Utveckling av tjänster för ett mobilt medium

Daniel Tyseng

Institutionen för Data- och Systemvetenskap,
Stockholms Universitet
April 2004

Abstract[1]

The mobile revolution is here to stay, and today most people have (at least) one cell phone at home. Some even a PDA (Personal Digital Assistant) as a complement for the simpler tasks at the office, e.g sending email, word processing, browsing the Internet etc. Many businesses have also given the employees the opportunity to do some of the business tasks inbetween customer visits with the help from a cell phone or a PDA.

Despite all this, the so called Internet connected mobile revolution is often mentioned as a big failure. Maybe because the people in general tended not to use the new technology as much as the companies predicted. This could be because they still haven't learned how it works, or just don't seem to have the need for the services provided.

After a couple of failures in the development process, almost every cell phone has support for mobile Internet today. Instead the responsibility lies on the content providers and cell phone operators. The users are going to need killer applications, e.g. services as email, chat, time tables, services they feel they can't be without. How is a service like this created then? And what kind of obstacles lies in the way of making this service available for everyone that uses a cell phone – not only the technical ones? What kind of responsibility do the operators have?

To change from the Internet based platform to the mobile platform could contain some difficult tasks. Even if some technical issues occured during the projekt, the main challenge is not the technical one. In fact, the technical issues should not be a problem for companies already working with Internet technology.

[1] Uppsatsen motsvarar 20 poäng.

The challenge would instead be how to implement the architecture, safety, usability, costs and to understand the real need for the system. It is not only about building up a new system, but also about buying new equipment such as new hardware, cell phones, PDA:s and also to educate the employees. Even safety issues might have to be taken care of, such as cryptology.

Sammanfattning

Den mobila revolution har kommit för att stanna, och idag har de flesta (minst) en mobiltelefon hemma, många även en s.k. PDA (Personal Digital Assistant) som komplement för att sköta enklare kontorssysslor såsom att skicka email, nå Internet etc. Många företag har dessutom anammat tekniken genom att förse sina anställda med Internetuppkopplade mobiltelefoner eller PDA's för att kunna sköta fler av de vanligaste sysslorna mellan kundbesök eller till och från jobbet.

Trots detta talas det ofta om den Internetuppkopplade mobilrevolutionen som ett fiasko. Kanske för att den stora massan mobiltelefonanvändare fortfarande inte lärt sig, eller inte känner sig ha behov av, användandet av mobiltelefonen som ett nytt medium att nå Internet med.

Efter en rad mer eller mindre misslyckade försök har idag i princip alla nya telefon-modeller på marknaden stöd för mobilt surfande. Nu ligger ansvaret hos tjänste-leverantörerna. Användarna vill ha "killer applications", tjänster som email, chat, tidtabeller etc, som de känner att de inte kan klara sig utan. Hur skapas då en sådan tjänst? Och vilka hinder ligger i vägen för att lyckas snärja den stora massan användare? Vad har operatörerna för ansvar?

Övergången från en Internetbaserad plattform till den mobila plattformen är inte helt enkel. Även om ett par tekniska svårigheter dök upp under projektets gång, är den stora utmaningen dock inte av teknisk karaktär. För företag som arbetar med Internetbaserade lösningar eller för ansvariga för Intranät, Internet etc. på ett företag torde den rent tekniska övergången ske utan betydande problem.

Utmaningen torde istället ligga i frågor som rör arkitektur, affärsnytta, säkerhet, användarvänlighet och kostnad. Det handlar nämligen inte enbart om att bygga upp ett system utan om eventuell införskaffning av ny hårdvara, nya mobiltelefoner, PDA:s till personalen som berörs av övergången samt utbildning. Dessutom, om det rör affärskritiska lösningar som hanterar viktig information, även säkerhet med kryptering.

1 Inledning

1.1 Bakgrund

Behovet av kommunikationsutrustning för människor på resande fot ledde till utvecklingen av mobiltelefonen. Även om möjligheten att kunna samtala fortfarande är det största användningsområdet, växer behovet efter mobila datatjänster, såsom möjligheten att nå Internet eller att kunna läsa sina email via sin telefon. Utvecklingen har gått minst sagt trögt, men trots detta ökar försäljningen av mobiltelefoner och s.k. PDA´s, med vilka man kan nå Internet.

Många operatörer har tagit fasta på detta och erbjuder numera sina kunder hela paket med tjänster, såsom möjligheten att surfa på Internet, få nyheter, sport eller börskurser till sin telefon.

Många företag erbjuder dessutom numer sina anställda visa företagsanknutna tjänster via mobiltelefon eller PDA. Det kan tex vara tillgång till email eller intranätet, eller mer sofistikerade tjänster, men som sparar tid och pengar åt företaget.

Tekniken har funnits länge. Både mobiltelefoner och PDA som stöder denna typ av tjänster samt serverprogramvara och utvecklarverktyg finns i överflöd på marknaden. Nu verkar bara en rent generell attitydförändring mot det mobila mediet behvöas för att utvecklingen ska ta en ny vändning.

1.2 Syfte och frågeställning

Syftet med detta examensarbete är att förklara tekniken bakom mobilt Internet, hur ett företag kan integrera sina existerande Internettillämpningar till den nya generationens mobiltelefoner – och därmed göra sina tjänster tillgängliga för människor på resande fot.

Uppsatsen kommer även att studera bakgrunden till mobilt Internet (dess historia), samt försöka förutspå åtminstone delar av dess framtid. Vilka tillämpningar kan med störst sannolikhet nå den stora massan användare? Vad krävs av operatörer, telefontillverkare samt tjänsteleverantörer för att så ska bli fallet?

Slutligen kommer en prototyp av en typisk Internettillämpning tillgänglig för ett mobilt medium skapas och i takt med att arbetet fortlöper även besvara några viktiga frågeställningar:

- Vilka för- och nackdelar samt problem finns det med de tekniker som använts? Vilka lösningar finns för dessa?
- Vilken utvecklingsmiljö, vilka verktyg och vilka metoder är lämpliga för konstruerandet av en mobil tjänst?
- Vilka typiska webbtjänster lämpar sig för mobilt Internet?

1.3 Avgränsning

Uppsatsen väljer att studera de ”mjuka” delarna av mobilt Internet, såsom mjukvara, protokoll och tjänster. Även om viss beskrivning kommer att ges om bakomliggande hårdvaruarkitektur, så är det en ytterst liten del av detta arbete. Det är ingenting som täcks närmare inom ramen av detta examensarbete.
Uppsatsen kommer heller inte ta någon hänsyn till de typer av applikationer som programmeras direkt mot den handburna enheter, tex små program eller applets som installeras i enheten och sedan kan komma åt tjänster via Internet. I stället har all vikt lagts vid applikationer som ”browsas” fram, alltså går via den handburna enhetens inbyggda microbrowser.

1.4 Metod

Eftersom mobilt Internet är ett ämne som ständigt växer - och förändras - är Internet det perfekta mediet att uppsöka information för en sådan här uppsats. En webbsida har den fördelen att den snabbt kan uppdateras och informationen därmed alltid kan hållas färsk. Man måste dock vara källkritisk vid sökandet av information på Internet. Vem har skrivit informationen? Vad är sant? De källor som använts är främst från seriösa institut, såsom stora branschtidningar, utvecklare samt universitet.

Det har dock framkommit mycket matnyttig information i litteraturen, något som legat till grund för beskrivning av olika arkitekturer etc.

Slutligen har, genom eget experimenterande, olika uppfattningar bildats om vilka tekniker etc. som passar bäst för diverse lösningar. Detta är främst något som använts vid byggandet av den prototyp som visas längre fram i uppsatsen.

Uppsatsen är en blandning av en ren litteraturstudie och en experimentstudie. Den är skriven på ett väldigt iterativt sätt med växlande programmering av prototypen och dokumenterande av de frågeställningar som då besvarats.

1.5 Målgrupp

Denna uppsats riktar sig till systemvetare, civilingenjörer inom datavetenskap eller andra som kan tänkas ha liknande förkunskaper för att kunna ta till sig innehållet. Läsaren förväntas ha grundläggande programmeringskunskaper (dock inte i något specifikt programmeringsspråk) samt kunskaper om Internet och dess uppbyggnad. Det förekommer även en hel del begrepp som kan verka nya och svårbegripliga för läsaren. De flesta av dessa har därför sammanfattats i en ordlista som infinner sig som ett appendix sist i uppsatsen.

2 Mobila Tjänster

2.1 Mobilitet

Med hjälp av den nya tekniken som mobiltelefonen tillhandahåller, skapas möjligheter till kommunikation oberoende av om personen är på resande fot och kanske inte har tillgång till en dator eller telefon. Tekniken skapar förutsättningar för människor att kommunicera utan att egentligen mötas i det verkliga livet. Detta kan leda till stora kostnadsfördelar både för privatpersoner och för företag som kan spara in pengar på till exempel rese-kostnader. Kan en anställd dessutom göra en del av det dagliga arbetet från och till sin arbetsplats kan detta leda till ökad effektivitet och minskad stress. På många företag krävs också att anställda arbetar ute på fältet och träffar kunder. För dessa kan det innebära en stor fördel att kunna sköta några av de dagliga kontorssysslorna från och till dessa möten.

Även bland privatpersoner ökar mobiltelefonanvändandet, och dagens ungdomar är väl förtrogna med den nya tekniken. Ungefär som äldre generationer växte upp med radio och TV har dessa vuxit upp med mobiltelefonen, och den är helt enkelt en del av deras privata accessoarer.

2.2 Mobila Nätverk

De mobila nätverken som utvecklades på 1970- och 1980-talet byggde oftast på analoga nät som stödde vanlig röstkommunikation. Senare utvecklades digitala mobilnätverk, i ibland kallat 2G – eller andra generationens mobiltelefoni. Dessa nät baserades inte på någon internationell standard, utan regionala standarder som utvecklades i Europa samt Nordamerika (och adopterades av resten av världen). Den standard som antogs i Sverige var GSM (Global System for Mobile Communications) och klarade precis som de Nordamerikanska standarderna (t.ex. IS 136 och IS 95A) dataöverföringar på maximalt 14.4 Kbps. Applikationer skapade för denna låga överföringshastighet kunde endast stödja text samt lågupplöst grafik. (DreamTeach, 2002)

Andra generationens mobiltelefoni utvecklades senare till 2.5G vilket kan stödja data-överföringar på mellan 64 och 144 Kbps. Ett exempel på detta är GPRS (General Packet Radio Service) som fungerar på GSM-nätet. Lite kort beskrivet kan sägas att GPRS gör det möjligt för GSM:s kretskopplade nät att arbeta paketförmedlat, alltså samma teknik som används på Internet. GPRS ersätter alltså inte GSM utan är ett tillägg till GSM-näten. En stor fördel med GPRS jämför med GSM är dock möjligheten för kunden att betala för den mängd data han laddar ner istället för den tid han är uppkopplad. Detta ger lägre kostnader då användaren inte behöver skicka/ta emot mycket data, men samtidigt vill vara uppkopplad länge, passande för olika tidningstjänster bland annat. (Ewert, 2001)

2.5G nätverken är nu på väg att utvecklas till 3G, tredje generationens mobilnät, vilket kommer att stödja dataöverföringar på mellan 384 och 2048 Kbps. Med dessa hastigheter kommer applikationer som videokonference, nedladdning av musik och högupplösta bilder, animationer etc. att stödjas. (DreamTech, 2002) I skrivande stund har de enda 3G-operatörerna i Sverige (*tre* och *telia)* endast ett fåtal abonnenter trots massiva

reklamkampanjer, så utvecklingen går minst sagt trögt. I andra länder som Japan, där enbart Mobiloperatören NTT Docomo har över en miljon användare och ett nät som nås av 96 procent av dess befolkning ser framtidsutsikterna bättre ut. (CS 6/10-03)

2.3 Mobilt Internet

För människor på resande fot har röstkommunikation än så länge varit det största användningsområdet. På senare år har dock en rad olika tjänster dykt upp på Internet, vilka även kan nås via handburna enheter. Trots detta tar inte användningen fart i den takt som operatörer, telefontillverkare och tjänstetillverkare önskar. Den vanligaste mobila enheten, med vilken tillgång till Internettjänster kan fås, är utan tvekan mobiltelefonen. Denna har som bekant även möjligheten till tal – och därmed en apparat som de flesta har.

Mobilt Internet kan även fås genom en s.k. PDA (Personal Digital Assistant), eller handdator i dagligt tal. Dessa har en större skärm än dagens mobiltelefoner och är därmed lite bättre lämpade för detta ändamål. Dessutom kommer de med ett annat grafiskt användargränssnitt som ofta påminner om det som finns på dagens persondatorer, vilket gör det lättare att hantera. Vissa av de PDA:s som säljs idag har inbyggd mobiltelefon, vilket gör dem till ett ypperligt redskap för mobilt surfande eftersom ingen extern telefon eller kort behöver anslutas.

Ytterligare ett sätt att nå Internet på resande fot är att ansluta en bärbar dator till sin mobiltelefon eller genom att ansluta ett kort för själva datatrafiken.

2.4 Mobila Tjänster

En mobil webbtjänst eller applikation skiljer sig på många sätt från en vanlig webbtjänst. Den har till exempel mycket högre grad av tillgänglighet än en dator eftersom det är något som de flesta bär med sig överallt. Samtidigt kan den endast erbjuda en reducerad funktionalitet jämfört med en vanlig tjänst. Det som skiljer dessa olika tjänster åt beror bland annat på : (White paper, Ericsson 2002)

- Mindre skärmstorlek
- Mindre processorkraft
- Reducerad bandbredd
- Mindre RAM
- Svårare navigering

Detta gör att utvecklaren av en mobil tjänst måste reducera mycket av funktionalitet och gränssnitt för att den ska kunna passa en mobil enhet. Användaren kommer till exempel använda sig av helt andra metoder för att navigera än vid en vanlig dator. Istället för att använda en mus kommer denne enbart ha tillgång till ett antal tätt lokaliserade knappar. Helst ska han kunna sköta hela navigeringen med en hand.

