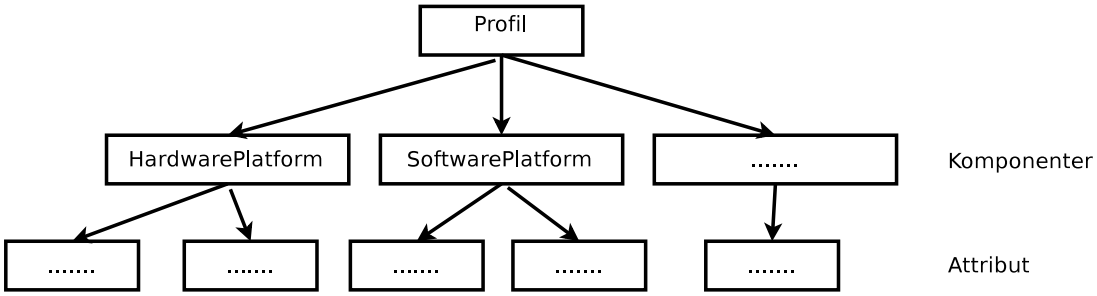
En CC/PP-profil kan ses som en trädstruktur i två nivåer (se figur 3.5), där en profil består av en eller flera komponenter, vars egenskaper beskrivs av ett eller flera attribut. Denna struktur representeras vanligen som en XML-serialisering av RDF-data. Ett exempel på en sådan serialisering ges i appendix B.

Attributen hos en komponent kan bestämmas på ett flertal sätt; attributen kan vara angivna direkt i profilen, eller anges utifrån en annan profil innehållande förvalda värden.

Om båda typerna skulle finnas tillgängliga har explicita värden i profilen alltid precedens över de förvalda värdena.

Flervärdesattribut i en CC/PP-profil representeras genom att de läggs under en anonym nod i trädstrukturen, som tilldelas ett attribut namngett *rdf:type* som anger om attributet representerar en sekvens eller en oordnad mängd. Sekvenser är av typen *seq*, medan mängder är av typen *bag*.

En egenskap hos CC/PP är att det inte definierar någon vokabulär (en uppsättning komponenter och attribut). Detta medför att en vokabulär måste fastställas för att kunna beskriva en viss resurs. CC/PP kan alltså inte ensamt användas för att beskriva egenskaper hos en resurs, utan en vokabulär måste tillhandahållas.



Figur 3.5: Strukturen hos en CC/PP-profil.

3.6.3 UAProf

UAProf [10], vilket är utvecklat av WAP Forum, baseras på CC/PP. Precis som CC/PP beskrivs en enhet av en trädstruktur i två nivåer, bestående av komponenter som beskrivs gentemot CC/PP.

En viktig skillnad är dock att UAProf tillhandahåller en standardiserad vokabulär, lämplig för att beskriva egenskaper hos WAP-telefoner och liknande enheter.

En UAPRof-baserad profil utgörs av fem obligatoriska komponenter:

- *HardwarePlatform:* denna komponent beskriver egenskaper hos terminalen (t.ex. mobiltelefonen), såsom skärmstorlek och tillgänliga metode för in- och utmatning.
- *SoftwarePlatform:* beskriver egenskaper hos terminalens mjukvara, t.ex. version av operativsystem och giltiga metoder för ljudkodning.
- *NetworkCharacteristics:* information om nätverket, såsom tillgängliga databärare och terminalens anslutningstyp.
- *BrowserUA:* beskrivning av egenskaperna för terminalens webbläsare.
- *WapCharacteristics:* egenskaper för terminalens webbläsare för tolkning av Wireless Markup Language (WML).

Dessa komponenter innehåller attribut, vilka är fastslagna i det standardiserade vokabulär som UAProf specificerar. Varje attribut kan ha en av de tre typerna *simple* (vilket anger att attributet utgörs av ett enda värde), *bag* eller *seq*. Egenskaperna för *seq* och *bag* förklarades i avsnitt 3.6.2. Ytterligare komponenter av godtycklig typ kan definieras vid behov.

Vidare anges typen av den data som attributet utgörs av enligt följande datatyper:

- *String:* en textsträng.
- *Boolean:* ett sanningsvärde, betecknat som *Yes* eller *No*.
- *Number:* ett positivt heltalsvärde.
- *Dimension:* två positiva heltalsvärden, separerade av ett 'x'.

Ett attribut associeras dessutom med en policy, vilken avgör vilket attribut som används i det fall där flera attribut med samma namn finns. Dessa är:

- *Locked:* attributets slutgiltiga värde är dess första förekomst.

- *Override:* attributets slutgiltiga värde är dess sista förekomst.
- *Append:* det slutgiltiga värdet är en lista, bestående av värden från alla förekomster.

3.6.4 Wireless Profiled HTTP

Hypertext Transfer Protocol (HTTP) är ett protokoll för att möjliggöra informationsutbyte mellan maskiner över ett nätverk. Protokollet bygger på att en klient begär data från en server, vilken svarar med önskad data.

En begäran från en klient inleds med en kod, vilken anger vilken handling klienten vill utföra. Exempel på detta är `GET`, vilket anger att klienten vill hämta data, eller `PUT`, som anger att uppladdning av data önskas. En begäran av `GET`-typ följs därefter av information gällande vilken fil som begärs, och vilken version av protokollet som används.

Denna första rad kan följas av en eller flera rader innehållande parametrar, där raden inleds med parameternamnet följt av ett komma, för att sedan följas av parameterns värde. Denna följd av data, bestående av begäran och eventuella parameterar benämns *header*.

WAP 2.0-kompatibla enheter transporterar profildata med hjälp av Wireless profiled HTTP (W-HTTP) [10], där headern på klientens begäran har försetts med extra information för att kunna beskriva klientens karaktäristik.

En URL till profilen sänds som parametern `X-Wap-Profile` i en header tillhörande en begäran av GET-typ. Ett exempel på en sådan header, skickad av en SonyEricsson Z600, kan ses nedan.

```
GET /~skellington/oldfiles HTTP/1.1
Host: bekk.hn.org
X-Wap-Profile: "http://wap.sonyericsson.com/UAprof/Z600R601.xml"
.....
```

En header kan även inkludera en eller flera differenser[8]. Dessa differenser appliceras

[8]Data i XML-format, vilken anger på vilka punkter klientens egenskaper skiljer sig från de som angivits

mot den huvudsakliga profilen för att erhålla en korrekt beskrivning av klientens karaktär. XML-serialiseringen av dessa differenser sänds direkt i headern, som parametern `X-Wap-Profile-Diff`. Dessa differenser måste sedan slås samman med den huvudsakliga profilen för att ta fram en slutgiltig profil. Ett exempel på en header som inkluderar differensdata, sänd av en SonyEricsson T68i, kan ses nedan.

```
GET / HTTP/1.1
Host: bekk.hn.org
X-Wap-Profile:
"http://wap.sonyericssonmobile.com/UAprof/T68R501.xml",
"1-jAyHPa0CPRS1qvjtFJPBaQ==", "2-XnrTOLDzBJdZHN2vSasoNA= =",
"3-Z+lJzsFTo4qT8wYH1WfzIQ==", "4-HC5l5j+eQ9tPRpdhsseJIQ=="
......
Accept-Language: en
User-Agent: SonyEricssonT68/R501
Accept-Charset: *
Accept-Encoding: deflate, gzip
TE: deflate, gzip
X-Wap-Profile-Diff: 1; <?xml version="1.0"?><rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:prf="http:// www.wapforum.org/UAPROF/ccppschema-19991014#">
......
X-Wap-Profile-Diff: 2; <?xml version="1.0"?><rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:prf="http:// www.wapforum.org/UAPROF/ccppschema-19991014#">
.......
```

Som illustrerats i det avkortade exemplet ovan efterföljs URL:en av ett eller flera extra

i profilen.

fält, vart och ett innehållande ett indexvärde och en kontrollsumma, vilken kan användas för att verifiera meddelandets integritet. Dessa indexvärden anger i vilken ordning applicering av ändringarna skall göras.

