Kan samma XML stödja både HTML och VoiceXML?

En studie i voice browsing och VoiceXML

Nordström, Linda; Andersson, Åsa

2001

Kan samma XML stödja både HTML och VoiceXML?

- **En studie i voice browsing och VoiceXML**

Magisteruppsats, 10 poäng, vid Institutionen för Informatik

Framlagd: April, 2001
Författare: Åsa Andersson
Linda Nordström
Handledare: Michael Svedemar

Innehållsförteckning

1 Introduktion

1.1 Inledning

I dagens moderna samhälle växer behovet och kraven på mobilitet och platsoberoende. Företag är utspridda nationellt och internationellt, vilket innebär att människor reser mycket och rör sig över stora geografiska områden. Detta ökar behovet av tillgång till information var man än befinner sig, även i situationer där man ej har tillgång till traditionell internetuppkoppling.

Ideligen sker förändringar, man vill förbättra, förenkla, förnya, bredda, kombinera och etablera nya möjligheter. Utvecklingen av nya elektroniska kanaler, verktyg och presentationstekniker bryter ständigt nya marker. Det börjar dyka upp alternativa sätt att få tillgång till information och innehåll på internet än via den traditionella webbläsaren. Nu börjar det talas om ett nytt koncept, så kallad *voice browsing*. Voice browsing innebär att rösten används som indata och syntetiskt tal fås som svar.

I och med de röststyrda mobiltelefonerna men också de röststyrda telefontjänster som blir alltmer vanligt förekommande för bokningssystem vid till exempel biljettköp, är numer även voice browsing en realitet. Infrastrukturen finns med telefoni och internet och därtill även fungerande teknik i form av nya standarder. Människor kommer alltmer att använda sig av rösten som navigeringsinstrument för olika företeelser, då inte bara för internet och webbsurfning utan även för till exempel affärstransaktioner i affärssystem.

En anledning till att internet har blivit så stort är bland annat webbläsarens utveckling och då denna blev gemene mans tillgång. Voice browsing har liknande möjligheter att bli stort tack vare telefonins utbredning. Ytterligare anledningar att voice browsing har förutsättningar att bli stort, är WAP-teknikens begränsningar med små displayer och mycket tidsödande knapptryckande. För handdatorer finns även andra hinder, bland annat gränsen för hur små tangentborden kan bli. Här kan då rösten bli ett attraktivt alternativt inmatningssätt.

Tal är den mest naturliga kommunikationsformen för människor (Andleigh & Thakrar 1996, Parfitt 2000), därmed också den dominerande (Fluckiger 1995). Tal hindrar heller inte händer eller blick, vilket är en klar fördel då man behöver och vill utföra flera aktiviteter simultant, som att ringa samtidigt som man har händerna fulla. Ett exempel är under resor, när

händerna ofta är upptagna av väskor och packning och ögonen riktade mot omgivningen. Vill man då samtidigt försöka använda händerna till att knappa in rätt tangenter på en handdator eller mobiltelefon för att se menyer och sökt information rullas upp på en display, får man som resande en hel del omak. Rörligheten reduceras genast. Man behöver inte ens dra exemplet så långt, det räcker med att försöka gå samtidigt som man stirrar ner på sin handdator för att kunna förnimma en viss möda. Kan man då istället tala in och få svar kommer man ifrån de vedermödorna, det blir smidigare och uppmärksamheten kan kvarhållas på omgivningen.

Därmed inte sagt att voice browsing helt kommer att ersätta den teknik vars navigering sker via knappar och där information presenteras på displayer. Troligt är istället att röstteknologier ofta kommer fungera som komplement till befintlig teknik. Ett exempel kan vara för just WAP, vars verktyg presenterar informationen på displayer, men där ett mer eller mindre omständligt stegande i menyer via tangentbord eller pekskärm, skulle kunna underlättas genom att använda rösten som indata.

1.2 Begreppsdefinition

Vi väljer att diskutera begreppsdefinitioner här då det gäller ett så pass nytt koncept och ny teknik. Detta för att vi först vill förklara begreppen för att ge en bakgrund inför problemdefinitionen. Vidare fördjupning av XML respektive VoiceXML återfinns i kapitel 2. Det finns även en ordlista i slutet där fler begrepp och förkortningar närmare förklaras.