Vid design av en tjänst för ett mobilt medium är enkelheten nyckeln till acceptans och uppskattning hos kunderna eftersom användaren ofta behöver koncentrera sig på andra saker samtidigt. I Design Guidelines som är ett dokument för utvecklare av WAP-tjänster och ges ut av Ericsson ges några riktlinjer för hur utformningen av en tjänst bör se ut:

Informationen bör vara kort och meningsfull. Listor och menyer bör vara tämligen korta; Användaren ska aldrig behöva skrolla mer än 4-5 skärmbilder samtidigt (16-20 rader). Långa ord och kryptiska förkortningar bör också undvikas till förmån för korta och lättförståeliga ord som inte kan misstolkas. Det finns dessutom en gräns för hur många bytes som faktiskt kan sändas till en enhet. (Detta är något som gäller alla telefoner och dessa storlekar varierar ganska kraftigt, vid utveckling av en tjänst bör denna därför testas på en rad olika telefoner). (Design Guides for r320, Ericsson)

Det kanske viktigaste vid design av en mobil tjänst som bygger på Internetteknologin är dock begränsningen av grafik. Då användaren måste betala mer för en sida med många bilder och att sidan tar längre tid att ladda bör grafiken reduceras till ett minimum.

I Ericssons White Paper ”How to make a web site mobile – Extending web services with a successful mobile service” delas olika tjänster in i grupperna *Here*, *Now*, *Fun* och *Cool* beroende på innehåll och vilka tänkta målgrupper som är intresserade av tjänsten.

Here

Mobila tjänster som tillfredställer en människas behov som uppkommer i ett mobilt sammanhang på den exakta plats användaren befinner sig – där inget annat dugligt alternativ existerar.

Tillgång från användarens plats till:
Information
Kommunikation
Transaktioner

Inga andra attraktiva alternativ finns för att:
Det finns ingen i närheten att fråga
Det finns ingen information på plats
Användaren är obenägen att lämna sin plats för att söka efter informationen
Det är brist på andra kommunikationsmedium
Tjänsten kräver användarens geografiska plats som input (kan ske automatiskt med WAP)

Exempel
Gula sidorna
Nyheter
Lokala kartor (till exempel närmsta bankomat, bus)
Bus och tågtidtabell
Skicka meddelande

Now

Mobila tjänster som tillfredställer en människas behov för en realtidstjänster eller tjänster relaterade till den exakta tidpunkten på dygnet.

Direkt tillgång till realtids-
Information
Kommunikation
Transaktion

Inga andra attraktiva alternativ finns för att:
Andra tjänster är för långt bort för att nås snabbt
Andra tjänster är inte online och uppdateras inte tillräckligt snabbt
Tjänsten kräver exakta tidpunkten som input (Automatiskt med WAP)

Exempel
Bus- och tågtidtabeller
Nyheter
Biljettbokning

Fun

Mobila tjänster som underhåller användaren när denne har ett par minuters strötid. ”Vänta på bussen-tjänster”.

Fun services tillåter användare att
Njuta av anonymiteten på Internet
Hålla sig uppdaterad på sådant som intresserar
Mentalt fly undan vardagen

Inga andra attraktiva alternativ finns för att:
Det finns ingen tillgänglig underhållning på plats
Det finns ingen i närheten att prata med
Användaren är obenägen att lämna sin plats i sökande efter annat nöje

Exempel
Nyheter
Interaktiva spel
Frågesporter
Tester
Nöjesinnehåll (kändisar, sport, erotik, serier)
Anonym kommunikation (chatt, blind date)

Cool

Mobiltelefonen har blivit en imagerelaterad accessoar, och erfarenheten visar att användare är beredda att betala ett högre pris för tjänster som uppfattas som tuffa inom användarens vänkrets.

Lyckosamma mobila tjänster höjer användarens ego genom att signalera:
Rikedom, Välfärd
Medlemskap i prestigefylld vänkrets eller social grupp
Tillgång till det senaste

Coola tjänster är:
Egocentriska
Lätta att kommunicera med vänner och folk i närheten.
Visuellt och ljudligt attraktiv och identifierbar
Nya
Dyra, extravaganta och exklusiva

Exempel:
Ladda upp och skicka bilder till andra
Nedladdningsbara ringsignaler, logos, bilder eller skärmsläckare
Multimedia clips

3 WAP

3.1 Historia

Tankarna om att kunna tillföra extra tjänster till mobila nätverk sträcker sig så långt tillbaka som till 1995, då Ericsson startade ett projekt för att ta fram ett koncept för just detta syfte. Projektet ledde fram till ett protokoll – ITTP, Intelligent Transfer Protocol, vilket kunde hantera kommunikationen mellan en applikationsserver och en ”intelligent” mobiltelefon.

Något år senare kom Nokia och dåvarande Unwired Planet[2] med liknande lösningar. Unwired Planet hade skapat ett koncept som hette HDML (Handheld Device Markup Language) samt HDTP (Handheld Device Transport Protocol). Dessa hade sitt ursprung ur HTML och HTTP, men var anpassade för trådlösa nät och enkla apparater.

Nokia Presenterade strax efter en teknik, med vilken man kunde nå Internet från mobiltelefonen med en teknik som byggde på SMS (Short Messages Service). Tekniken använde sig av ett språk som hette TTML (Tagged Text Markup Language).

Ett antal olika koncept hade nu på relativt kort tid skapats för ett gemensamt syfte – Att öka funktionaliteten i mobiltelefoner och handdatorer. Tillverkarna såg snart risken för att denna uppsjö olika tekniker inte skulle bli kompatibla med varandra och därmed skapa en uppdelad marknad. Detta ledde till att Ericsson, Nokia, Motorola och Unwired Planet började arbeta för en gemensam standard för att utveckla tjänster till mobilnäten och därför grundades 1997 samarbetsorganisationen Wap Forum.

Redan april 1998 hade Wap Forum presenterat sin första standard – WAP 1.0. Samtidigt öppnades WAP Forum så att alla som hade intressen i tekniken kunde ansöka om medlemskap. Idag finns ett hundratal fullvärdiga medlemmar och ungefär lika många stödmedlemmar (Ewert, 2001).

WAP 1.0 blev ingen succé och endast ett fåtal apparater stödde denna standard. De stora aktörerna väntade istället på den uppdaterade versionen 1.1, vilken kom i juni 1999.

WAP 1.2 kom i november 1999. Denna standard använder sig av ett språk som kallas WML (Wireless markup Language), ett språk som bygger på XML (eXtensible Markup Language). Språket påminner mycket om HTML, det vanligaste språket på Internet idag, men med den skillnaden att det är kraftigt nedbantat. Detta på grund av att den microbrowser som finns i mobiltelefonen har en mycket begränsad kapacitet jämfört med en browser på en vanlig dator. Standarden har även stöd för WMLScript, vilket påminner mycket om JavaScript. WMLScript utvecklades för att skapa mer interaktivitet i innehållet.

[2] Unwired Planet heter numera heter Openwave

Det fanns mycket dåligt stöd för grafik i dessa tidiga versioner av WAP och enbart bildformatet WBMP (Wireless Bitmap) stöddes. Det är en sorts lågupplöst version av Bitmap, och speciella verktyg eller plugins till bildbehandlingsprogram krävdes för att skapa dessa (DreamTeach, 2002).

En annan nackdel med WML är att allt material som redan finns tillgängligt på Internet, skrivet i HTML, måste skrivas om för att passa en mobiltelefon. Även om det skulle vara teoretiskt möjligt att skriva en applikation som automatiskt konverterar innehållet mellan de olika plattformarna skulle resultatet troligtvis inte bli bra eftersom WML bara stöder ett fåtal av HTML-taggarna.

Detta ledde till utvecklingen av ett nytt *märkspråk*. Alla märkspråk baseras på XML, precis som WML. XML kan ses som ett metaspråk, dvs. ett språk för utveckling av nya språk. Den nya standarden kom att kallas XHTML, eXtensible HTML och stöds från och med WAP 2.0. Rent funktionsmässigt påminner XHTML och HTML mycket om varandra, men på en syntaktisk nivå skiljer sig XHTML genom att det kräver användaren att vara mer försiktig med syntaxen (Sony Eriksson, Developers Guidelines – Web Browser, 2003).

XHTML Basic är en del av XHTML som används för små terminaler.

I princip alla telefonmodeller stöder numera WAP, även om relativt få modeller stöder den senaste WAP-standarden, 2.0. Denna är dock bakåtkompatibel på så sätt att den stöder både WML och XHTML.

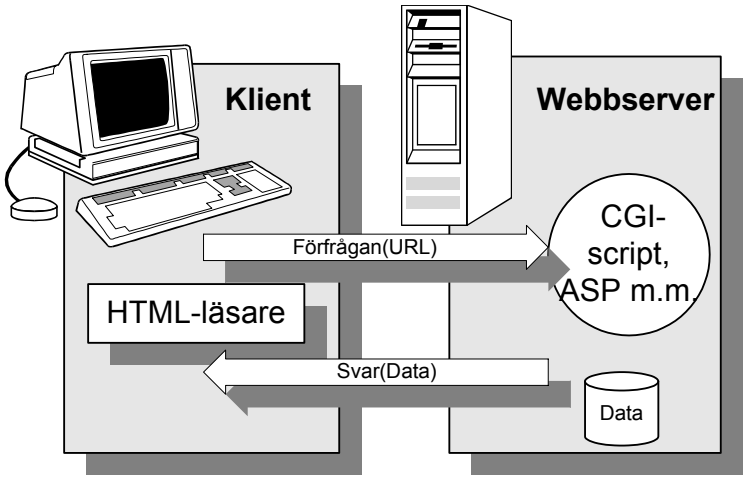
3.2 Arkitektur

Arkitekturen hos WAP påminner mycket av om dagens Internetteknik. Detta är en stor fördel för dem som tillverkar tjänster eller applikationer, bland annat för att det går snabbt att sätta sig in i den nya tekniken samt att existerande teknik kan återanvändas. Därför förklaras först Internetmodellen för att sedan gå över till WAP-modellen. (www.wapforum.org)

3.2.1 Internetmodellen

Internetmodellen fungerar så att när användaren vill surfa till en viss sida, och därmed skriver in adressen i adressfältet i en webbläsare (t.ex. Internet Explorer), skickas en förfrågan, s.k. Request, till en webbserver. Förfrågan skickas till ett direkt namngivet objekt som rent fysiskt ligger på webbservern i form av en URL, Uniform Resource Locator, t.ex. http://www.wapmail.com.

Servern kommer då att svara på förfrågan genom att skicka tillbaka den efterfrågade tjänsten, i form av standardiserad data, till webbläsaren. (Ewert, 2000)



figur 1, WWW-modellen

Inom Internetmodellen finns ett flertal mekanismer för skapandet av en så generell miljö som möjligt.

Standard Namning Model	En standardmodell för namngivning, t.ex. att alla servers och sidor på nätet är angivna med en URL
Contest typning	All data på WWW är av en standardiserad typ, vilket gör att en webbläsare kan hantera det på ett korrekt sätt [RFC2045, RFC2048]
Standard Contest format	Alla webbläsare stöder ett antal standardiserade format, som t.ex. HTML och JavaScript.
Standard Protocols	Standardiserade nätverksprotokoll gör det möjligt för en webbläsare att kommunicera med vilken server som helst. Det vanligaste protokollet på WWW är HTTP och FTP. I grunden finns TCP/IP.

Tabell 1 , Standardkomponenter i WWW (Källa: Ewert, 2000)

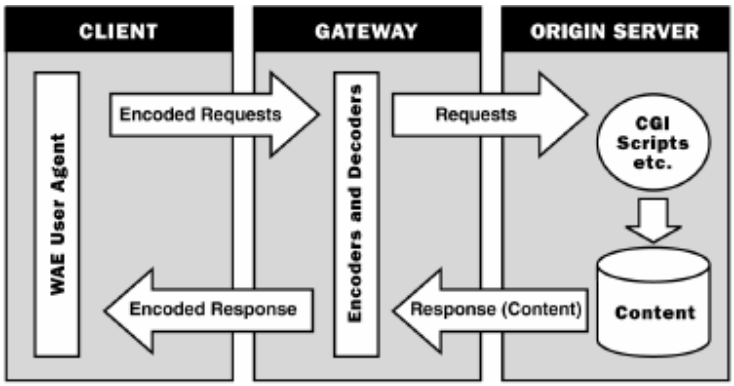
3.2.2 WAP-modellen

WAP-modellen är i princip WWW-modellen med några förbättringar. Fördelarna med att adoptera WWW-modellen ligger bland annat i en redan känd utvecklingsmiljö, en bevisligen fungerande arkitektur och möjligheten att använda redan existerande verktyg (t.ex. webbservrar, XML-verktyg etc.). Optimeringar och tillägg har dock gjorts för att anpassa standarden till en trådlös miljö. Där det har varit möjligt har däremot redan existerande standarder adopterats, eller legat till grund för utvecklandet av WAP-teknologin. (Wapforum.org)

Precis som i WWW har även WAP de standardkomponenter som visades i tabell 1 ovan. Det är till exempel samma modell för namngivning, d.v.s. en URL, som ligger till grund för adressering.