Efter detta följer själva datan att applicera, som sektioner av typen `X-Wap-Profile-Diff`. I dessa sektioner föregås datan av sekvensnummer, vilka indexerar datan så att den kan appliceras i den ordning som anges i `X-Wap-Profile`.

Den slutgiltiga profilen bestäms genom att, enligt de regler för resolution och ordning som tidigare specificerats, till den ursprungliga profilen applicera de eventuella ändringar som sänds i headern.

3.7 Servlets och Java Server Pages

Servlets tillsammans med Java Server Pages (JSP) är en teknologi som blivit mycket populär för att utveckla dynamiska webbsidor [11].

3.7.1 Servlets

En servlet är ett javaprogram som körs på en webb- eller applikationsserver. En servlet måste kompileras av utvecklaren innan den tas i bruk. Den agerar som en mellanhand mellan inkommande HTTP-anrop från webbläsare (eller andra HTTP-klienter) och andra applikationer som finns på HTTP-servern (t.ex. databaser eller olika tjänster). Figur 3.6 ger en översiktlig bild över detta.

En servlets uppgifter består i att:

1. *Läsa data som explicit angivits av en användare.* Sådan data kan vara värden som angivits i något inmatningsfält på en webbsida. Det kan även vara data som kommer från en java-applet[9].

[9] körbar javakod som kan bäddas in i en webbsida och exekverar lokalt hos klienten.

2. *Läsa den data som implicit skickas från webbläsaren.* Denna typ av data är inte direkt angiven av någon användare. Data som implicit skickas av webbläsaren inkluderar bland annat cookies, information angående mediatyper och hur data skall vara strukturerad.

3. *Generera resultat.* Servleten utför den uppgift den blivit ombedd att göra. Uppgiften kan bestå av att kommunicera med en databas, administrera en dator, utföra beräkningar, osv.

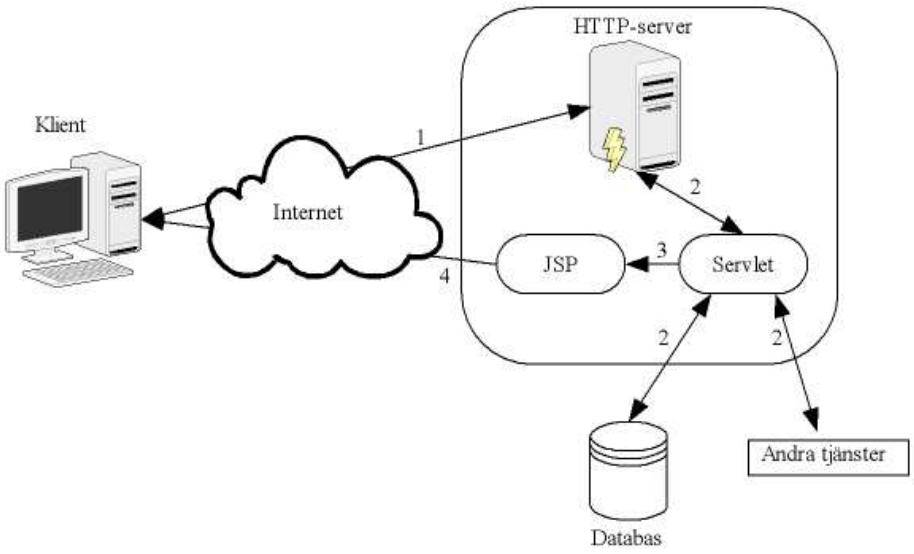
4. *Skicka explicit data till användaren.* Denna data kan till exempel vara själva dokumentet. Dokumentet kan skickas i olika format: text (t.ex. HTML eller XML), binärt (t.ex. bilder), eller komprimerat. Det vanligast förekommande tillvägagångssättet är att returnera resultatet i form av en HTML-sida.

5. *Skicka implicit data till användarens klient.* Denna typ av data inkluderar: vilken typ av dokument som returneras (t.ex. HTML), vilka cookies och inställningar som önskas, och så vidare.

3.7.2 Java Server Pages

Mycket förenklat kan JSP ses som HTML-sidor med integrerad javakod, medan servlets kan ses som javaprogram med integrerad HTML-kod. Faktum är att ett JSP-dokument är en annan metod för att skriva en servlet. JSP-sidor översätts till servlets, dessa servlets kompileras i realtid för att sedan köras under exekvering. En regel är att servlets används för att representera logiken i applikationen, medan JSP-sidor används för att representera presentationen av applikationen. Genom utveckla applikationer enligt detta designmönster erbjuds en rad fördelar vilket diskuterades mer i detalj i avsnitt 3.5.

Figur 3.6 illustrerar ett översiktligt händelseförlopp med servlets och JSP. Händelse$\pm - förloppetbeskrivsnedan.$



Figur 3.6: Händelseförlopp med JSP och servlets.

1. En klient skickar en begäran till en webbserver.

2. En servlet behandlar klientens begäran och gör diverse anrop mot externa processer.

3. Resultatet från servletens exekvering kompileras tillsammans med en JSP-sida.

4. Den kompilerade JSP-sidan returneras som HTML till klienten.

Kapitel 4

Applikationen

I detta kapitel kommer applikationens arkitektur och implementation att diskuteras. Det är dock lämpligt att här nämna att diskussionen kommer att hållas på en relativt övergripande nivå, då det faktum att applikationen ej kan anses helt slutförd gör en beskrivning på detaljnivå irrelevant. En mera utförlig redogörelse för implementationens status återfinns i kapitel 5.

4.1 Applikationens arkitektur

Applikationens design lägger stor vikt vid att applikationens olika funktionella delar ska isoleras från varandra (i enlighet med designmönstret MVC vilket diskuterades i kapitel 3.5). Detta hjälper till att uppnå ett flertal önskvärda mål:

4.1.1 Isolering av plattformsberoende

Vissa delar, såsom inhämtning av koordinater från klienten, kan ej lösas på ett fullt plattformsoberoende vis. Isolering av dessa delar gör plattformsspecifika modifieringar enklare.

4.1.2 Datakälleoberoende

Byte av bakomliggande datakälla för lagring av bild- och användardata bör ej kräva förändringar i applikationen. I den mån sådana krävs, ska de kunna göras inom en så isolerad del som möjligt.