1.2.1.1 Voice browsing

Voice browsing kan man säga består av två delar. Dels handlar det om att användarna kan använda tal som indata (istället för att till exempel peka och klicka) och dels att innehållet på webbsidorna, e-posten eller affärssystemet läses upp (ger svar) med konstgjort tal.

1.2.1.2 XML

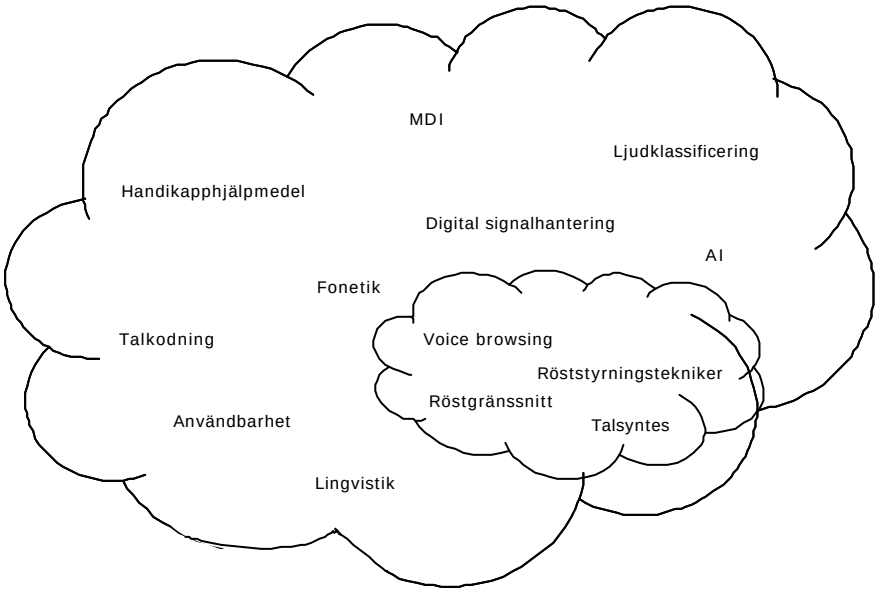
eXtensible Markup Language (XML) är ett *utbyggbart märkspråk*, egentligen en standard för hur man *skapar* märkspråk. Märkspråk innebär att speciella märkord eller taggar används för märkning i dokument, enligt Svenska datatermgruppen (2000). XML är en förenklad variant av *Standard Generalised Markup Language (SGML)* och anpassad för användning på internet.

1.2.1.3 VoiceXML

Voice eXtensible Markup Language (VoiceXML) är ett märkspråk vilket designades för att göra information på internet tillgängligt via röst och telefon (VoiceXML Forum 2000a). Språket är även en ny standard, vilken antogs av World Wide Web Consortium (W3C) i maj 2000 (Parfitt 2000). W3C är en organisation vilken försöker främja användandet av internationella standarder vid mjukvaruutveckling av internetprodukter. VoiceXML är designat för att skapa audiodialoger som tillhandahåller funktioner såsom syntetisk röst, digitalt ljud, igenkänning av tal- och tonvalsinmatning, inspelning av röst med mera (VoiceXML Forum 2000b).

1.3 Problemområde

I figur 1.1 har vi försökt visa ett antal variabler som alla kan ses som ett eget ämne, men som samtidigt kan tillhöra samma genre genom att adderas med eller appliceras på varandra. Voice browsing omfattar en stor mängd andra relaterade områden vars inverkan och påverkan tydligt kan prägla varandra. Tanken är att bilden, med nämnda variabler placerade i ett moln, ska försöka bidra till att positionera voice browsing inom ett område med relaterade attribut. Variablernas kopplingar samt paralleller till voice browsing kan vara mer eller mindre påtagliga. Genom att använda molnet som form vill vi också visa på ämnets storlek och otydliga gränser.



Figur 1.1. Olika ämnen och områden vilka angränsar till voice browsing (egen referens).