En stor skillnad mellan WWW-modellen och WAP-modellen är att en klient inte direkt kan fråga en server om data, utan all denna kommunikation måste först ske via en gateway (kallas i vissa sammanhang för proxy). Denna gateway är ofta lokaliserad hos

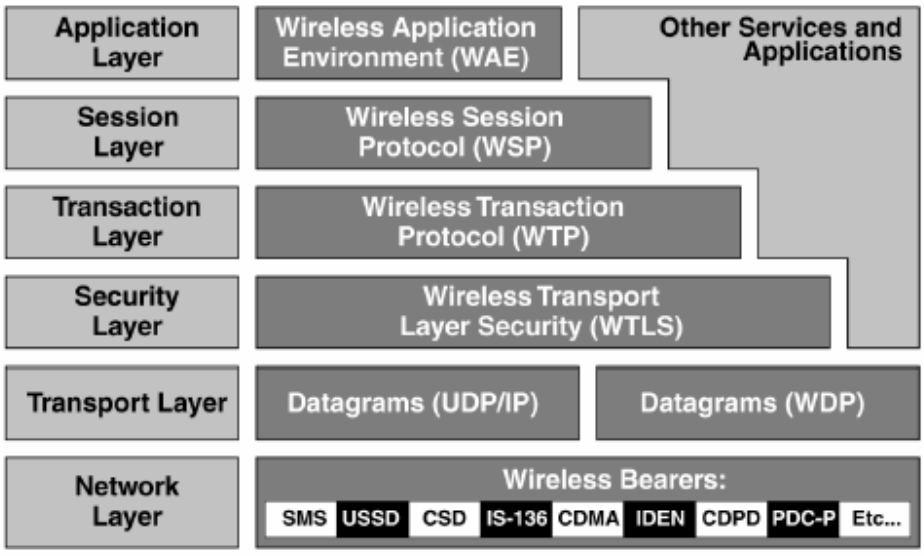
operatören, hos vilken klientägaren har sitt telefonabonnemang. När operatörens gateway har fått en förfrågan från klienten i form av en URL (t.ex. wap.wapmail.com), kodar den om informationen och skickar dem vidare som helt vanliga HTTP-request över Internet till den webbserver där tjänsten ligger. Webbservern skickar sedan tillbaka dessa förfrågningar tillbaka till gatewayen, som kodar tillbaka informationen till et format som WAP-klienten kan läsa. (Wapforum.org)



Figur 2, WAP-modellen (källa: Wapforum.org)

Precis som hos WWW-arkitekturen erhåller WAP-arkitekturen en skalbar miljö för utveckling av mobila tjänster. Detta erhålls genom en skiktad design av hela protokoll-stacken. Denna påminner mycket om OSI-modellen och precis som i denna är alla lager i stacken åtkomliga för lagren ovanför. (Wapforum.org)

Nedan visas en bild på de olika komponenterna inom den skiktade WAP-arkitekturen, vilken följs av en beskrivning av respektive del av WAE. Då denna uppsats inte täcker några större tekniska beskrivningar av dessa delar, kommer inga ingående resonemang runt dessa att föras.



Figur 3, Protokollstacken hos WAP

3.2.2.1 WAE – Wireless Application Environment

WAE är en generell applikationsmiljö där operatörer och tjänsteleverantörer kan skapa applikationer och tjänster för olika trådlösa plattformar. WAE skapades som en kombination av WWW- och mobilteleteknologier som ska ta hänsyn till exempelvis små skärmar, begränsad möjlighet till textinmatning, begränsad bandbredd etc. WAE hanterar följande teknik:

XHTML Mobile Profile – är en del av XHTML Basic, men med ytterligare funktionalitet. Denna är tillgänglig från och med WAE v.2 (WAP 2.0)

XHTML Basic – Även denna stöds eftersom det är ett giltigt XHTML Mobile Profile document. (Även denna är tillgänglig från och med WAE v.2)

WML – Wireless Markup Language, ett märkspråk baserat på XML och fortfarande det vanligaste språket som stöds av alla mobiltelefoner som stöder WAP.

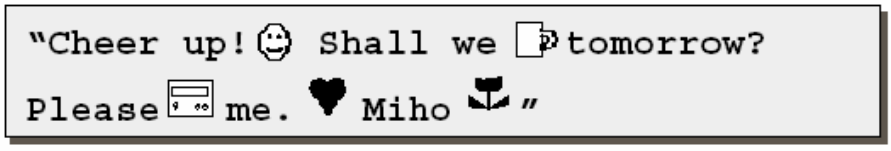
WCSS – Wireless Cascading Style Sheets, ett dokument som anger utformningen av en sida. W3C anger även en specifikation för CSS Mobile Profile, ett minimalt antal delar vilket ska vara passande för mobila enheter.

WMLScript – Ett scriptspråk som påminner mycket om JavaScript, men är nedbantat för att passa mobila enheter.

WBXML – En binär representation av XML som har designats för att reducera den mängd data som skickas över nätet. Här kan en enda byte betyda t.ex. <deck>, vilket går snabbare och är mer kostnadseffektivt.

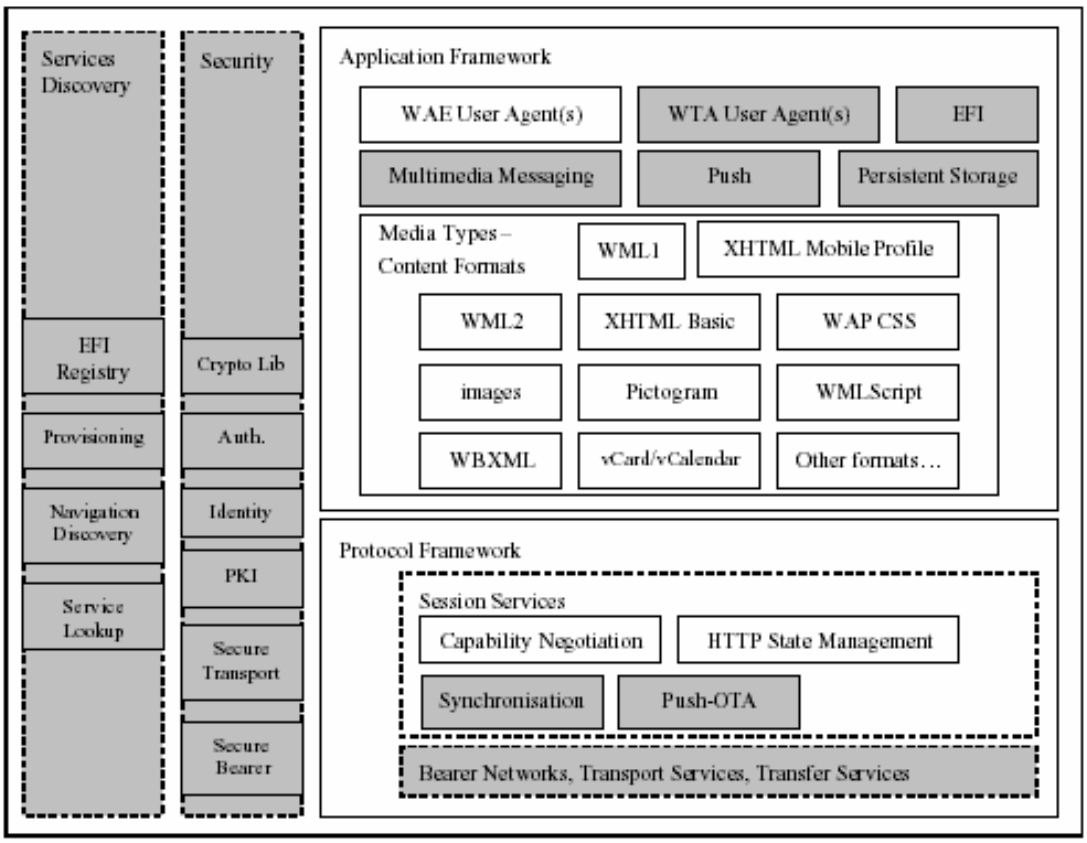
vCard, vCalender – olika format för att kunna skicka objekt från och till den mobila enhetens telefonbok och kalender.

Pictogram – en ikonliknande bild som renderas i en text. I dagligt tal kallat ”smiley”.



figur 4, pictogram (Källa: Wapforum.org)

WTA, WTAI – Wireless Telephony Application (Interface). Olika telefonitjänster samt det programmeringsgränssnittet som används för det. Se kapitel 3.4.



figur 5, WAE-komponenter (källa: Wapforum.org)

3.3 Wireless Markup Language

WML står för Wireless Markup Language och är baserat på XML. Det är utvecklat för att presentera innehåll på handburna terminaler. WML skapades främst av fyra anledningar: (Wapforum.org)

1. Begränsad bandbredd hos GSM-nätet (normalt 9600 bps), vilket medför långa svarstider.
2. Begränsad display på terminalerna. (när WML skapades var de mindre än idag)
3. Begränsat gränssnitt. Standarden stödde enbart bildformatet WBMP
4. Begränsade Resurser. Små processorer, lite RAM, få navigeringsmöjligheter pga. den lilla knappsatsen.

Den som har skrivit hemsidor i HTML eller XHTML kommer relativt snabbt in i WML:s syntax. Precis som i andra märkspråk, innehåller elementen en starttag, ett innehåll och en sluttag: `<tag>` innehåll `</tag>` eller `<tag />`. WML-attribut specificerar extra information om ett element. Dessa är alltid specificerade i elementets starttag: `<tag attr="abcd"/>`.

Jämfört med HTML är WML mycket striktare i sin syntax. Det är till exempel inte tillåtet att nästla olika taggar, t.ex. `<p><b>ett ord </p></b>` kommer att generera ett syntaktiskt fel. Dessutom är WML Case-sensitive, vilket innebär att `<B>` inte är detsamma som `<b>`.

WML inkluderar ett antal olika funktionsområden för textpresentation och layout. Det finns ett antal olika formaterings- och layoutkommandon, t.ex. `<b>` för fet stil. Dessutom är all information i en WML-fil är organiserad efter en metafor där en kortlek (deck) innehåller allting som mobiltelefonen laddar in i minnet samtidigt. Kortleken innehåller sedan ett antal olika kort (cards), vilket utgör det som visas på skärmen för tillfället. De olika korten kan sedan innehålla länkar till varandra, vilket innebär att användaren kan navigera runt på sidorna utan att behöva ladda ner dem en åt gången. Detta är tids- och kostnadsbesparande. (Wapforum.org)

Nedan visar ett enkelt exempel på en första WAP-sida skrivet i märkspråket WML. Likheterna med XHTML är slående, bortsett från `<WML>`-taggen och `<CARD>`-taggen:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML
1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card id="card1" title="WapMail - Login">

      <p>
            Välkommen!!
      </p>


</card>
</wml>
```

Exempel 1. En enkel sida skriven i WML

Figur 6. Förhandsgranskning av sidan i en telefon

WML innehåller även stöd för att händelsehantering då användaren trycker ner knappar på mobiltelefonen. Detta är en viktig funktion hos WML - och ett bra komplement till vanliga länkar, då det skapar en möjlighet till lättare navigering.

```
<do type=”accept” label=”next”>
      <go href=”card2” />
</do>
```

Detta kan till exempel styra telefonens options-meny till att inkludera möjligheten att navigera vidare till nästa sida.

Att utförligt beskriva WML:s syntax vore ett alldeles för omfattande arbete för denna uppsats. Tanken är att presentera olika tekniker och lösningar för läsaren, med vilka denne sedan kan läsa vidare om de delar som speciellt intresserar honom. Därför beskrivs endast kortfattat hur ett WML-dokument är uppbyggt och fungerar. För en fullständig dokumentation om WML rekommenderas ett besök på www.wapforum.org.

3.4 WMLScript

För att ge utvecklaren möjlighet att tillföra mer intelligens till klientsidan, samt kunna skapa ett mer dynamiskt användargränssnitt, skapades WMLScript. WMLScript är ett enkelt, procedurorienterat scriptspråk och påminner mycket om JavaScript. En stor skillnad mot JavaScript är dock att ett WMLScript måste ligga i en separat fil med filändelsen .wmls. Detta är dock något som utvecklingen av XHTML pekat på länge, med separata filer för data (.html), layout (.css) samt script (.js). Tanken har varit att få ett mer moduluppbyggt skapande av sidor. (Ericsson.com), (Ewert, 2000), (wapforum.org), (w3c.org).

WMLScript skapades bland annat som ett komplement till det annars statiska WML. Det var främst inom fyra områden där WMLScript ansågs ha stor potential: (wapforum.org)

- Kontrollera det som användaren matar in.
- Få tillgång till telefonspecifika funktioner. T.ex. att låta användaren komma åt funktioner som att kunna ringa samtal, skicka SMS, få tillgång till SIM-kortet etc.
- Generera meddelanden och dialogrutor lokalt, och därför minimera trafiken över nätet.
- Tillåta utbyggnader till apparatens mjukvara samt att kunna konfigurera den efter apparaten tagits i bruk.

Som nämndes tidigare, kan ett WMLScript inte infogas direkt i WML-filen, utan måste läggas separat i en fil med filändelsen .WMLS. Ett anrop till ett WMLSript kan se ut som i exemplet nedan:

test.wml:

```
<a href="test.wmls#addera($(value1),$(value2))">Addera</a>
```

test.wmls:

```
...
function addera(x,y) {
      var summa = Lang.parseInt(x) + Lang.parseInt(y);
      return summa;
}
...
```

3.4.1 Bibliotek

3.4.1.1 WMLScript Standard Library

WMLScript innehåller ett bibliotek med färdiga funktioner för att öka funktionaliteten och ökar möjligheten att skriva avancerade program. Funktionerna är indelade i ett antal bibliotek:

Lang – generella funktioner

Detta bibliotek innehåller de mest generella funktionerna i WMLScript.

```
tex: var b = Lang.abs(-3);          //b = 3
```

Float – flyttal

Detta bibliotek innehåller aritmetiska funktioner för flyttal.

```
Tex var a=Float.round(3.5);         //a = 3
```

String – strängar

Detta bibliotek innehåller funktioner för stränghantering

```
Tex var a= String.length(123);      //a = 3
```

URL

Detta bibliotek innehåller funktioner för att hantera URL:er.

```
Tex var a = URL.getHost(”http://www.wap.com/pub”);
//a = ”www.wap.com”
tex var b = URL.getQuery(”http://www.wap.com?id=23&namn=daniel”);
// b = ”id=23&namn=daniel”
```

WMLBrowser

Detta bibliotek innehåller funktioner för samarbete med WML och browsern.

```
Tex WMLBrowser.getVar(”namn”);
```

Dialogs

Detta bibliotek innehåller funktioner för gränssnittet mot användaren.