Detta tillåter även applikationen att med relativt enkla och väl avgränsade modifieringar kunna skalas för att hantera ökade krav på prestanda. För ett litet antal klienter kan det exempelvis anses fullt tillräckligt att använda sig av filsystemsbaserad lagring av kartdata, medan ökad användning kan kräva en databas för detta ändamål, eller rentav användning av någon bakomliggande klusterlösning (flera samarbetande datorer).

Vår design medger att sådana ändringar genomförs utan att övriga delar av applikationen påverkas.

4.1.3 Isolering av specifik funktionalitet

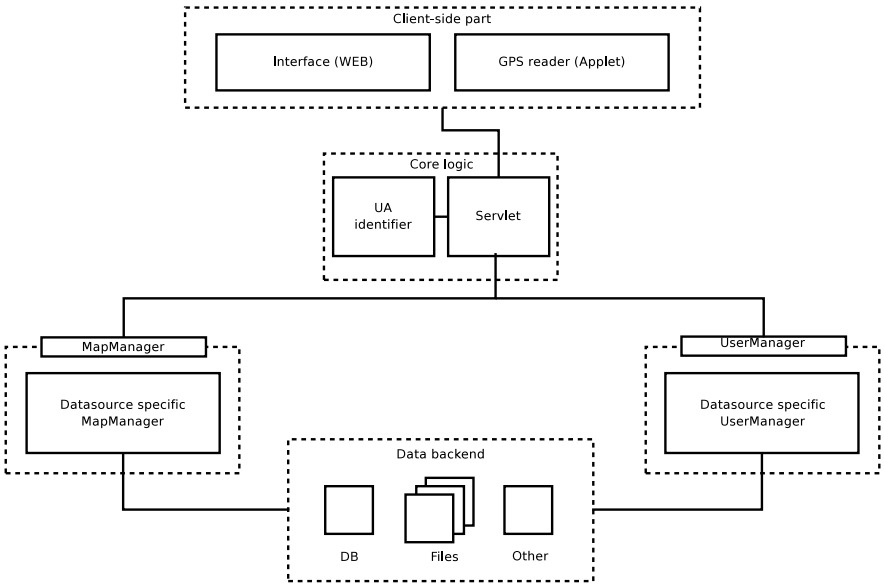
Klasser inom olika verksamhetsområden bör kommunicera med andra funktionalitetsdelar genom ett fåtal, väl definierade gränssnitt. Detta bidrar till att göra applikationen enklare att modifiera och underhålla.

4.2 Applikationens beståndsdelar

I figur 4.1 beskrivs övergripande applikationens arkitektur. Som synes kan applikationen anses bestå av fem huvudsakliga delar.

4.2.1 Användardel

Denna del representerar applikationens gränssnitt mot användaren, och låter denna interagera med applikationen. Här finns även funktionalitet för att hämta in klientens position, vilket för närvarande har implementerats i form av en applet.



Figur 4.1: Översikt över applikationens arkitektur.

Själva gränssnittet är implementerat i form av HTML-sidor, vilka kan använda sig av ett flertal CSS[1]-dokument för att anpassa gränssnittet till klienter med olika krav.

4.2.2 Applikationskärna

I denna del ligger applikationens sammanhållande logik, i form av en servlet. Här finns även logik för att identifiera en mobil klients karaktäristik utifrån dess UAProf-dokument (se avsnitt 3.6.3).

4.2.3 Karthantering

Här finns funktionalitet för olika aspekter av karthantering, såsom att kunna söka efter kartor över ett visst område eller att kunna markera positioner på en given kartbild för senare visning för klient.

[1] Cascading Style Sheets, innehåller regler för att beskriva presentationen av ett strukturerat dokument.

Detta är även en logisk plats att lägga funktionalitet för administration av kartdata, såsom borttagning, modifiering och tilläggande av kartdata.

4.2.4 Användarhantering

Denna del låter applikationen hantera olika aspekter såsom inloggning, kontroll av användarnamn, rättigheter och lösenord. Dock kan denna del inte anses tillhöra applikationens kärnfunktionalitet. Den inkluderades främst för framtida utveckling av applikationen.

4.2.5 Datadel

Här sköts all datalagring, av såväl kartdata samt eventuell användardata. Denna del kan vara implementerad på en stor mängd olika sätt och designen ställer i sig inga krav på en viss uppsättning specifika lagringsmetoder.

Tänkbara möjligheter kan vara filsystem, databas eller andra datalagringslösningar.

4.3 Användargränssnitt

Då gränssnittet i skrivande stund ej är färdigimplementerat kommer ingen detaljbeskrivning av detta att göras i denna sektion. Dock kommer en övergripande diskussion av dess implementation och funktion att ges.

Då uppdragsgivaren hade specificerat att applikationen skulle använda HTTP för kommunikation mellan klient och server samt att användarens befintliga webbläsare skulle användas för att interagera med applikationen var valet av XHTML / CSS för implementation av gränssnittet enkelt (XHTML är HTML, anpassat så att det är en giltig form av XML).

På grund av att applikationen skall kunna användas från mobila klienter var det här nödvändigt att kontrollera vilka teknologier för webbaserade gränssnitt sådana enheter kan garanteras stödja. Något som kunde anses ha garanterat stöd är WML (ett webbspråk

liknande HTML, men avsett för mobiltelefoner), detta skulle dock medföra att två helt skilda användargränssnitt skulle behöva utvecklas och underhållas, vilket ansågs önskvärt att undvika.

Vid läsning av standarder för WAP 2.0 fanns det att denna standard garanterar att enheter vilka implementerar den garanteras stödja en delmängd av XHTML kallad XHTML Mobile Profile, vilken visade sig innehålla en tillräcklig delmängd ur XTHML för att möjliggöra utveckling av en enda uppsättning sidor för gränssnittet, lämpat för alla enhetstyper.

Vidare fanns att WAP 2.0 specificerar stöd för CSS version 2, vilket möjliggör separering av sidornas innehåll och presentation. Detta medför att, om behov skulle finnas, olika gränssnitt för mobila och fasta klienter kan skapas genom att använda olika CSS-dokument för de olika typerna.

Av dessa anledningar fattades beslutet att kräva att mobila klienter skall stödja WAP 2.0, vilket ej upplevdes vara ett orimligt krav, då stöd för denna standard har varit vanligen förekommande sedan ett flertal år.

4.3.1 Applikationen

Gränssnittet i den grundläggande applikationen är mycket enkelt, och erbjuder endast den nödvändigaste funktionaliteten.

Då användaren först når applikationens startsida erbjuds denne ett val mellan att läsa in koordinaterna automatiskt (vilket kommer att diskuteras närmare i avsnitt 4.3.2), eller att manuellt skriva in sina koordinater.

Då användaren har sänt sin position på ett av ovanstående vis sänds denne vidare till en sida där en kartbild över positionen visas, med användarens position markerad. På denna sida erbjuds även användaren att välja mellan flera kartbilder som illustrerar dennes position, om flera finns.

4.3.2 Inläsning av koordinater

Ett önskemål från uppdragsgivarens sida var att koordinatinläsning skulle automatiseras så långt som möjligt. På grund av applikationens webbläsar-baserade natur visade sig detta vara en stor utmaning.