```
Tex Dialogs.alert(”felaktig email-adress”);
```

3.4.1.2 WMLScript Crypto Library

Förutom de standardbibliotek som finns i WMLScript, finns dessutom ett bibliotek helt avsett för säkerhet. Utvecklarna förstod att utan ett säkert sätt att kunna verifiera vem som använder en viss tjänst, så kommer utvecklingen av mobila tjänster att gå mycket långsamt. Som parallell kan nämnas rädslan för att lämna ut kreditkortsnummer på Internet, något många fortfarande är rädda för trots att det statistiskt sker fler kreditkortsbedrägerier i den verkliga världen.

Crypto.signText är en krypteringsfunktion som visar en textsträng och som användaren sedan signerar. Signaturen skickas sedan till servern som tillhandahåller tjänsten tillsammans med datat för att verifiera att det som står där verkligen stämmer. (wapforum.org), (Ewert, 2000)

3.4.1.3 WTAI Library (public.research.mimesweeper.com)

WTAI betyder Wireless Telephony Application Interface och är gränssnittet för ett antal telefonitjänster som kan skapas direkt för telefonen. WTAI är tillsammans med WTA, WML och WMLScript funktioner som finns inbyggt i microbrowsern i telefonen. Microbrowsern är i sin tur en del av WAE (Wireless Application Environment, se ovan). I WTAI Library finns bland annat funktioner för att ringa upp samtal, ta emot samtal, skicka och ta emot textmeddelanden och hantera telefonboken. Dessa funktioner kan av säkerhetsskäl dock bara anropas om de finns på en WTA-server. (wapforum.org), (Ewert, 2000)

De olika funktionerna som återfinns inom WTAI är fördelade inom olika bibliotek (se nedan), beroende på funktion och tillgänglighet. Funktionerna kan anropas både genom WML, genom att använda en URL, eller genom WMLScript. (wapforum.org)

Function Library	***Name***	***Description of Library***
Public WTAI	"wp"	Public available WTAI functions.

Tabell 2, *Public WTA Function Libraries (Wapforum.org)*

Function Library	***Name***	***Description of Library***
Call Control	"cc"	Call Control library. Handles call setup and control of device during an ongoing call
Network Text	"nt"	Network Text library. Sending and retrieval of network text.
Phonebook	"pb"	Phonebook library. Manages the entries in the device phonebook.
Miscellaneous	"ms"	Handling of miscellaneous features. An Example is logical indications.

Tabell 3, *Network Common WTA Function Libraries (Wapforum.org)*

Tillgång till WTAI funktionsbibliotek via WML kan ske genom URI-anrop (URI betyder Uniform resource Identifier, RFC2396). Enkelt uttryckt kan sägas att en länk till funktionen skapas. Genom användandet av fördefinierade referenser till specifika WTAI funktionsbibliotek tillsammans med själva funktionsnamnet bildas WTAI URI:

```
Wtai://<library>/<function> (; <parameter>)* [! <result> (; <result> )* ]
```

<library>	Name that identifies the type of function, ie Call Control uses the library name "cc".
<function>	Function identifier within specific library. An example is "ac" for the function "Accept Call" residing in the library "Network Common WTA".

<parameter>	Zero or more parameters to be sent to the function. Delimiter between subsequent parameters must be a semicolon ";".
<result>	<result> Start of the result data section is indicated by an exclamation mark "!". Result is zero or more names of variables that will be set in the WTA user-agent context as a result from the function call. Delimiter between subsequent result data must be a semicolon ";".

Tabell 4, *WTAI URI Schema (Wapforum.org)*

Nedan följer ett antal olika exempel på olika WTAI-funktioner för respektive bibliotek. Listan är dock långt ifrån komplett och finns endast med för att demonstrera hur principen fungerar. En komplett lista återfinns i slutet av uppsatsen i form av ett appendix. Vill läsaren däremot ha utförliga beskrivningar av respektive funktion rekommenderas ett besök på www.wapforum.org.

Public WTAI

Funktioner inom detta bibliotek är även tillgängliga för applikationer som hämtats från en WTA-server, d.v.s. third party-applikationer. Skillnader ligger bland annat i att användaren måste kunna avbryta specifika operationer innan de är färdiga.

Make Call

Description	This function is used to initiate a mobile originated call using the specified *number*. The user must explicitly acknowledge the operation. The *Make Call* function can be used from within any application, not only WTA, to present the user with a number that can be dialled.
URI:	wtai://wp/mc ; <number>
WMLScript:	makeCall(*number*);
Function ID:	0
Parameters:	<number> = String: Destination number to call. May use any valid telephony number characters and digits.
Output:	-
Examples:	**URI**: wtai://wp/mc; 5554367 **WMLScript:** WTAPublic.makeCall("5554367");
Associated Events:	-
Notes:	The call must be terminated using the standard MMI.

Tabell 5, *Make Call (Wapforum.org)*

Network common WTAI

Funktioner i detta bibliotek fungerar på alla typer av mobila nätverk, där WAP fungerar.

Send Text

Description	Sends a network text message, if feature is available in the network, to a destination identified by number.
URI:	-
WMLScript:	send (number, text);

Function ID:	0
Parameters:	<number> = String: Destination number. Any valid telephony characters and digits. <text> = String: Network text data structure, the text to send
Output:	<result> = String: Integer value below zero indicates unsuccessful execution.
Examples:	WMLScript: WTANetText.send ("5554567", "WAP Forum");
Associated Events:	-
Notes:	-

Tabell 6, *Send Text (Wapforum.org)*

GSM specific WTAI

Förutom de generella delarna inom WTAI finns även funktioner som är specifika för respektive nätverk som används. Här återfinns t.ex. funktioner för GSM, IS-136 och PDC. Den GSM-specifika delen innehåller bland annat funktioner för att t.ex. parkera samtal, vidarekoppla samtal och skapa telefonkonferenser. (wapforum.org)

WTAI är ett kraftfullt komplement till WML och WMLScript, med möjligheten att skapa kraftfulla och ”smarta” lösningar. Trots detta är det inte många företag som tagit vara på denna teknik. Ett undantag är dock Eniro, kanske mest känt för att ha lagt ut telefonkatalogen på Internet. Användaren kan numera även söka privatpersoner via sin mobiltelefon och ringa upp direkt utan att behöva skriva in telefonnumret. En funktion som saknas är dock möjligheten att spara personen i adressboken. WTAI biblioteken skapar även gyllene möjligheter för de olika mobiloperatörerna att skapa bättre och mer praktiska portaler för sina kunder.

Exempel på anrop av WTAI

Nedan följer ett exempel på en WTAI-funktion genom ett URI-anrop. Användaren väljer vilket slags mat han är intresserad av (de olika alternativen återfinns inom <OPTION>-taggarna). När ett alternativ sedan är valt kopplas ett samtal upp genom raden <DO> som är kopplad till WTAI-funktionen make call. Det första ”kortet” innehåller endast en länk till beställningssidan som finns i kort 2.

```
<WML>
   <CARD>
     <DO TYPE="ACCEPT" TASK="GO" URL="#eFood"/>
     Welcome!
   </CARD>

   <CARD NAME="eFood">
     <DO TYPE="ACCEPT" TASK="GO" URL="wtai://cc/mc;$FoodNum"/> Choose Food:
<SELECT KEY="FoodNum">
   <OPTION VALUE="5556789">Pizza</OPTION>
   <OPTION VALUE="5551234">Chinese</OPTION>
   <OPTION VALUE="5553344">Sandwich</OPTION>
   <OPTION VALUE="5551122">Burger</OPTION>
<SELECT>
</CARD>
</WML>
```

Nedan följer ett exempel på en WTAI-funktion genom WMLScript-anrop.

WMLScript:

```
function CallFood(N) {
var i = wtaCallControl.Setup(N;1);

if (i >= 0) {
      // Call is good, show call is done
      Browser.setVar("Msg", Called");
      Browser.setVar("Nmbr", N);
}
else {
      // Call failed, we could tell user why
      Browser.setVar("Msg", "Error");
      Browser.setVar("Nmbr", $i);
   }

   Browser.go("displayMsg");
}
```

WML:

```
<WML>
   <CARD>
  <DO TYPE="ACCEPT" TASK="GO" URL="/script#CallFood($FoodNum)"/>
Choose Food:

<SELECT KEY="FoodNum">
   <OPTION VALUE="5556789">Pizza</OPTION>
   <OPTION VALUE="5551234">Chinese</OPTION>
   <OPTION VALUE="5553344">Sandwich</OPTION>
   <OPTION VALUE="5551122">Burger</OPTION>
<SELECT>
</CARD>

<CARD NAME="displayMsg">Call Status: $Msg $Nmbr </CARD> </WML>
```

Exemplen ovan visar hur enkelt det är att komma igång med dessa kraftfulla men ändå enkla metoder för att skapa WTAI-tjänster. Möjligheten till felkontroll i det senare

exemplet är en bra anledning till att välja WMLScript-metoden, även om den kan verka svårare att komma igång med.

Vid en genomgång av specifikationer av de vanligaste mobiltelefonerna från Sony Ericsson visar det sig att flera av modellerna stöder Make Call, Send DTMF och add Phone Book

3.4.1.4 ”tel” scheme

tel-schemat specificerar ett telefonnummer. På de flesta hemsidor på Internet idag finns en länk till en emailadress, vilken låter besökarna skicka ett email till t.ex. webmaster eller support. Om sidan däremot besöks från en mobiltelefon skulle det ju vara mer passande med en länk för att ringa upp.

Exempel: `<a href="tel:+46708231223">Ring oss! </a>`

Dessutom kan ett antal DTMF-toner kopplas till länken för att på så sätt styra samtalet till t.ex. support, voice mail och så vidare.

Exempel

```
<a href="tel:+46708231223;postd=2922#">
Supportavdelningen</a>
<a href="tel:+46708231223;postd=9#">
Voice mail</a>
```

(“URLs for Telephone Calls", RFC2806, www.ietf.org/rfc/rfc2806.txt”), (Ericsson.com)

Vill tillverkaren av sidan skapa ytterligare möjligheter för besökaren att kontakta finns ytterligare tre möjligheter med smsto, mailto och mmsto. I alla dessa exempel öppnas en editor för respektive tjänst.

Exempel

```
<a href="smsto:+46708231223">skicka ett SMS!</a>
<a href="mailto:support@wapmail.com">skicka ett email!</a>
<a href="mailto:maillist@wapmail.com?subject=subscribe">
anmäl dig till listan.</a>
<a href="mmsto:+46708231223">skicka ett MMS!</a>
```

Dessa tjänster verkar dock inte speciellt spridda bland de senaste telefonmodellerna, men troligtvis något som kommer dyka upp mer och mer.

3.5 XHTML

Det språk som de flesta förknippar med Internet idag är HTML (HyperText Markup Language). Det är ett språk som bygger på SGML (Standard Generalized Markup Language), vilket togs fram för seriöst dokumentarbete och har varit en internationell standard för hur man märker en text sedan 1986. SGML är ett kraftfullt språk, där all typ av information i en text kan märkas och definieras otvetydigt.

Precis som andra SGML-språk använder sig HTML av en DTD (Document Type Definition) för att specificera reglerna för hur syntaxen ska tolkas, tex. Utan dessa regler skulle en webbläsare inte veta hur den ska tolka en sida skriven i t.ex. HTML. Här återfinns dock ej beskrivningen för att tex en <b>-tagg ska generera fet stil utan snarare hur själva tagg-uppbyggnaden är konstruerad. Tanken var att det i sidhuvudet skulle finnas en länk till vald DTD, så att programmet som visade sidan (t.ex. en webbläsare) skulle kunna ”slå upp” de regler som gällde för just det här dokumentet. I HTML är dock DTD:n inbäddad i webbläsaren. (Boumphrey et Al, 28)

Att både t.ex. Internet Explorer och Netscape hade inbyggda DTD:er och att dessa dessutom stödde ett flertal egenutvecklade regler för hur en sida kunde visas, gjorde att en sida visad i t.ex. Netscape kunde se helt annorlunda ut i Internet Explorer. Dessutom var dessa webbläsare konstruerade så att de även kunde visa inkorrekt skriven HTML (Något som dock har bidragit till Internets popularitet, eftersom i princip alla kunde lära sig skapa en hemsida). För varje ny version av respektive webbläsare hade några nya taggar lagts till i den inbyggda DTD:n och programmen bara växte och växte.

Att programmen var så snälla med att tillåta felaktigt skriven HTML gjorde att det blev krångligt att programmera för dem och att programmen blev stora och processorkrafts-krävande, eftersom en webbläsare behöver använda mycket av datorns processorkraft till att ”räkna ut” hur det skulle se ut. Problemet med detta synsätt, som skulle visa sig senare, är anslutandet av andra användaragenter, såsom en mobiltelefon eller PDA. Dessa agenter har varken den processorkraft för att klara av dåligt skriven kod eller det utrymme som skulle krävas för det allt större utrymmet som krävdes av applikationerna.

Detta ledde till utvecklandet av ett nytt språk – XML (eXstensible Markup Language). Tanken var att det skulle vara striktare än HTML och lätt att lära sig och använda över Internet. Dessutom skulle det vara kompatibelt med SGML och dess idé om beskrivande snarare än strukturell uppmärkning (Taggar skulle alltså beskriva innehållet i texten snarare än hur det skulle presenteras). (www.w3.org, Boumphrey et Al, 28)

XML skiljer sig dessutom från HTML genom att det är tänjbart. Nya taggar kan skapas när som helst, av vem som helst. Trots detta kräver inte språket någon DTD eftersom språket är så strikt. Alltså så länge dokumentet är välskrivet behöver inte användaragenten (t.ex. browsern) en DTD för att kunna läsa det. Även detta reducerar processorkraft, vilket är av yttersta vikt för användning i t.ex. en mobiltelefon. (www.w3.org, Boumphrey et Al, 28)

Ur XML skapades sedan XHTML, en kombination av XML och HTML. Språket har tagit med sig fördelarna från båda språken genom att använda sig av HTML-språkets vokabulär, vilket gör det lätt att lära sig, samtidigt som det använder sig av XML-språkets grammatik, vilket gör att det kan visas på alla XML-användaragenter. (www.w3.org, Boumphrey et Al, 28)

Märkspråken som stöds hos WAP 2.0 är förutom WML även XHTML Basic och XHTML Mobile. Dessa två undergrupper till XHTML stöds idag av alla webbläsare på Internet.