Det metod som passade oss bäst var att lägga funktionaliteten i en java-applet. På grund av att NMEA-protokollet specificerar seriell kommunikation tycks många datoranslutna GPS-mottagare emulera en serieport, varför ett API för kommunikation med hårdvara ansluten till sådana portar eftersöktes.

Det API för detta som hittades var `javax.comm` [12], vilket är specificerat av Sun, men som inte förekommer i javas standardbibliotek, då en bakomliggande plattformsspecifik implementation av den bakomliggande funktionaliteten krävs. Sådana implementationer finns i dagsläget för windows, solaris och linux. Detta medför att användaren av appleten måste installera detta API, vilket inte är en så smidig lösning som skulle önskas, men som dock är den bästa som har kunnat hittas.

En klass för att läsa positionsdata från en GPS-mottagare har skrivits, med hjälp av detta API, dock har denna ännu ej satts in i en applet för ändamålet. Denna klass, `GPSReader`, tillhandahåller följande funktionalitet:

- Lista samtliga tillgängliga serieportar på systemet.
- Testa om en given serieport har en NMEA-kompatibel enhet ansluten.
- Söka efter NMEA-enheter på samtliga portar.
- Läsa av positionsdata från en NMEA-kompatibel enhet.

Automatisk avkänning av GPS-enheter görs genom att lyssna på den angivna porten i två sekunder, och därefter matcha eventuell mottagen data mot ett reguljärt uttryck, matchande ett korrekt NMEA-meddelande.

Tolkning av koordinater genomförs genom att läsa ett NMEA-meddelande av typ GLL från mottagaren, och sedan parsa ut relevant data från detta.

4.4 Karthantering

I en applikation av denna typ är naturligtvis karthantering en central del, varför stor vikt lagts på denna del under utvecklingen av applikationen. Detta avsnitt beskriver hur vi löst karthanteringen.

Karthanteringsdelen är uppdelad i ett flertal `klasser` och `interface`. Interfacet `MapManager` representerar övriga applikationens åtkomstväg till karthanteringsfunktionaliteten. Den specifika implementationen av karthanteraren skall implementera detta interface, så att applikationen lätt kan flyttas mellan olika datakällor. Två implementationer av denna har skapats, en MySQL-baserad variant namngiven `MySQLMapManager`, vilken inte kommer att användas i den första versionen av applikationen, då krav på databas ansågs överdrivet i den första, minimala versionen.

Den variant som används i den slutförda applikationen är en enklare, filbaserad variant (`FileMapManager`) vilken ej stödjer administration av kartdata, såsom tilläggning och radering av kartpaket. Denna modul tillhandahåller metoder för att ladda lagrade kartpaket från en katalog i det lokala filsystemet, och sedan kunna söka inom dessa.

Två metoder för att finna en kartbild över en position finns, en metod vilken returnerar en `Enumeration` över samtliga tillgängliga kartor, och en metod som endast returnerar den kartbild som täcker det största området runt den angivna positionen.

En existerande kartbild representeras i applikationen av två klasser, där `MapImage` representerar själva kartbilden och tillhandahåller metoder for att placera markeringar på denna. `MapData` innehåller metadata om en karta, såsom information om det område bilden täcker. Denna uppdelning i två klasser gjordes för att kunna minimera den mängd data som hålls i minnet under normal användning.

Medan metadata om kartor bör vara, och är, laddad under applikationens körning, vore det onödigt att konstant hålla all bilddata laddad, denna bör i stället laddas endast då den krävs. Av denna anledning gjordes ovanstående separation.

Koordinater representeras av klassen `GPSCoordinate`, vilken innehåller data gällande en klients longitud och latitud, och tillhandahåller metoderna `northOf`, `southOf`, `eastOf` och `westOf`, användbara vid jämförelse av koordinater.

Klassen `MapPackage` är avsedd att kunna representera ett kartpaket och därmed innehålla de `MapData`-objekt som representerar de till paketet tillhörande kartorna, samt metadata för paketet i sin helhet. Denna klass är avsedd att förenkla utvecklandet av administrativa funktioner för karthantering, och har därför utelämnats från den enklare, filbaserade versionen av applikationen.

4.4.1 Paketering av kartdata

För att applikationens lagring av kartor inte skall bli ohanterlig när mängden kartor överstiger ett fåtal krävs något sätt att gruppera, administrera och distribuera kartor.

En karta för en positionsbaserad-applikation kan anses bestå av två delar, *bilddata* och *geodata*. Geodata utgör den information som beskriver vilka koordinater bilden täcker in, medan bilddatan utgör själva kartbilden, vilken ges en mening i applikationen av geodatan.

Lagring av bilddata

Bilddata kan uppdelas i två kategorier: *vektorbaserade* respektive *rasterbaserade* format. Rasterbaserade format lagrar bilden i form av färgdata för varje individuell pixel, medan vektorbaserad bilddata lagras i form av information om vektorer, vilket gör det möjligt att utifrån denna data återställa bilden.

Applikationens karthantering baserades på rasterformat, främst på grund av att Java i sitt standard-API har ett utmärkt stöd för att hantera många vanligen förekommande rasterformat, såsom JPEG (Joint Photographic Experts Group), TIFF (Tagged Image File

Format) och PNG (Portable Network Graphics). De enda begränsningarna gällande vilka rasterformat som kan användas av applikationen sätts av Javas `ImageIO`-API, vilket kan utökas med stöd för ytterligare format, om så skulle krävas.

Lagring av geo- och metadata

För lagring av geo- och metadata valdes att basera filformatet på XML, då det ger goda möjligheter att kunna beskriva strukturerad data i ett format som relativt enkelt kan hanteras manuellt. En annan faktor som spelade in vid detta beslut var att Java tillhandahåller API:er för enkel hantering av XML-baserade dataformat.

Den data som lagras för varje individuell kartbild är följande:

- `Name:` ett namn, lämpligen beskrivande kartbilden.
- `Filename:` används för att kunna koppla den metadata som anges i XML-filen till en kartbild.
- `Type:` identifierar typen av bilddata genom att ange dess MIME[2]-typ.
- `Topleftcoord, toprightcoord:` koordinaterna för övre vänstra respektive nedre högra hörnen.

Paketformat

Behovet att kunna gruppera relaterade kartor var redan tidigt i planeringen uppenbart. Den metod att uppnå detta som valdes var en metod, liknande den som vanligen används av ett flertal Java-relaterade teknologier.

För att kunna gruppera, hantera och distribuera kartor grupperas ett flertal kartor i en komprimerad fil i zip-format, där kartbilderna läggs i en underkatalog namngiven `maps`.

[2]Multipurpose Internet Mail Extensions, anger i detta fall information om filens innehåll.

Geo- och metadata, samt information om kartpaketet anges i en fil namngiven `mmd.xml`, vilken väntas finnas i roten på zip-arkivet.

Denna lösning påminner om den som används för distribution av webapplikationer i *war*-format (Web Aplication Archives), och valdes då den kan hanteras med hjälp av vanligen förekommande applikationer, utan behov av specialverktyg.

Den data som kan anges i filen mmd.xml för att beskriva ett kartpaket är följande:

- `Name:` kartpaketetets namn.
- `Vendor:` kartpaketetets leverantör.
- `Description:` en beskrivning av kartpaketetet.
- `Date:` ett datum i formatet *ÅÅÅÅMMDD*, lämpligt för att ange tidpunkt för kartornas skapande.

En komplett beskrivning av formatet kan ses i appendix A, där formatets specifikation finns i form av en DTD (Document Type Definition) vilken specifierar syntaxen för specifikationen.

4.5 Användarhantering

Användarhanteringen var egentligen en funktion som lades till och implementerades som en utökad funktion. Anledningen för detta var att på något sätt kontrollera vilka som har rättigheter att använda och administrera den tjänst som beskrivits i denna uppsats. Med avseende på den miljö i vilken tjänsten skulle erbjudas (över Internet) ansågs det som en logisk utökning.

4.5.1 Användartyper

Applikationen har stöd för två olika användartyper: användare och administratör.

Användare

Användaren benäms i applikationen som `User`. Det är den individ som är avsedd att använda sig av den LBS-tjänst (se avsnitt 3.4) applikationen erbjuder.

Objektet User innehåller ett användarnamn, ett lösenord samt en flagga `isAdmin`, vilken kan anta ett booleskt värde. Denna flagga är satt till `true` ifall användaren har administratörsrättigheter, `false` ifall användaren inte har administratörsrättigheter.

Administratör

Administratören är en användare med med flaggan `isAdmin` satt till true. Denne har förutom de rättigheter en användare har dessutom administratörsrättigheter, t.ex. lägga till användare, administrera kartor, etc.

4.5.2 Användarhanterare

Hanteringen utav användare sköts av en användarhanterare. Den grundfunktionalitet som måste implementeras hos en användarhanterare ges av ett interface vilket är namngivet `UserManager`. Då användarhanteraren innehåller implementationen för administration av användare och användardata (även karthantering) har denne en interaktion med någon form av datalagringteknik. Användarhanteraren tenderar att bli datalagringsspecifik. Vid konstruktion av en användarhanterare måste därför hänsyn tas till i vilken form användardata skall lagras (t.ex textfiler, databas). För att garantera grundfunktionaliteten hos en användarhanterare implementeras metoder från detta interface. Givetvis kan användarhanteraren utökas med ytterligare funktionalitet om det önskas.

De metoder interfacet innehåller är:

- *boolean createUser(String, String, boolean):* är till för att skapa användare. Den tar emot ett användarnamn, ett lösenord samt en booleskt värde som anger ifall användaren är administratör och returnerar `true` ifall proceduren lyckades.

- *boolean deleteUser(String):* är till för att ta bort användare. Den tar emot ett användarnamn och returnerar `true` ifall proceduren lyckades.
- *boolean updateUser(User):* är till för att kunna kunna uppdatera befintlig användardata. Den tar emot en `User`-objekt och returnerar `true` ifall proceduren lyckades.
- *boolean userExists(String):* anger om användaren existerar. Den tar emot ett användarnamn och returnerar `true` ifall användaren existerar.
- *User getUserByName(String):* är till för att erhålla befintlig användardata. Den tar emot ett användarnamn och returnerar ett `User`-objekt.

4.5.3 Applikationens användarhanterare

Användarhanteraren vilken benäms *SQLUserManager*är implementerad för att använda sig av en MySQL-databas för lagring utav användardata. Den är således anpassad för just denna typ av databas. Användarhanteraren har implementerat de metoder som är specifierade från interfacet, samt ytterligare funktionalitet i form av fel- och valideringskontroller av inmatad data från användare. Denna implementation av användarhanterare kommer dock inte att användas i den första versionen av applikationen, då krav på databas ansågs överdrivet i den första minimala versionen.

Den första versionen använder sig av en enklare variant (*FileUserManager*) vilken ej stöder administration av användare samt användardata. FileUserManager implementerar endast en metod, en metod vilken kontrollerar användarnamn och lösenord.

Lagring av användardata

Lagringen utav användare och användardata sker i den första versionen av applikationen lokalt i en textfil. Användardatan måste skrivas dit manuellt, detta sköts alltså inte av användarhanteringen i detta fall.

Kapitel 5

Fortsatta arbeten

Applikationen uppfyller i sitt grundutförande de krav som har ställts, och kan användas för det avsedda syftet. Dock har applikationen potential och utbyggnadsmöjligheter, vilka skulle tillåta den att fylla flera uppgifter och därmed få ett bredare användningsområde.

5.1 Hantering av användarrättigheter

Som tidigare har berörts i denna rapport finns den bakomliggande funktionaliteten för detta redan färdig, dock har denna funktionalitet inte integrerats i applikationen.

Att integrera denna funktionalitet skulle kräva en viss mängd arbete med användargränssnittet och applikationens kärna. Detta skulle öppna upp möjligheter till en mängd vidare funktionalitet, såsom tillgång till kartor på användarbasis och webbaserad administration av kartdata, något som krävs om applikationen skall vara praktiskt användbar vid större mängder kartdata.

5.2 Stöd för flera bildformat

Stöd för vektorbaserade bildformat skulle i stor grad bidra till applikationens flexibilitet från användarsynpunkt. Dessa filformat skulle, genom sin vektorbaserade natur, tillåta fri panorering och förlustfri zoomning i kartbilder.

Om detta skulle kombineras med teknologier för mera responsiv användarinteraktion skulle detta resultera i en mera intressant användarupplevelse. Detta skulle troligen kräva att kraven på funktionalitet även på t.ex. mobiltelefoner frångås.

Detta skulle kräva ingrepp i användargränsnittet och troligen omfattande ändringar i karthanteringsdelen. En ytterligare faktor som kräver undersökande gällande detta är huruvida existerande implementationer av funktionalitet för att hantera dessa bildformat existerar, och om dessa går att använda här.

Om existerande kod för detta inte skulle finnas skulle detta kräva en stor mängd arbete att implementera.

5.3 Finnande av intressepunkter

En stor del av existerande positionsbaserade applikationers funktionalitet rör hantering av specifika intressepunkter, t.ex. att finna närliggande restauranger. Stöd för detta skulle kunna implementeras. Dock finns några utmaningar gällande detta:

- Ett problem rör kategorisering av sådana punkter, en uppsättning standardkategorier med vilka intressepositioner inom ett område kan märkas upp måste tas fram. Dessa kategorier måste vara allmängiltiga nog för att täcka all vanligen förekommande fall.
- Ett mindre problem är att kartpaketsformatet måste modifieras för att tillåta att intressepunkter bifogas. Detta skulle dessutom kräva vissa modifieringar i karthanteringsdelen, användargränssnittet och datalagringsdelen.

Kapitel 6

Resultat

I detta kapitel kommer utfallet av detta projekt diskuteras i närmare detalj, med avseende på vad som har producerats, vad som saknas och hur detta arbete kunde ha utförts på ett bättre sätt.

Som har berörts tidigare i rapporten har projektet ej helt kunnat slutföras, såtillvida att en applikation uppfyllande de ställda kraven ej i skrivande stund har kunnat levereras.