```
<?xml version=”1.0” encoding=”UTF-8”?>
<!DOCTYPE html PUBLIC “-//WAPFORUM//DTD XHTML Mobile 1.0//EN”
  “http://www.wapforum.org/DTD/xhtml-mobile10.dtd”>
<head><title>WapMail </title></head>

<body>
          <p>Välkommen till WapMails webgränssnitt</p>
     </body>
</html>
```

Exempel 2. En enkel sida skriven i XHTML

Exemplet ovan visar en enkel sida som både är ett välutformat XHTML-dokument och XML-dokument. Likheterna med HTML är slående bortsett från de första taggarna i dokumentet. Som nämndes tidigare behövs detta för att en ren XML-användaragent inte har någon inbyggd DTD och följaktligen måste dokumentet länka till en sådan. Beroende på vilken DTD som används måste olika doctype-taggar användas. (www.ericsson.com)

XHTML Mobile Profile (stöds från och med WAP 2.0)

```
<!DOCTYPE html PUBLIC ”-//WAPFORUM//DTD XTHML Mobile 1.0//EN”
”http://www.wapforum.org/DTD/xhtml-mobile10.dtd”>
```

XHTML Basic (stöds från och med WAP 2.0)

```
<!DOCTYPE html PUBLIC ”-//W3C//DTD XTHML Basic 1.0//EN”
”http://www.w3.org/TR/xhtml-basic10.dtd”>
```

WML 1.3 (stöds från och med WAP 1.2.1)

```
<!DOCTYPE wml PUBLIC ”-//WAPFORUM//DTD WML 1.3//EN”
”http://www.wapforum.org/DTD/wml10.dtd”>
```

Alla WAP 2.0 webbläsare stöder både XHTML Mobile och XHTML Basic. XHTML Basic är en del av XHTML Mobile profile och stöds av andra applikationer än WAP browsers. Denna version är alltså mer generell och flera funktioner har plockats bort för att göra den så tillgänglig som möjligt. Om sidan ska göras tillgänglig för andra applikationer än de som stöder WAP 2.0 är denna alltså att föredra framför XHTML Basic.

Enligt Ericssons developers guideline, ett dokument som ges ut för utvecklare av WAP-tjänster, står att WML 1.3 enbart stöds för bakåtkompabilitet. Detta tyder alltså på att WML håller på att försvinna som det självklara språket hos WAP.

En stor fördel med XHTML jämfört med WML är att sidorna inte måste konstrueras om beroende på för vilket medium de är skapade. Sidan ovan skulle t.ex. fungera bra i såväl en webbläsare som en modern mobiltelefon eller PDA. Det bör dock nämnas att relativt få mobiltelefoner i dagsläget stöder XHTML, som inkluderas i WAP 2.0. Dessutom kan sidorna variera i utseende beroende på skärmstorlek, upplösning etc.

En annan stor fördel jämfört med WML är möjligheten att kunna skapa bättre design på sidorna. Genom att länka till en extern fil med filändelsen .css (Cascading Style Sheet), kan skaparen sätta upp ett antal regler för hur sidan ska presenteras. Det är även möjligt att länka flera formatmallar till samma dokument, beroende på vilken användaragent som kommer visa sidan. På detta sätt kan skaparen skapa en plattformsoberoende hemsida, som kommer att visas och se bra ut i alla användaragenter, för vilka han har specificerat en stilmall.

I exemplet nedan har ett XHTML-dokument länkats samman med två olika formatfiler som kommer avgöra utseendet för respektive plattform.

hemsida.html

```
<head>
   <link rel="stylesheet" type="text/css" media="handheld" href="mobil.css" />
   <link rel="stylesheet" type="text/css" media="screen" href="dator.css" />
</head>
```

dator.css

```
p {   font-size:12pt;
      margin-left:3%;
      color:yellow;
      background-image: url(stor_bild.gif);
}
```

mobil.css

```
p {   font-size:8pt;
      color:black;
}
```

Exempel 3. En sida, två utseenden

Här anger kommandot `media="handheld"` samt `media="screen"` vilken formatmall som ska användas för respektive användaragent. Begreppet handheld är ett ganska vitt begrepp än så länge, och syftar till alla handburna enheter. Behövs en snävare avgränsning än så, t.ex. en formatmall för Ericsson-telefoner, en för Nokia-telefoner och en för Palm-PDA's, måste WAG UAPROF användas. Detta är en komponent som finns

inbyggt i WAP och innehåller information om klienten. Denna information skickas sedan till servern och där kan informationen användas för att skapa skräddarsydda sidor.

Det går även att åstadkomma samma funktion som `media="handheld"` med funktionen @media. På så sätt kan alla formateringsinstruktioner ligga i samma fil.

```
@media screen {
      h1 { font-size:18pt; }
}

@media handheld {
      h1 { font-size:12pt;}
}
```

Exempel 4. Användning av @media

På samma sätt som det går att länka externa CSS-dokument till en XHTML-sida, går det även att länka JavaScript till sidan för att skapa ett mer dynamiskt innehåll. Syntaxen för JavaScript täcks inte inom ramen för denna uppsats eftersom det är alldeles för omfattande, men för den intresserade finns hyllmeter av litteratur att läsa.

4 Utveckling av mobila tjänster

4.1 Grundkunskaper

Att utveckla mobilbaserade lösningar som bygger på internetteknik, innebär att man måste sätta sig in i en hel del nya tekniker och arkitekturer. Till skillnad från t.ex. vanlig applikationsprogrammering för windowsplattformen, där en hel applikation kan skapas enbart med tillgång till ett verktyg (t.ex. Visual C++), måste för skapandet av en mobil lösning en rad olika applikationer installeras och läras. Även ett flertal språk/tekniker måste behärskas innan man kan skapa en fullgod applikation.

Innan konstruktionen av ett mobilbaserat system startar, bör tillverkaren ha svar på en rad frågor, för att redan från början förbereda sig så mycket som möjligt med rätt kunskaper, verktyg etc. Några frågor som kan vara bra att besvara direkt är:

- Ska en helt ny tjänst skapas, eller konverteras från en redan existerande?
- Vilken publik/målgrupp riktar sig tjänsten till?
- Vilka plattformar ska tjänsten fungera på?
- Vad är viktigast – bakåtkompabilitet eller framåtkompabilitet?
- Är tjänsten en gratistjänst eller behövs ett system för att debitera användarna?

Dessa frågeställningar har framkommit under det här projektets gång och därmed skapat en del onödigt merarbete. Att t.ex. under projektets gång ändra tjänstens förväntade målgrupp eller vilka plattformar den är tänkt att fungera på, kan mer eller mindre innebära att projektet måste starta om.

Tanken med det här projektet var att skapa en fungerande prototyp av en mobilbaserad emailklient. Användaren ska alltså först besöka en vanlig Internetbaserad hemsida för att där skapa ett användarkonto, där han anger en eller flera olika pop3-konton, t.ex. en emailadress till jobbet, en privat, en för skola, föreningsaktivitet etc. På grund av den tid det tar att ständigt behöva logga in på diverse sidor med olika användarnamn och lösenord för att läsa av alla sina emailkonton, sågs det enklare om användaren alltså kunde läsa email från alla dessa konton – och dessutom kunna nå dessa via sin mobila klient. Servern kommer i det här fallet alltså både att agera som webbserver och WAP-server. Då uppsatsens primära mål är mobilt Internet kommer däremot inte någon större vikt läggas vid publicering av den webbaserade delen. Kodexempel från denna prototyp återfinns längst bak, under kapitlet Källkod, och kommer ständigt refereras till.

För detta projekt skapades en ny tjänst för den breda publiken (med bred i det här fallet menas att alla telefoner med WAP inbyggt är kompatibla. Tjänsten är alltså inte låst till någon speciell telefonleverantör eller operatör.) Tjänsten ska fungera på alla plattformar som stöder WAP 1.3 eller senare, vilket de flesta WAP-telefoner klarar av. Detta innebär att även de nya telefonerna, som numer stöder XHTML och alltså bara WML på grund av bakåtkompabilitet, även kan presentera tjänsten. Om tjänsten hade skapats i XHTML istället, hade den troligtvis kunnat presenteras bättre på de nyare terminalerna på marknaden, men en hel målgrupp med äldre/billigare telefoner hade alltså gått förlorad. I detta avseende är tjänsten alltså snarare bakåtkompatibel än framåtkompatibel. Tjänsten

skapas som en ren gratistjänst, dock ur en operatörs synvinkel. Detta blir alltså en simulerad operatörstjänst.

4.2 Webbservern

När väl riktlinjerna för tjänsten är beskrivna, måste olika tekniker läras för att konkretisera dessa idéer. Till att börja med, måste tjänsten rent fysiskt ligga någonstans. Precis som vid skapandet av en ren Internettjänst, läggs de olika delarna som ska presenteras för besökaren, upp på en webbserver. Idag finns en mängd olika webbservrar på marknaden, för denna tjänst har dock **Apache Webserver 2.0.45** valts. En genomgång av de olika alternativ till denna täcks tyvärr inte av denna uppsats.

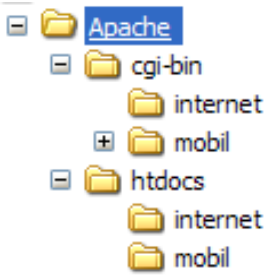
Installation av en webbserver är inga som helst svårigheter och är mer eller mindre förväntade kunskaper hos läsaren. Något som däremot bör nämnas är konfigurering för visning av andra MIME-typer. En webbserver är standardkonfigurerad för att kunna visa filer med filändelser som .html, htm och ibland även .cgi, .pl och .asp etc. utan speciell konfiguration. Ska servern däremot kunna fungera som en WAP-server, kan det bli nödvändigt att ställa in dessa. En utförligare lista av olika MIME-typer återfinns sist i uppsatsen som ett appendix.

Innehåll	MIME-typ	Filändelse
WML	text/vnd.wap.wml	.wml
WML, kompilerad	application/vnd.wap.wmlc	.wmlc
WMLScript	text/vnd.wap.wmlscript	.wmls
WMLScript, kompilerad	application/vnd.wap.wmlscriptc	.wmlsc
Wireless Bitmap	image/vnd.wap.wbmp	.wbmp

Tabell 7, MIME-typer i Apache

Ligger webbservern inte lokalt på den egna datorn, utan t.ex. är en hyrd tjänst kanske hos någon ISP (Internet Service Provider), måste dennes administratör alltså kontaktas för att kunna ställa in de rätta MIME-typerna.

För att kontrollera att servern fungerar riktigt och skapa startsidan till prototypen, skapades index.wml (se kap. Källkod). Denna sparades ner i den katalogen som är förinställd för att presentera sidan (hos Apache heter denna htdocs). Det bör nämnas vikten av att på en gång skapa ordentliga undermappar för att hålla reda på vilka filer som hör till vad. Index.wml sparades alltså ner till Apache/htdocs/mobil, där även resten av .wml-filerna kommer sparas.



Hur kontrolleras resultatet då? Jo, för det behövs någon som simulerar en mobiltelefon eller PDA. För dem med fast tillgång till Internet kan hitta en sådan på www.wapsilon.com. Denna kan sedan testa sidan över nätet. För dem som behöver kunna testa sin sida löpande, utan tillgång till Internet finns flera bra emulatorer att ladda ner gratis på Internet.

Figur 7, Apache-trädet med htdocs för wml-filer.

4.3 Utvecklingsmiljöer och Emulatorer

Det finns ett par utvecklingsmiljöer speciellt anpassade för skapandet av mobila ”hemsidor”. Flera mobiltelefontillverkare har skapat sådana och de kommer ofta med emulatorer som påminner om just deras egna telefoner. Detta för att applikationerna ska se så bra ut med just deras egna telefonmodeller. Dessa utvecklingsmiljöer kan fungera bra vid byggandet av statiska sidor. Bygger lösningen däremot på ett antal bakomliggande serverscript som genererar WML-koden dynamiskt är denna lösning tyvärr inte att föredra eftersom koden ändå måste skrivas om för respektive script-språk. Om tex en wml-sida ska genereras automatiskt via språket perl skrivs tex taggen `<wml>` som `print ”<wml>”;`

Däremot kan dessa verktyg användas för att skapa koden som de serverbaserade scripten sedan ska generera. Verktygen kan även fungera bra för att skapa rena prototyper.

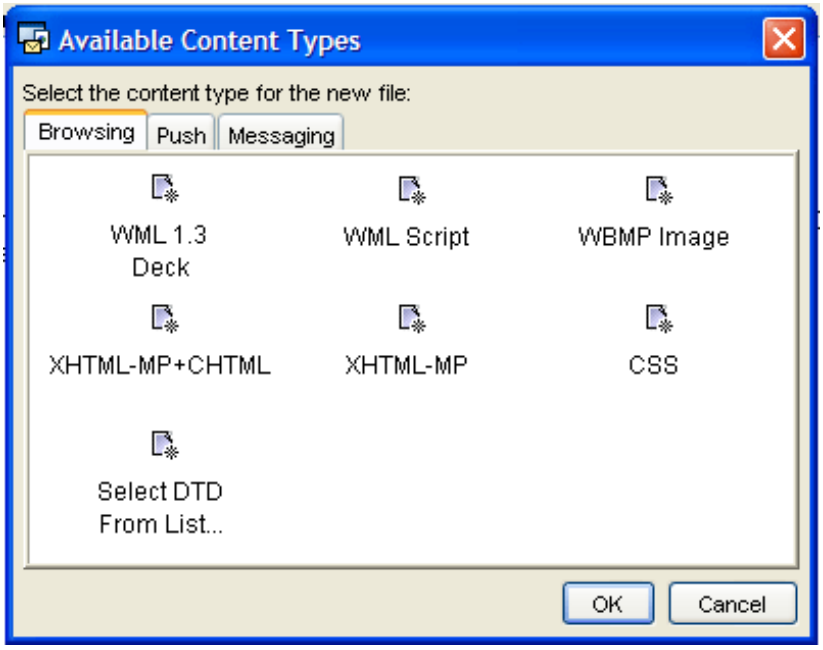
Det går också givetvis att använda sig av en vanlig texteditor, såsom Notepad, Textpad eller Emacs. Fördelen med dessa är att de är billiga, lätta att få tag i och att man kan sätta igång med kodandet på en gång. Det behövs alltså ingen inlärningstid för själva applikationen först. Programmen har dock varken syntaxkontroll eller emulatorer inbyggt, men ger en bättre kontroll över koden, eftersom de inte lägger till eller gör ändringar i den.

Notepad är kanske det enklaste och vanligaste verktyget. Eftersom det ingår i Windows-plattformen är det kanske det vanligaste verktyget att börja med. Textpad fungerar precis som notepad, men med den fördelen att programmet kan skapa lättläsligare kod. Med några få inställningar skapar programmet en färgglad kod med indenteringar på rätt platser.

Nedan följer en genomgång av några lite mer sofistikerade utvecklingsverktyg, speciellt anpassade för skapandet av WAP-sidor.

4.3.1 Nokia Mobile Internet Toolkit 4.0

NMIT är en gratis utvecklingsmiljö för skapandet av mobila tjänster. Efter att applikationen laddats ner från företagets hemsida, är den lätt att komma igång med. Antingen för att öppna ett befintligt dokument eller skapa ett nytt. I det senare fallet kommer en ruta upp som frågar precis vad man vill. Här kan användaren välja mellan att skapa ett dokument med WML, WMLS, WBMP-bild, XHTML, CSS samt även Push-meddelande eller MMS. Oberoende av vilket val som görs, så kommer ett färdigt skal upp som det sedan bara är att fortsätta på. En ypperlig tjänst för nybörjare som inte är så insatta i språket.



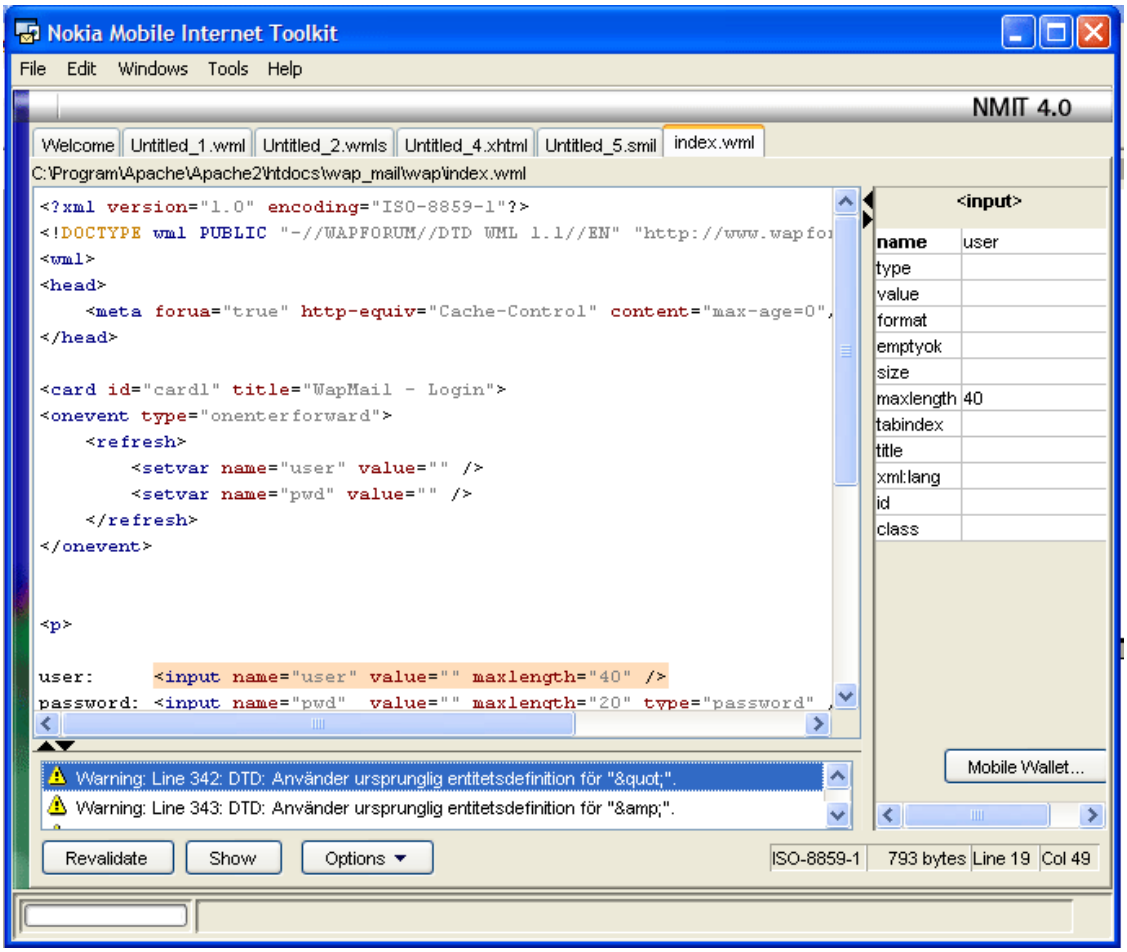
figur 8, nytt dokument

Programmet ger även syntaktisk kontroll av allt som skrivs, och ger ett meddelande om error eller varningar. Detta spar givetvis en mycket tid, jämfört med en vanlig texteditor, där fel kommer att upptäckas först vid förhandsgranskning. Dessutom ger den förslag på attribut och ett lättanvänt gränssnitt gör det snabbt att lära sig.

Andra bra funktioner är att programmet noterar storleken på den kompilerade filen (som sparas i samma katalog med filändelsen .wmlc). Det går även att ställa in så att programmet ger varningar om storleken överstiger vissa värden. Här går att ställa in olika för olika telefoner. Givetvis endast Nokia-telefoner, vilket är lite synd. Detta är annars en riktigt bra funktion, då det är den kompilerade filen som kommer skickas till mobil-telefonen och det är viktigt att den inte blir för stor.

När en sida är skapad kan den med ett knapptryck förhandsgranskas i en inbyggd emulator. Denna är dock klumpig och ful och visar inte sidorna som de flesta andra emulatorer. Den har inställningar för höjd och bredd, med det hade varit fördelaktigare

med ett val för olika telefonmodeller. Stöd för WTAI samt cookies får dock anses som fördelaktigt.



figur 9, WML-dokument

På det hela får NMIT ett gott betyg, bortsett från den inbyggda emulatorn. Detta kan dock lösas genom att ladda ner en separat emulator från någon annan tillverkare. En telefonoberoende emulator är M3Gate's 3tl WML emulator (www.mywap.to). Vill man däremot ha en emulator som påminner mer om de som säljs på marknaden kan man testa Deck-it WML Previewer (www.pyweb.com). Denna ser ut precis som en Nokia 7110. Den absolut bästa emulatorn är dock Ericssons wapIDE 3.2.1 browser. I denna finns nämligen möjligheten att växla mellan olika telefonmodeller (givetvis Ericssons egna).

Ytterligare en utvecklingsmiljö som bör nämnas är WML Editor 3.22. Denna editor fungerar precis som notepad, men med skillnaden att den skapar själva kodskalet, samt att det går att förhandsgranska i flera externa emulatorer (Någon sådan följer alltså inte med). Den är enkel att använda, men kräver lite mer kunskaper om språket. (http://www.jan-winkler.de/dev/d_wmle.htm)

Det finns alltså ingen helt perfekt utvecklingsmiljö för WAP-sidor, utan den bästa miljön skapar man själv genom att kombinera den/de verktygen som verkar bäst. En sida bör

även kontrolleras på en rad olika emulatorer för att garanterat se bra ut såväl på en Nokia-telefon som på t.ex. en Ericcson-telefon.

4.4 Scriptmotorn

Den första sidan skapades ganska enkelt i WML och genom att spara den i htdocs-katalogen kunde den visas på en mobiltelefon. Som syns i bilden nedan finns en ruta där användaren får mata in sitt användarnamn samt lösenord. Vad händer sedan när han trycker på *go*?

Det är här scriptprogrammeringen kommer in. Tanken är att länken ska kopplas till ett script som jämför de inmatade uppgifterna med en databas, och beroende på vilken användare, ladda in viss information till nästa sida.

För att kunna använda script på sidorna måste en s.k. scriptmotor installeras på servern. Det behövs eftersom webbservern inte från början har stöd för alla tänkbara programmeringsspråk som finns på Internet idag. Det finns en uppsjö olika alternativ att välja bland, t.ex. ASP (Active Server Pages), Perl, PHP etc. För detta projekt har programmerings-språket Perl valts, som passar utmärkt ihop med Apache.

För att installera Perl räcker det med att ladda hem installationsfilen (ActivePerl-5.8.0.806) från dess hemsida (www.perl.org). Beroende på operativsystem klickar man sedan på setup (windows) eller kompilerar filerna själv (linux). Själva installationen är det inget konstigt med. Däremot måste webb-servern konfigureras en gång till för att den ska veta att filer med filändelsen .pl först måste gå via script-motorn innan sidan kan skickas till webbläsaren.

Figur 10, inloggningsfönster

För att testa att scriptmotorn är rätt installerad och rätt konfigurerad med webbservern, bör en enkel testsida skapas.

Testsida.pl