6.1 Vad har producerats

De delar av applikationen som kan anses slutförda är följande:

- *Karthantering:* Delen för karthantering kan anses vara färdig. Funktionalitet för att ladda lagrade kartor från ett lokalt filsystem finns på plats. Vidare finns kod för att söka bland lagrade kartor utifrån en given koordinat, samt kod för att markera en position på en given kartbild.

- *Användarhantering:* Denna del tillhandahåller möjlighet att kunna lägga till, söka efter och kontrollera en användares lösenord vid inloggning. Denna del har testats, och design av en bakomliggande databas har genomförts och testats.

- *Koordinatinhämtning:* Kod för att läsa positionsdata från en NMEA 0183-kompatibel GPS-mottagare har skrivits och testats med gott resultat. Dock har denna kod ännu ej integrerats i en applet för användning för koordinatinläsning.

- *Hantering av UAProf:* Som tidigare nämnts har denna del visat sig orsaka de största problemen. Dock har en betydande inläsning gällande detta skett, vilket gör att en egen implementation av hantering för detta nu är genomförbar.

- *Datalagring:* Ett väl fungerande format för lagring och distribution av kartor har skapats. Ett förslag till en databas, lämplig för lagring av kartor, och relaterad information har skapats och testats.

- *Servlet och användargränssnitt:* Dessa delar är planerade, men implementationen har ej i skrivande stund kunnat slutföras.

Som synes är majoriteten av applikationens bakomliggande funktionalitet klar, medan sammanställandet av dessa delar till en fungerande applikation fortfarande återstår.

6.2 Återstående arbete

För att applikationen skall anses färdig enligt de krav som ställdes upp i början av arbetet krävs att följande arbete utförs:

- *Implementation av parser för UAProf:* Denna parser krävs för att kunna välja lämpliga bildformat och bildstorlekar för en mobil klient. På grund av bristen på lämpliga färdiga tillämpningar kommer en egen implementation att behöva skrivas.

- *Inläsning av koordinater:* Som tidigare har nämnts finns kod klar för att hantera inläsning och parsning av NMEA-meddelanden, dock krävs ytterligare arbete för att integrera denna kod i en applet.

- *Applikation och gränssnitt:* Dessa delar måste slutföras. Då dessa delar är slutföra kan applikationen sättas samman för att testas i sin helhet.

- *Dokumentation och distribution:* Applikationen skall packas till ett format lämpligt för distribution, och förses med en enklare användar- och installationshandledning.

6.3 Problem och lärdomar

I denna sektion kommer de problem diskuteras, vilka fick till följd att applikationen ej i skrivande stund är klar, samt vilka lärdomar som kan dras av detta för att undvika liknande misslyckanden i framtiden.

En viktig anledning, som mycket väl kan vara den huvudsakliga anledningen till försening är brist på tillräckligt pessimistisk planering. Då arbetet påbörjades planerades en mängd extrafunktionalitet i syfte att göra applikationen intressantare. Då implementationsarbetet påbörjades ansågs det finnas gott om tid att implementera applikationens kärnfunktionalitet, varför extrafunktionalitet togs med i utvecklingsarbetet redan från början.

Detta medförde att tid spilldes på icke-central funktionalitet, tid som hade behövts för att arbeta på applikationens grundläggande funktionalitet. Följden av detta är att såväl extra såsom grundläggande funktionalitet befinner sig i ett icke färdigt stadium.

Vidare orsakades andra förseningar av en något naiv tillit till utomstående personers kod. Som tidigare nämnts finns tredjepartsbibliotek för att hantera UAProf[10] tillgänglig, vilken inte visade sig vara av den kvalitet som hade önskats.

Det verkliga problemet som uppstod här orsakades av att det faktum att en existerande tredjepartsapplikation antogs fungera på ett tillfredsställande sätt, vilket medförde att tid lades ned på annan verksamhet, då den borde ha lagts ned på utvärdering av tredjepartsmjukvaran i syfte att säkerställa dess funktionalitet.

En tredje faktor är en brist på perspektiv som upplevs då arbete med ett specifikt problem pågår. Det är i detta fall lätt hänt att stora mängder arbetstid går förlorad då

programmeraren är fokuserad på en mycket liten del, och därmed tappar det stora perspektivet.

Den sista bidragande faktorn är att den resurs som handledaren/uppdragsgivaren utgör borde ha utnyttjats på ett bättre sätt. Problem borde ha förts fram till denne oftare, för att kunna dra nytta av dennes ytterligare kompetens, och för att få ett annat perspektiv på möjliga problem/lösningar.

Detta kan sammanfattas i tre punkter, vilkas efterlevande hade tillåtit ett mera lyckat slutresultat:

- *Planera pessimistiskt:* Gör en bedömning av den tiden som anses krävas för att slutföra uppgiften, och räkna sedan upp den tiden ytterligare, för att kompensera för problem som *kommer* att uppstå. Börja i liten skala, bygg en komplett applikation med den minimala funktionaliteten först, för att först efter det arbeta sig vidare med extrafunktionalitet. Att, som försöktes i detta projekt, bygga stora delar av applikationen parallellt för att sist av allt sättas samman kräver att planeringen är mycket exakt, och att inga komplikationer tillstöter.

- *Var misstänksam mot färdig kod:* Lita aldrig blint till att kod från tredje part för att lösa en uppgift verkligen utgör en lösning. Tredjepartsmjukvara bör testas och utvärderas noggrant *innan* den planeras in i applikationen och betraktas som lösning till en uppgift.

- *Förlora inte helhetsperspektivet:* Som tidigare nämnts kan överdriven fokusering på ett problem leda till brist på helhetssyn och förlorande av tidsperspektivet. Det kan vara mera tidseffektivt att försöka finna ett sätt att kringgå ett problem än att försöka lösa det.

Litteraturförteckning

[1] *ENIRO,*
http://www.eniro.se/, besökt 050418

[2] *TELIASONERA,*
http://www.teliasonera.se/, besökt 050418

[3] *TOMTOM,*
http://www.tomtom.com/, besökt 050418

[4] *GLONASS,*
http://www.glonass-center.ru/, besökt 050321

[5] *EUROPA - Energy and Transport - GALILEO,*
http://europa.eu.int/comm/dgs/energy_transport/galileo/index_en.htm,
besökt 050320

[6] *WGS84 - World Geodetic System 84,*
http://www.wgs84.com/, besökt 050405

[7] Frank Buschmann: *Pattern-oriented Software Architecture, Volym 1: System of Patterns*
Wiley, 1996

[8] *Resource Description Framework (RDF),*
http://www.w3.org/RDF/, besökt 050423

[9] *Composite Capabilities/Preferences Profile Public Home Page,*
http://www.w3.org/Mobile/CCPP/, besökt 050423

[10] *User Agent Profiling Specification,*
http://www.openmobilealliance.org/tech/affiliates/wap/wap-248-uaprof-20011020-a.pdf, besökt 050503

[11] Marty Hall, Larry Brown: *Core Servlets And Javaserver Pages, andra utgåvan.*
Pearson Professional Education, 2003

[12] *javax.comm*
http://java.sun.com/products/javacomm/index.jsp, besökt 050423

Bilaga A

Formatspecifikation för mmd.xml

```
<!ELEMENT package (vendor, description, date, map+)>
<!ATTLIST package name CDATA #REQUIRED>
<!ELEMENT vendor (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT map (filename, type, topleftcoord, bottomrightcoord)>
<!ATTLIST map name CDATA #REQUIRED>
<!ELEMENT filename (#PCDATA)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT topleftcoord (longitude, latitude)>
<!ELEMENT bottomrightcoord (longitude, latitude)>
<!ELEMENT longitude EMPTY>
<!ATTLIST longitude
        value CDATA #REQUIRED
        eorw (E|W) #REQUIRED
>
<!ELEMENT latitude EMPTY>
```

```
<!ATTLIST latitude
        value CDATA #REQUIRED
        nors (N|S) #REQUIRED
>
```

Bilaga B

CC/PP-profil SonyEricsson Z600

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:mms="http://www.wapforum.org/profiles/MMS/ccppschema-20010111#"
         xmlns:prf="http://www.openmobilealliance.org/tech/profiles/ \
UAPROF/ccppschema-20030226#">

  <rdf:Description rdf:ID="Profile">

    <prf:component>
      <rdf:Description rdf:ID="HardwarePlatform">
      <rdf:type rdf:resource="http://www.openmobilealliance.org/tech/profiles/ \
       UAPROF/ccppschema-20030226#HardwarePlatform"/>
        <prf:BitsPerPixel>16</prf:BitsPerPixel>
        <prf:ColorCapable>Yes</prf:ColorCapable>
        <prf:ScreenSize>128x160</prf:ScreenSize>
        <prf:ImageCapable>Yes</prf:ImageCapable>
        <prf:InputCharSet>
```

```
      <rdf:Bag>
        <rdf:li>ISO-8859-1</rdf:li>
        <rdf:li>US-ASCII</rdf:li>
        <rdf:li>UTF-8</rdf:li>
        <rdf:li>ISO-10646-UCS-2</rdf:li>
      </rdf:Bag>
    </prf:InputCharSet>
    <prf:Keyboard>PhoneKeypad</prf:Keyboard>
    <prf:Model>Z600R601</prf:Model>
    <prf:NumberOfSoftKeys>2</prf:NumberOfSoftKeys>
    <prf:OutputCharSet>
      <rdf:Bag>
        <rdf:li>ISO-8859-1</rdf:li>
        <rdf:li>US-ASCII</rdf:li>
        <rdf:li>UTF-8</rdf:li>
        <rdf:li>ISO-10646-UCS-2</rdf:li>
      </rdf:Bag>
    </prf:OutputCharSet>
    <prf:PixelAspectRatio>1x1</prf:PixelAspectRatio>
    <prf:ScreenSizeChar>15x8</prf:ScreenSizeChar>
    <prf:StandardFontProportional>Yes</prf:StandardFontProportional>
    <prf:SoundOutputCapable>Yes</prf:SoundOutputCapable>
    <prf:TextInputCapable>Yes</prf:TextInputCapable>
    <prf:Vendor>Sony Ericsson Mobile Communications</prf:Vendor>
    <prf:VoiceInputCapable>Yes</prf:VoiceInputCapable>
  </rdf:Description>
</prf:component>
```

```
<prf:component>
  <rdf:Description rdf:ID="SoftwarePlatform">
  <rdf:type rdf:resource="http://www.openmobilealliance.org/tech/profiles/ \
   UAPROF/ccppschema-20030226#SoftwarePlatform"/>
    <prf:AcceptDownloadableSoftware>No</prf:AcceptDownloadableSoftware>
    <prf:CcppAccept>
      <rdf:Bag>
        <rdf:li>image/gif</rdf:li>
        <rdf:li>image/jpeg</rdf:li>
        <rdf:li>image/vnd.wap.wbmp</rdf:li>
        <rdf:li>image/bmp</rdf:li>
        <rdf:li>image/png</rdf:li>

        <rdf:li>text/x-iMelody</rdf:li>
        <rdf:li>text/x-vMel</rdf:li>
        <rdf:li>text/x-eMelody</rdf:li>

        <rdf:li>audio/amr</rdf:li>
        <rdf:li>audio/midi</rdf:li>
        <rdf:li>audio/x-midi</rdf:li>
        <rdf:li>audio/mid</rdf:li>
        <rdf:li>audio/iMelody</rdf:li>

        <rdf:li>text/x-vCard</rdf:li>
        <rdf:li>text/x-vCalendar</rdf:li>
        <rdf:li>text/x-vNote</rdf:li>
```

```
<rdf:li>application/java</rdf:li>
<rdf:li>application/java-archive</rdf:li>
<rdf:li>text/vnd.sun.j2me.app-descriptor</rdf:li>

<rdf:li>application/vnd.eri.thm</rdf:li>
<rdf:li>application/vnd.mophun.application</rdf:li>
<rdf:li>application/vnd.mophun.certificate</rdf:li>
<rdf:li>application/xhtml+xml</rdf:li>
<rdf:li>text/css</rdf:li>
<rdf:li>text/html</rdf:li>
<rdf:li>application/vnd.oma.drm.message</rdf:li>
<rdf:li>application/vnd.oma.dd+xml</rdf:li>
<rdf:li>application/vnd.sonyericsson.mms-template</rdf:li>

<rdf:li>application/vnd.wap.wmlc</rdf:li>
<rdf:li>application/vnd.wap.wbxml</rdf:li>
<rdf:li>application/vnd.wap.wmlscriptc</rdf:li>
<rdf:li>application/vnd.wap.multipart.mixed</rdf:li>
<rdf:li>application/vnd.wap.sic</rdf:li>
<rdf:li>application/vnd.wap.slc</rdf:li>
<rdf:li>application/vnd.wap.coc</rdf:li>
<rdf:li>application/vnd.wap.sia</rdf:li>
<rdf:li>application/vnd.wap.wtls-ca-certificate</rdf:li>
<rdf:li>application/vnd.wap.xhtml+xml</rdf:li>
<rdf:li>application/x-wap-prov.browser-settings</rdf:li>
<rdf:li>application/vnd.wap.connectivity-wbxml</rdf:li>
```

```
  </rdf:Bag>
</prf:CcppAccept>
<prf:CcppAccept-Charset>
  <rdf:Bag>
    <rdf:li>US-ASCII</rdf:li>
    <rdf:li>ISO-8859-1</rdf:li>
    <rdf:li>UTF-8</rdf:li>
    <rdf:li>ISO-10646-UCS-2</rdf:li>
  </rdf:Bag>
</prf:CcppAccept-Charset>
<prf:CcppAccept-Encoding>
  <rdf:Bag>
    <rdf:li>base64</rdf:li>
  </rdf:Bag>
</prf:CcppAccept-Encoding>
<prf:JavaEnabled>Yes</prf:JavaEnabled>
<prf:JavaPlatform>
  <rdf:Bag>
    <rdf:li>MIDP</rdf:li>
    <rdf:li>CLDC</rdf:li>
  </rdf:Bag>
</prf:JavaPlatform>
<prf:JavaPackage>
  <rdf:Bag>
    <rdf:li>jsr-135</rdf:li>
  </rdf:Bag>
</prf:JavaPackage>
```