```
#!C:\program\Perl\bin\perl.exe

use CGI;
use CGI qw(:standard);

print "Content-type: text/vnd.wap.wml\n\n";
print "<?xml version=\"1.0\"?>\n";
print "<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN\"
\"http://www.wapforum.org/DTD/wml_1.1.xml\">\n";
print "<wml>\n";

print"<card id=\"testsida\" title=\"testsida\">\n";
print"<p> Det fungerar</p>";
print "</p></card></wml>\n";
```

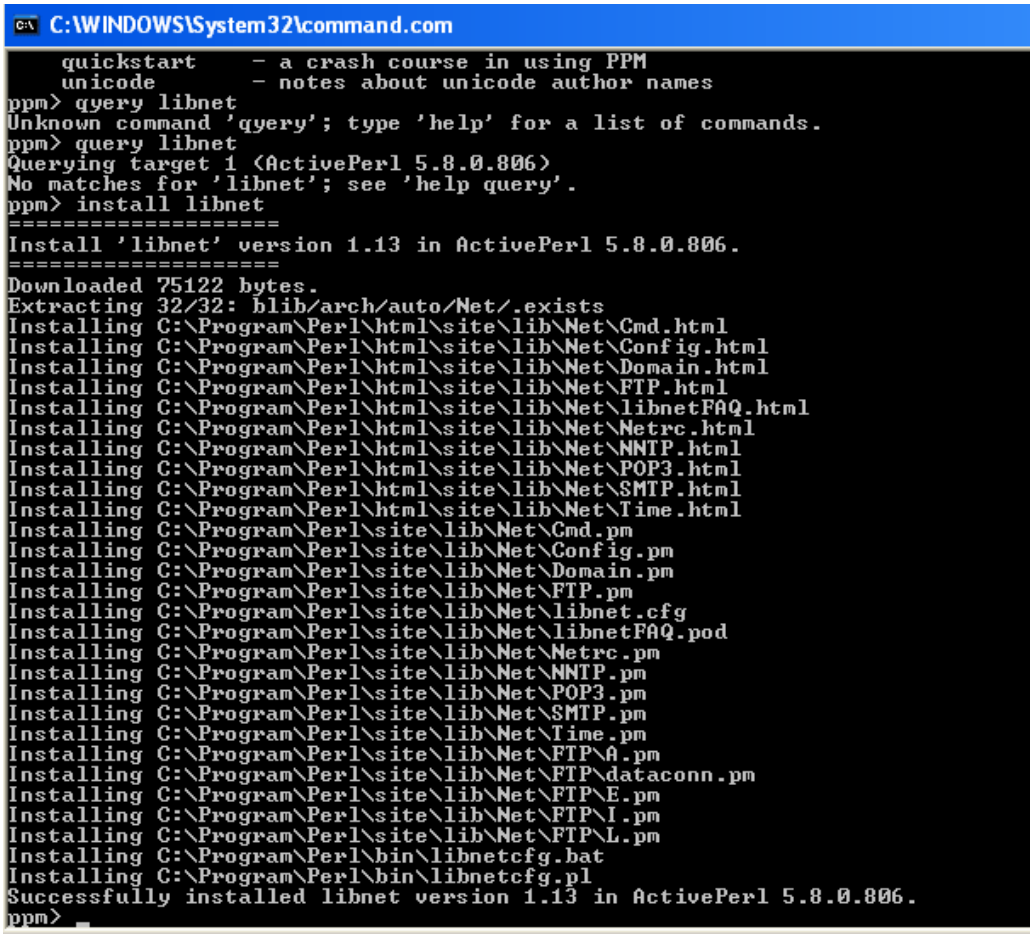
Som framgår av exemplet ovan, ser det precis ut som en vanlig WML-sida, bortsett från att alla WML-taggar måste omslutas med `print " <tagg> ";` Denna Perl-sida genererar alltså en WML-sida som kan visas i en mobiltelefon eller PDA. Överst på sidan återfinns kommandot `print "Content-type: text/vnd.wap.wml\n\n";`
vilket visar att det är en WML-sida som skapas.

Ytterligare några saker måste göras för att inloggningsscriptet ska fungera. Perl kommer med ett antal förinstallerade moduler som räcker för ganska mycket. Ibland bör dessa dock kompletteras med nyare för lite mer specifika behov. För detta projekt krävs moduler för databaskoppling, samt en modul för nätverksprotokoll.

Modul	**Beskrivning**
Libnet 1.13	Collection of Network protocol modules
DBI 1.37	Database independent interface for Perl
DBD-mysql 2.1026	A MySQL driver for the Perl5 Database Interface (DBI)

Tabell 8, moduler hos perl

Dessa installeras genom att skriva ppm i kommandoprompten. (PPM är ett inbyggt program i Perl, och följer med vid installation) Väl inne i ppm installeras modulerna med kommandot *install*. Detta är dock något som kräver en Internetuppkoppling.



figur 11, Installera moduler med PPM

4.5 Databasen

Ytterligare en komponent saknas för att kunna skapa en fullgod affärsapplikation – databasen. Tanken var ju att användaren skulle kunna registrera sig på hemsidan med uppgifter om emailkonton, användarnamn och lösenord etc. Någonstans måste denna information lagras och det görs i databasen.

Även här finns en uppsjö olika alternativ. Jag har dock valt mySQL, vilket är en gratis DBMS (DataBase Management System). Denna kan hittas på www.mysql.org och installeras utan problem. mySQL administreras via kommandoprompten, vilket går ganska smidigt efter en liten inlärningsperiod. Dessutom kan script skapas och sparas i vanliga text-filer för att sedan köras vid ett senare tillfälle. Detta är passande under utveckling, speciellt för att fylla databasen med testdata.

För detta projekt skapades en databas med tre olika tabeller

- Accounts – Lagrar information om olika POP3-konton.
- Messages – Lagrar alla email.
- Person – Lagrar alla personliga data.

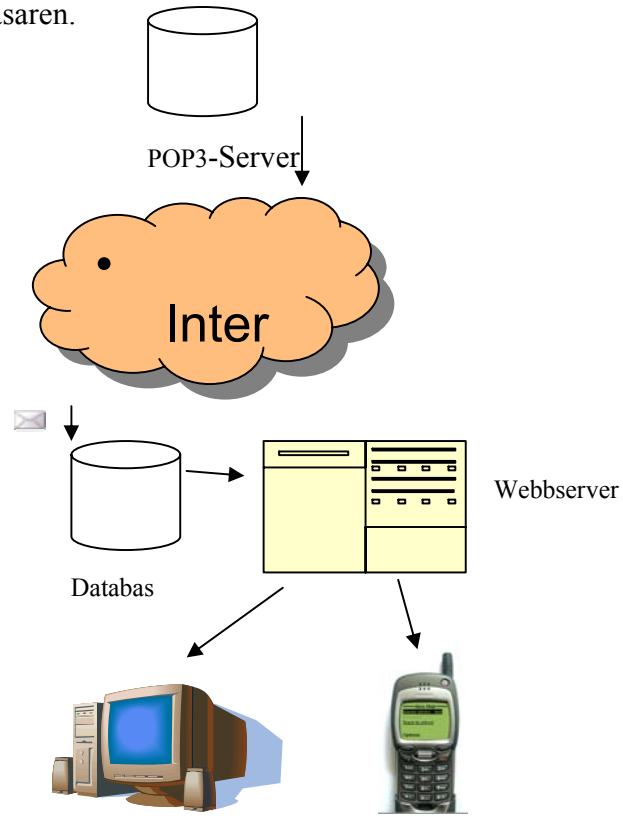
4.6 Projektets arkitektur

Projektet fungerar genom att ett batch-script hela tiden går igenom tabellen Accounts och plockar ut kontouppgifter till alla konton som finns där. För varje konto som hittas loggar det in på respektive kontos server med adress, användarnamn och lösenord, frågar om det finns några nya email i inkorgen, och i så fall laddar det hem dem och sparar ner alla i tabellen Messages. Det är här som nätverksprogrammeringen med modulen Libnet kommer in. Libnet skapar ett API (Application Programmer Interface) mot olika nätverksprotokoll, däribland POP3 som används här.

USER	Användarnamn
PASS	Lösenord
LIST [" " msg]	listar msg:es och storlek(octets)
RETR msg	hämtar meddelande
TOP	returnerar header + i rader
DELE msg	Raderar meddelandet
RSET	ångra radera
NOOP	kolla om servern lever
QUIT	avsluta session

Tabell 9 , POP3-kommandon

När meddelandena väl är lagade i databasen, är det härifrån de kommer kunna läsas via mobiltelefonen och webbläsaren.



Figur 12, projektets arkitektur

4.7 Svårigheter

Under utvecklingens gång uppkom vissa tekniska problem

4.7.1 För mycket information

Ett brevhuvud innehåller relativt mycket information, som kanske går an att läsa från en vanlig webbläsare på Internet. Från en mobiltelefon däremot, där skärmen är betydligt mindre, kan en del av denna information utelämnas till förmån för ett tydligare gränssnitt. Nedan visas ett exempel på all information som laddats ner från POP3-servern.

X-Auth-OK: daniel.wistrom@home.se Return-Path: Received: from daniel.wistrom@home.se [213.100.17.107] by home.se with NetMail ModWeb Module; Thu, 15 May 2003 21:18:17 +0200 **Subject:** Test..testar **From:** "mr D." To: daniel.wistrom@home.se **Date:** Thu, 15 May 2003 21:18:17 +0200 X-Mailer: NetMail ModWeb Module X-Sender: daniel.wistrom@home.se MIME-Version: 1.0 Message-ID: <1053026297.9772f560daniel.wistrom@home.se> Content-Type: text/plain; charset="ISO-8859-1" Content-Transfer-Encoding: quoted-printable TEST... testar lite bara...

De viktigaste fälten för normalanvändaren borde å andra sidan vara:

SUBJECT	Hejsan.. testar
FROM	daniel.wistrom@home.se
DATE	Thu, 15 May 2003-05-15
MSG	Detta är meddelandet... testar lite bara

Alla övriga fält borde egentligen rensas bort eftersom de tar onödigt mycket plats.

4.7.2 Få ut själva brevinnehållet

FROM och de övriga fälten som har en tagg, är lätta att plocka ut med en loop av formeln:

```
foreach $line (@lines)
      {
         if($line =~ m/^From: (.*)/)
         # om vi hittar  "From: "
         {
            $from = $1;
            # spara i variabeln $from
            $from =~ s/"|<.*>//g;

            # rensa upp strängen
            # max 40 tecken
                        $from = substr($from, 0, 39);
         } #if
             #...
      } # foreach
```

Själva meddelandet däremot, som inte föregås av någon specifik tagg är lite svårare att lokalisera bland resten av informationen.

4.7.3 Olika teckenuppsättningar och specialtecken

Olika teckenuppsättningar och diverse specialtecken kan skapa problem både då brevet ska sparas ned i databasen och när det ska visas i den mobila microbrowsern. Här kan en kontrollfunktion skapas som kontrollerar alla konstiga tecken och ersätter dessa med tillåtna. Det viktiga är att vara konsekvent med kodning av olika specialtecken så att dessa sedan kan avkodas i omvänd ordning där tex databasen och microbrowsern har inte klarar de olika standarderna.

Quoted-Printable gör tex om å, ä och ö till =E5, =F6 etc.
T.ex.: testar f=F6r andra g=E5ngen... (testar för andra gången)

HTML character entities ger ett format som kan bli problematiskt att lagra i databasen då de innehåller semikolon. Detta är något som an skapa problem vid SQL-kodning.
T.ex.: < (<), > (>)

Exempel på en rutin som tar bort dessa tecken kan se ut så här:

```
Foreach … {
        $contents =~s/</&lt;/g;
        $contents =~s/>/&gt;/g;
}
```

Denna funktion ersätter alla “<” och “>”-tecken med rätt kodning i WML.

4.7.4 Attachments

Vid mottagning av email med t.ex. ett Word-dokument som bilaga, skickades hela filen som en rad kryptiska tecken. Detta måste sorteras bort. Antingen borde filen kunna laddas ned (för de telefoner som stöder filformatet) eller borde den förkastas eller sparas på servern för åtkomst via webbgränssnittet.

5 Slutsatser

5.1 övergången till mobilt

Övergången från en Internetbaserad plattform till den mobila plattformen är inte helt enkel. Även om ett par tekniska svårigheter dök upp under projektets gång, är den stora utmaningen dock inte av teknisk karaktär. För företag som arbetar med Internetbaserade lösningar eller för ansvariga för Intranät, Internet etc. på ett företag torde den rent tekniska övergången ske utan betydande problem.

Utmaningen torde istället ligga i frågor som rör arkitektur, affärsnytta, säkerhet, användarvänlighet och kostnad. Det handlar nämligen inte enbart om att bygga upp ett system utan om eventuell införskaffning av ny hårdvara, nya mobiltelefoner, PDA:s till personalen som berörs av övergången samt utbildning. Dessutom, om det rör affärskritiska lösningar som hanterar viktig information, även säkerhet med kryptering etc.

En annan stor utmaning kan även vara den attitydförändring som krävs för att få användare på ett företag att acceptera ny teknik. Ericsson delade ju in användarna/tjänsterna i grupperna *Here, Now, Fun, Cool* beroende på tilltänkt målgrupp. Denna typ av metodik kan även användas för andra tjänster; det viktigaste är att lära sig förstå vad användarna värdesätter högst. Det rör alltså inte enbart utbildning utan att få anställda att känna sig bekväma med ett nytt informationsmedium. Datorn har ju trots allt haft ett par år på sig att förankra sig som arbetsredskap. Att tro att en mobiltelefon eller PDA gör samma sak i en handvändning är troligtvis ett stort misstag. Här borde dock de yngre, lite mer teknikmottagliga anställda ha lättare att ta till sig den nya tekniken. De passar också väl in i gruppen *cool*, vilka ser tjänsterna som ett sätt att höja egot genom att signalera t ex rikedom eller tillgång till det senaste.

5.2 Val av tillämpningar

Vilka typer av tillämpningar som kommer att sälja i framtiden beror i stor mån på i vilket syfte tillämpningen byggdes från början. För ett företag som kan låta sina anställda göra en del av det dagliga arbetet mellan olika kundmöten etc., finns säkert tid att spara – och därmed även pengar att tjäna. För tjänstelevererande företag som tänkt bygga upp en konsumentinriktad tjänst med tex mobiltelefonen som medium, bör nog stort fokus läggas på själva kundnyttan i första hand.

Här kan knytas an till Ericssons White Paper ”How to make a web site mobile – Extending web services with a successful mobile service” där olika tjänster delas in i grupperna Here, Now, Fun och Cool beroende på innehåll och vilka tänkta målgrupper som är intresserade av tjänsten.

Som ett exempel här kan nämnas företaget Eniro som tagit steget från den rent webbaserade telefonkatalogen till att nu även tillhandahålla tjänsten i ett mobilt gränssnitt. Användaren får upp en direktlänk till telefonnumret och behöver på så sätt alltså inte ens skriva in numret till den personen han vill nå. Användaren erhåller även en karta till den rätta platsen om han så önskar. Ett typexempel på ”Here” med andra ord.

Ett annat mycket bra exempel är SL:s wap-sida, på vilken användaren enkelt kan skriva in varifrån och vart han vill åka. Sidan räknar sedan (förvånansvärt snabbt) ut snabbaste resvägen, hur lång tid resan tar etc. Även detta ett exempel på en tjänst som skulle klassas som ”Here”.

Vilka typer av tjänster kan då tänkas nå den stora massan användare? Förutsatt att användarna tar till sig tekniken kommer troligtvis tjänster som innebär nytta, nöje eller tidsfördriv alltid att sälja. Tjänster som wap.sl.se och wap.eniro.se som nämndes tidigare innebär ju en stor tids- och kostnadsvinst jämfört med att ringa upp en telefonist för att få samma information. Tjänster som wap.svd.se, wap.aftonbladet.se och andra nyhetssajter innebär inte samma nytta, men kan bli nog så populärt som tidsfördriv eller rent informationssökande. Att kunna få realtidsuppdaterade börskurser eller väderprognoser kan också räknas till denna kategori. Även klassiska webbtjänster som email och chatt torde kunna bli stora framgångar eftersom dessa kanske mer än de andra riktar sig till en yngre, mer teknikmottaglig generation som redan anammat denna teknik i datormiljö.

5.3 Varför blev det ingen succé?

Att Wap inte blev någon succé eller än mindre dragplåster för försäljningen av mobiltelefoner har nog inte undgått någon. Vad beror det på? Tyvärr ligger nog begränsningen inte bara hos tjänsterna utan hos tekniken som används:

När första versionen av Wap kom var terminalerna små och svartvita och det fanns få tjänster som var intressanta. Det var svårt att övertyga kunder om varför de skulle använda Internet via mobiltelefonen. Dessutom kom Wap innan GPRS, vilket gjorde att användaren alltså fick betala en minuttaxa motsvarande ett vanligt samtal – dyrt med andra ord. Detta utgjorde ingen god start vid lanseringen av Wap, och vid tiden då version 1.1 kom hade nog de flesta användare som använder telefonen till samtal redan blivit bortskrämda av den nya tekniken. Tekniken kändes helt enkelt gammal redan då den kom. Dessutom stödde de olika mobila plattformarna olika delar av standarderna för mobilt Internet. Detta gjorde att en mobil lösning som passade vissa modeller av tex Nokia inte alls passade andra telefonmodeller. Om branschen hade enats snabbare om vilka tjänster som skulle implementeras och presenterat dessa för användarna hade inte mobilt Internet behövt bli så ”svårförståligt” ur användarsynpunkt.

Ett annat stort hinder för Wap kan nog sägas ligga hos operatörerna också. Då en användare köper en telefon, vanligtvis subventionerat med ett visst abonnemang, finns inga inställningar förinställda i telefonen. För att den vanliga användaren ska kunna nå de mobila tjänsterna, måste denne alltså först ringa upp sin mobiloperatör och be om de

inställningar, tex IP-adress till Wap-Gateway, som sedan ska matas in i telefonens ”kontrollpanel”. Inget för Svensson-användaren alltså. Sedan kunde dessa uppgifter dock sms:as till telefonen vid förfrågan, men det var fortfarande en stor målgrupp som troligtvis gick förlorad på grund av detta.

Nu – flera år efter första versionen av Wap – går det att köpa telefoner som är förinställda med operatörens inställningar och ofta även en direktknapp på telefonens panel som går till operatörens tjänsteportal. Detta får dock anses som ett steg i rätt riktning.

Andra hinder som kan tänkas ligga i vägen för utvecklingen av mobilt Internet är arkitekturen. Användaren måste till exempel gå via mobiloperatörens Wap-Gateway innan denne kan få tillgång till några tjänster. Detta är helt klart en flaskhals för Internettrafiken, och risken att dessa blir överbelastade är stor. Dessutom har historien visat att en mer ”demokratisk” uppbyggd arkitektur, som Internet har, skapar flexiblare lösningar, större konkurrens och därmed bättre tjänster.

Som parallell kan nämnas Internet och Prestel i Storbritannien eller Teletel i frankrike. Både Prestel och Teletel kan sägas vara föregångare till Internet, dock mer spartansk utseende som ej stödde rullningsbara lister (scrolling), hypertextlänkar eller avancerad grafik. Dessutom var användarna tvungna att köpa speciella terminaler för dessa. Prestel blev dock ett stort misslyckande medan Teletel blev en succé. Vad berodde detta på?

British Telecoms Prestel förblev en relativt dyr tjänst som få hade råd med och misslyckades därmed nå den stora massan användare som kunde ha pressat priserna. Därmed var det få som ville investera tid och pengar i interaktiva och dagligen uppdaterade sidor. Ett annat problem med Prestel var att om användaren inte redan hade en dator som kunde hantera videotex (textformatet) var alternativet att köpa en speciell prestel-terminal och koppla upp den till TV:n.

Minitel å andra sidan, som även den byggde på samma teknik som Prestel, subventionerades av det franska telefonbolaget och erbjöds varje telefonanvändare istället för en konventionell telefonbok. Kort därefter öppnades systemet för privata tjänsteleverantörer, först de stora nyhetstidningarna som var snabba med att popularisera det nya mediet. Därefter dök ett antal privata ”chattlinjer” upp (oftast med sexuellt innehåll). Minitels succé var ett faktum.

De slutsatser och paralleller till mobilt Internet som kan göras är:

- Systemet måste vara öppet för externa tjänsteleverantörer för att det ska bli en framgång. Både för att skapa konkurrens och mångsidighet bland tjänsterna.

- En kritisk massa av användare måste nås för att priserna ska kunna pressas och externa tjänsteleverantörer ska lockas investera i projektet.

- Systemet bör leverera användarna möjligheten att tillfredställa sitt kommunikationsbehov utan censur eller förmyndarmentalitet.

5.4 WAP:s Framtid

Troligtvis kommer den mobila revolution komma – om än senare än de flesta hade trott. Utvecklingen kommer antagligen att göra en vändning mot mer operatörsberoende tjänster, såsom portaler etc. – tjänster som enbart abonnenterna har tillgång till – och alltså ett led i att värva fler kunder. Dessutom kommer nog 3G göra att fler får upp ögonen för mobila tjänster, även om 3G än så länge har sålt dåligt i Sverige. Dessutom kommer nog högre konkurrens leda till lägre priser, vilket i sin tur kan locka fler kunder.

WAP som standard kommer bli mer moduluppbyggt och därmed lättare för externa tjänsteleverantörer att hantera. Det lite krångligare och mer begränsade WML kommer att försvinna helt – till förmån för XHTML, CSS och JavaScript. Utvecklingen går alltså mot mer Internetbaserad utveckling och därmed ska det vara lättare att konvertera existerande Internetlösningar till mobila lösningar.

6 Referenser

Litteratur:

Boumphrey et Al, *Börja med XHTML*, Sundbyberg, Pagina, 2000

Ewert, Magnus, *WAP : ett steg mot framtiden,* Lund, Studentlitteratur, 2000.

Ewert, Magnus, *GPRS,* Lund, Studentlitteratur, 2001.

Dreamtech Software Team, *WAP, Bluetooth, and 3G programming : cracking the code* New York, N.Y. , Hungry minds, 2002.

Medinets, David, Perl by example**,** Indianapolis, IN : Que, 1996

Stein, Lincoln, Official guide to programming with CGI.pm, New York : Wiley, Cop. 1998

Wall, Larry, Programming Perl, Cambridge, Mass. : O'Reilly, 2000

White Papers & andra för denna uppsats viktiga dokument:

Developers Guidelines – Web Browser, Augusti 2003 (Sony Ericsson)

2001 by Wireless Application Protocol Forum Ltd.- WAP 2.0 Technical White Paper

WAP-191-WML-20000219 (Wapforum.org)

Följande sidor gett information av mer teknisk karaktär:

http://www.wapforum.org (2004-02)
http://www.ericsson.com (2004-02)
http://www.nokia.com (2004-02)
http://www.wapsilon.com (2004-02)
http://www.w3schools.com/wap/wml_reference.asp (2004-02)
http://www.3tl.com/home.asp (2004-02)
http://www.sonyericsson.com (2004-02)
http://www.oma.com (2004-02)
http://public.research.mimesweeper.com (2004-02)

7 Bilagor

Appendix A, WTA URI och WMLScript Function Libraries

Public WTA

Public WTAI

Lib/Func ID	*URI*	*Script call*	*Description*
512.0	`wtai://wp/mc`	`WTAPublic.makeCall`	Make a call

Network Common WTA

Call Control

Lib/Func ID	*URI*	*WMLScript call*	*Description*
513.0	`wtai://cc/sc`	`WTACallCont.setup`	Setup a new call
513.1	`wtai://cc/ac`	`WTACallCont.accept`	Accept an incoming call
513.2	`wtai://cc/rc`	`WTACallCont.release`	Release a call
513.3	`wtai://cc/sd`	`WTACallCont.sendDTMF`	Send DTMF

Network Text

Lib/Func ID	*URI*	*WMLScript call*	*Description*
514.0	-	`WTANetText.send`	Send network text
514.1	-	`WTANetText.read`	Read network text
514.2	-	`WTANetText.delete`	Delete network text
514.3	-	`WTANetText.getFieldValue`	Get Field Value

Phonebook

Lib/Func ID	*URI*	*WMLScript call*	*Description*
515.0	-	`WTAPhoneBook.write`	Write phonebook entry
515.1	-	`WTAPhoneBook.read`	Read phonebook entry
515.2	-	`WTAPhoneBook.delete`	Delete phonebook entry
515.3	-	`WTAPhoneBook.getFieldValue`	Get Field Value

Miscellaneous

Lib/Func ID	*URI*	*WMLScript call*	*Description*
516.1	-	`WTAMisc.indication`	Logical Indications
516.2	`wtai://ms/ec`	`WTAMisc.endcontext`	Terminates user agent context

Appendix B, tekniskt stöd hos Sony Ericsson

Följande tabell visar vilka telefonmodeller som stöder vilka funktioner hos Sony Ericssons telefoner. (www.ericsson.com)

Model	Version	WML	XHTML	IHTML	CSS	DRM Forward-lock	OMA Download
R320	R1a	Yes					
R520	R202	Yes					
T39	R202	Yes					
T68	R201	Yes	Yes				
	R301*	Yes	Yes	Yes			
	R401*	Yes	Yes	Yes	Yes	Yes	
	R402*	Yes	Yes	Yes	Yes	Yes	
	R501	Yes	Yes		Yes	Yes	
	R502	Yes	Yes		Yes	Yes	
	R503**	Yes	Yes		Yes	Yes	
T300	R101	Yes	Yes				
T302**							
T306*	R201	Yes	Yes			Yes	
T310	R201	Yes	Yes			yes	
T312							
T610	R101	Yes	Yes	Yes	Yes	Yes	Yes
T618**							
Z600							

*) US market.

**) Chinese market.

Appendix C , Media typer

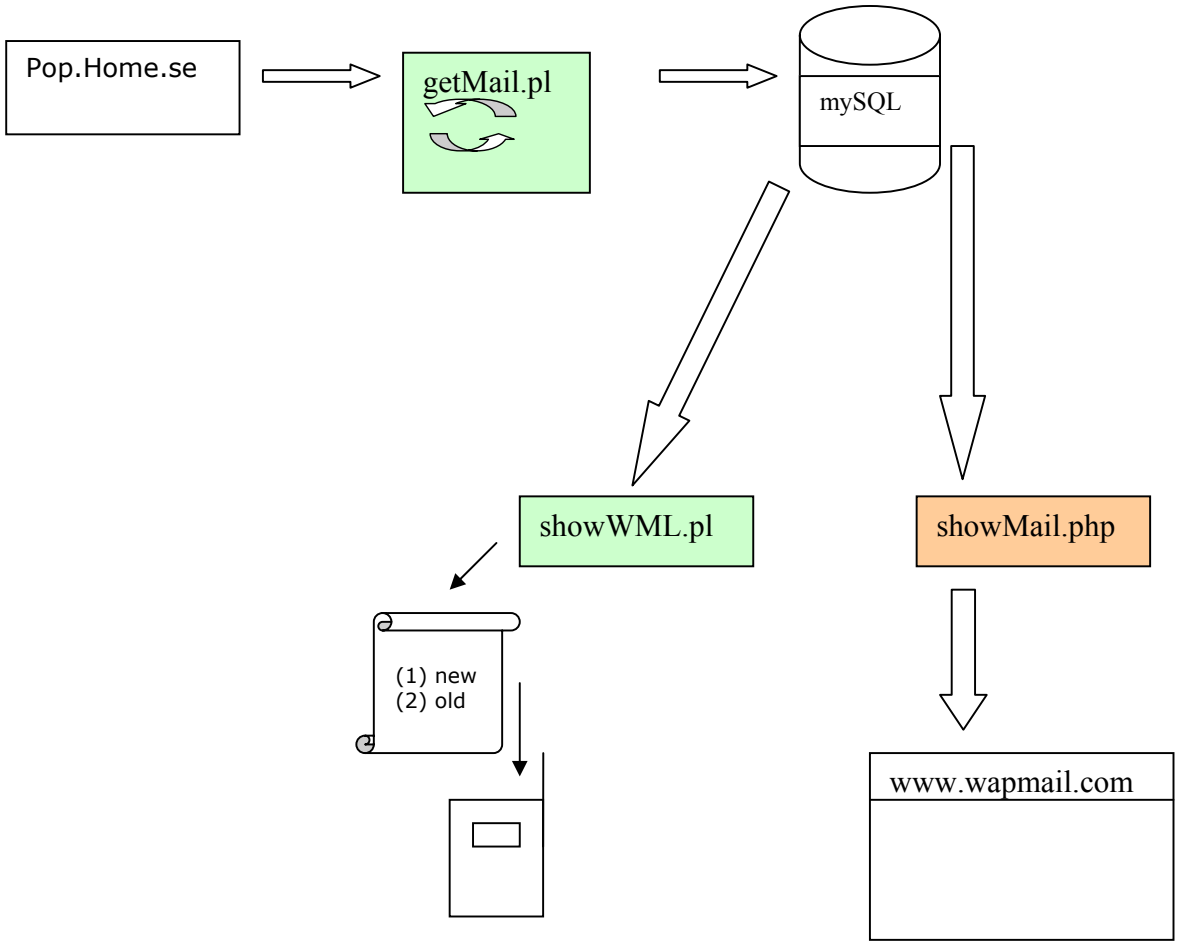
Datatyp	MIME mediatyp	filändelse
WML1 textform	text/vnd.wap.wml	.wml
WML1 binär form	application/vnd.wap.wmlc	.wmlc
WML2	application/vnd.wap.wml+xml	.wml
XHTML Basic	application/xhtml+xml text/html	.xhtml, .xht
XHTML Mobile Profile	application/vnd.wap.xhtml+xml application/xhtml+xml; profile=http://www.wapforum.org/xhtml1 text/html	.xhtml, .xht
WAP CSS	text/css	.css
WMLScript textform	text/vnd.wap.wmlscript	.wmls
WMLScript binär form	application/vnd.wap.wmlscriptc	.wmlsc
WBXML	application/vnd.wap.wbxml	
WBMP	image/vnd.wap.wbmp	.wbmp
vCard	text/x-vCard	.vcf
vCalendar	text/x-vCalendar	.vcs
WTA-WML textform	text/x-wap-wta-wml	
WTA-WML binär form	application/x-wap-wta-wmlc	
Textform av Multipart-meddelanden som används delarna i meddelandets body är oberoende och bör följas i specifik ordning	multipart/mixed	

Appendix D, Ordlista

3G	Third Generation, Tredje generationens mobiltelefoni.
API	Application Programmer Interface
Apache	En av de mest spridda Webbservrarna idag.
ASP	Active Server Pages - Ett vanligt scriptspråk på Internet idag.
CSS	Cascading Style Sheet, Externt dokument innehållande själva formateringen av dokumentet.
DTD	Document Type Definition. Ett dokument om regler för hur syntaxen ska tolkas
DTMF	Ett antal toner telefonen kan sända för tex direktkoppling i telefonväxel.
Emulator	Ett program som simulerar en mobiltelefon eller PDA.
GPRS	General Package Radio Service
GSM	Global System for Mobile Communications
HDTP	Unwired Planet skapade ett koncept som hette HDML (Handheld Device Markup Language) samt HDTP (Handheld Device Transport Protocol). Föregångare till WAP
HTML	HyperText Markup Language
MIME	Multipurpose Internet Mail Extensions
MMS	Multimedia Messages Service
MySQL	En av de mest spridda databashanterarna idag.
NMIT	Nokia mobile Internet toolkit.
PDA	Personal Digital Assistant
Pictogram	I dagligt tal kallat ”Smiley”
Perl	Ett vanligt scriptspråk på Internet idag
PHP	Ett vanligt scriptspråk på Internet idag
POP3	Protocol
Scriptmotor	Applikation som gör att webbserverna kan hantera olika scriptspråk
SGML	Standard Generalized Markup Language, som togs fram för seriöst dokumentarbete och har varit en internationell standard för hur man märker en text sedan 1986
SMS	Short Message Service
SSL	Secure Socket Layer

Tagg	Kodsnutt som anger en viss formatering tex. <b>
tel – scheme	tel-schemat specificerar ett telefonnummer. Tex Ring
TLS	Transport Layer Security
TTML	Tagged Text Markup Language, 16
URL	Uniform Resource Locator
vCard, vCalendar	Ett format för att kunna bearbeta telefonens telefonbok och kalender.
WAE	Wireless Application Environment
WDP	Wireless Datagram Protocol
WAP	Wireless Application Protocol
WBMP	Wireless Bitmap
WBXML	En binär representation av XML som har designats för att reducera den mängd data som skickas över nätet. Här kan en enda byte betyda t.ex. <deck>, vilket går snabbare och är mer kostnadseffektivt.
WML	Wireless Markup Language
WMLScript	Wireless Markup Language Script
WSP	Wireless Session Protocol
WTA	Wireless Telephony Applications
WTAI	Wireless Telephony Applications Interface
WTLS	Wireless Transport Layer Security
WTP	Wireless Transport Protocol
XHTML	eXstensible HyperText Markup Language
XML	eXtensible Markup Language

Appendix E, Projektets flödesschema



7 Källkod

Index.wml – ett exempel

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<head>
      <meta forua="true" http-equiv="Cache-Control" content="max-
age=0"/>
</head>

<card id="card1" title="WapMail - Login">
<onevent type="onenterforward">
      <refresh>
            <setvar name="user" value="" />
            <setvar name="pwd" value="" />
      </refresh>
</onevent>


<p>

user:     <input name="user" value="" maxlength="40" />
password: <input name="pwd"  value="" maxlength="20" type="password" />
<br />

<anchor title="Send">go! <go href="http://127.0.0.1/cgi-
bin/wap_mail/wap/cell_login.pl" method="post">
            <postfield name="user" value="$(user:noesc)"/>
            <postfield name="pwd" value="$(pwd:noesc)"/>


      </go>
</anchor>
</p>


</card>
</wml>
```