```
  </rdf:Description>
</prf:component>

<prf:component>
  <rdf:Description rdf:ID="NetworkCharacteristics">
  <rdf:type rdf:resource="http://www.openmobilealliance.org/tech/profiles/ \
   UAPROF/ccppschema-20030226#NetworkCharacteristics"/>
    <prf:SecuritySupport>
      <rdf:Bag>
        <rdf:li>WTLS-1</rdf:li>
        <rdf:li>WTLS-2</rdf:li>
        <rdf:li>WTLS-3</rdf:li>
        <rdf:li>signText</rdf:li>
      </rdf:Bag>
    </prf:SecuritySupport>
    <prf:SupportedBearers>
      <rdf:Bag>
        <rdf:li>GSM_GPRS_IPV4</rdf:li>
        <rdf:li>GSM_CSD_IPV4</rdf:li>
      </rdf:Bag>
    </prf:SupportedBearers>
  </rdf:Description>
</prf:component>

<prf:component>
  <rdf:Description rdf:ID="BrowserUA">
  <rdf:type rdf:resource="http://www.openmobilealliance.org/tech/profiles/ \
```

```
   UAPROF/ccppschema-20030226#BrowserUA"/>
    <prf:BrowserName>Sony Ericsson</prf:BrowserName>
    <prf:FramesCapable>No</prf:FramesCapable>
    <prf:PreferenceForFrames>No</prf:PreferenceForFrames>
    <prf:TablesCapable>Yes</prf:TablesCapable>
  </rdf:Description>
</prf:component>

<prf:component>
  <rdf:Description rdf:ID="WapCharacteristics">
  <rdf:type rdf:resource="http://www.openmobilealliance.org/tech/profiles/ \
   UAPROF/ccppschema-20030226#WapCharacteristics"/>
    <prf:WapDeviceClass>C</prf:WapDeviceClass>
    <prf:WapVersion>1.2.1</prf:WapVersion>
    <prf:WmlDeckSize>10000</prf:WmlDeckSize>
    <prf:WmlScriptLibraries>
      <rdf:Bag>
        <rdf:li>Lang</rdf:li>
        <rdf:li>Float</rdf:li>
        <rdf:li>String</rdf:li>
        <rdf:li>URL</rdf:li>
        <rdf:li>WMLBrowser</rdf:li>
        <rdf:li>Dialogs</rdf:li>
      </rdf:Bag>
    </prf:WmlScriptLibraries>
    <prf:WmlScriptVersion>
      <rdf:Bag>
```

```
            <rdf:li>1.0</rdf:li>
          </rdf:Bag>
        </prf:WmlScriptVersion>
        <prf:WmlVersion>
          <rdf:Bag>
            <rdf:li>1.1</rdf:li>
            <rdf:li>1.2</rdf:li>
            <rdf:li>1.3</rdf:li>
          </rdf:Bag>
        </prf:WmlVersion>
        <prf:WtaiLibraries>
          <rdf:Bag>
            <rdf:li>WTAPDC</rdf:li>
          </rdf:Bag>
        </prf:WtaiLibraries>
        <prf:DrmClass>
          <rdf:Bag>
            <rdf:li>ForwardLock</rdf:li>
          </rdf:Bag>
        </prf:DrmClass>
<prf:OmaDownload>Yes</prf:OmaDownload>
      </rdf:Description>
    </prf:component>

    <prf:component>
      <rdf:Description rdf:ID="PushCharacteristics">
      <rdf:type rdf:resource="http://www.openmobilealliance.org/tech/profiles/ \
```

```
   UAPROF/ccppschema-20030226#PushCharacteristics"/>
    <prf:Push-Accept-AppID>
      <rdf:Bag>
        <rdf:li>x-wap-application:wml.ua</rdf:li>
      </rdf:Bag>
    </prf:Push-Accept-AppID>
    <prf:Push-MsgSize>1500</prf:Push-MsgSize>
    <prf:Push-MaxPushReq>1</prf:Push-MaxPushReq>
  </rdf:Description>
</prf:component>

<prf:component>
  <rdf:Description rdf:ID="MMSCharacteristics">
  <rdf:type rdf:resource="http://www.wapforum.org/profiles/\
   MMS/ccppschema-20010111#MmsCharacteristics" />
    <mms:MmsMaxMessageSize>261120</mms:MmsMaxMessageSize>
    <mms:MmsMaxImageResolution>120x160</mms:MmsMaxImageResolution>
    <mms:MmsCcppAccept>
      <rdf:Bag>
        <rdf:li>image/jpeg</rdf:li>
        <rdf:li>image/gif</rdf:li>
        <rdf:li>image/vnd.wap.wbmp</rdf:li>
        <rdf:li>image/bmp</rdf:li>
        <rdf:li>image/png</rdf:li>

        <rdf:li>audio/amr</rdf:li>
        <rdf:li>audio/x-amr</rdf:li>
```

```
<rdf:li>audio/midi</rdf:li>
<rdf:li>audio/x-midi</rdf:li>
<rdf:li>audio/mid</rdf:li>
<rdf:li>text/x-iMelody</rdf:li>
<rdf:li>audio/iMelody</rdf:li>

<rdf:li>text/x-vCard</rdf:li>
<rdf:li>text/x-vCalendar</rdf:li>
<rdf:li>text/x-vNote</rdf:li>

<rdf:li>application/java-archive</rdf:li>
<rdf:li>application/vnd.mophun.application</rdf:li>
<rdf:li>application/vnd.mophun.certificate</rdf:li>

<rdf:li>text/plain</rdf:li>
<rdf:li>application/smil</rdf:li>
<rdf:li>application/x-sms</rdf:li>
<rdf:li>application/vnd.3gpp.sms</rdf:li>
<rdf:li>application/vnd.eri.thm</rdf:li>
<rdf:li>application/vnd.sem.mms.protected</rdf:li>

<rdf:li>application/vnd.wap.mms-message</rdf:li>
<rdf:li>application/vnd.wap.multipart.mixed</rdf:li>
<rdf:li>application/vnd.wap.multipart.alternative</rdf:li>
<rdf:li>application/vnd.wap.multipart.related</rdf:li>

<rdf:li>application/vnd.oma.drm.message</rdf:li>
```

```
          </rdf:Bag>
        </mms:MmsCcppAccept>
        <mms:MmsCcppAcceptCharSet>
          <rdf:Bag>
            <rdf:li>US-ASCII</rdf:li>
            <rdf:li>ISO-8859-1</rdf:li>
            <rdf:li>ISO-8859-2</rdf:li>
            <rdf:li>UTF-7</rdf:li>
            <rdf:li>UTF-8</rdf:li>
            <rdf:li>UTF-16</rdf:li>
            <rdf:li>KOI8-R</rdf:li>
            <rdf:li>windows-1251</rdf:li>
            <rdf:li>ISO-10646-UCS-2</rdf:li>
          </rdf:Bag>
        </mms:MmsCcppAcceptCharSet>
        <mms:MmsVersion>
          <rdf:Bag>
            <rdf:li>1.0</rdf:li>
          </rdf:Bag>
        </mms:MmsVersion>
      </rdf:Description>
    </prf:component>
  </rdf:Description>
</rdf:RDF>
```