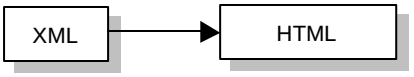
Tillsammans med relaterade ämnen kan problemområdet runt voice browsing bli hur stort som helst. Storleken bidrar i sin tur till att ämnets komplexitet kan utökas i det oändliga och därmed även uppsatsens omfång. Det är lätt att falla in på olika sidospår och sväva ut i intressanta diskussioner om relaterade ämnen, dock förloras lätt den röda tråd som ska följa genom hela uppsatsen. För att inte få en uppsats vars problemområde drar och stretar åt för många olika håll, har vi valt att lägga uppsatsens fokus på voice browsing och inte närmare beröra andra relaterade ämnen.

Nämnda variabler som till exempel lingvistik, talkodning och röststyrningstekniker, ska inte på något sett förringas, de flesta har alla en central del vid voice browsing, många av dem är direkt essentiella. De är dock inte aktuella i det skede vår uppsats innefattar, utan i ett senare led, till exempel vid en vidareutveckling av det arv vår uppsats lämnar efter sig.

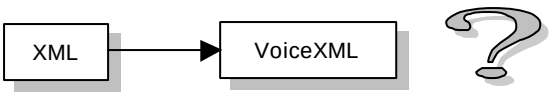
Det ska påtalas att återgivna variabler inom uppritat moln är exempel på variabler vi funnit nämnda i samband med voice browsing i den litteratur vi använt. Dessa variabler är inte på något sätt de absoluta och enda för olika relationer inom ämnet. Det finns säkerligen andra än de vi tagit upp och fler kommer högst sannolikt dyka upp med utvecklingen. Vidare har den inbördes positioneringen av variablerna ingen direkt betydelse utan är, av oss, slumpmässigt utplacerade. I det inre mindre molnet finns variabler vars samband oftare nämns inom den litteratur vi tittat på, det vill säga de ligger lite närmare voice browsing än de övriga.

1.3.1 Implementering av VoiceXML med XML

Grundtanken är att från start försöka utnyttja de fördelar XML-filer besitter (se kapitel 2.2.5), så även vid en implementation av VoiceXML. Kombinationen med XML och HTML, vilken illustreras i figur 1.2, vet vi fungerar, dels för att XML bland annat utvecklades för att göra HTML dynamiskt. Man ville gå ifrån webbdokument med statiskt innehåll till aktiva interaktioner. Med andra ord behövdes något utöver HTML för att få dynamiska webbsidor (Nuance 2000).

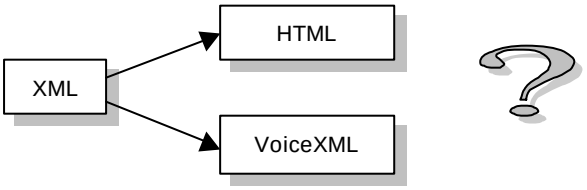


Figur 1.2. Kombinationen XML och HTML fungerar.



Figur 1.3. Kommer kombinationen XML och VoiceXML att fungera?

Vad vi undrar är om XML fungerar med VoiceXML (se figur 1.3). Det är möjligt att andra fördelar finns eller skapas om kombinationen XML och VoiceXML fungerar. En möjlighet vi är väldigt intresserade av är om *samma* XML-fil kan fungera med både HTML och VoiceXML (se figur 1.4).



Figur 1.4. Kan samma XML stödja både HTML och VoiceXML?

1.3.2 Forskningsfrågor

Sammanfattningsvis avser vi att finna svaren på de två relaterade frågorna: Fungerar kombinationen XML och VoiceXML? Om detta fungerar blir följdfrågan; kan samma XML-fil som används för HTML lika väl användas för VoiceXML?

1.4 Syfte

Voice browsing väckte vårt intresse mycket tack vare att det är ett spännande och relativt nytt ämne. Intresset att titta lite närmare på voice browsing förde oss vidare till ämnet för denna uppsats. Då vi började vår praktik på Sync Business Process Integration AB (Sync) kom vi i kontakt med den nya standarden VoiceXML. Vi tyckte att tekniken verkade intressant och vi fick en möjlighet att utföra undersökningen hos Sync som en del av vår praktik. Att voice browsing kan stödjas med VoiceXML vet vi redan. Huvudsyftet för uppsatsen är istället att undersöka kompatibiliteten mellan XML och VoiceXML och vid en positiv respons, även undersöka om samma XML som används vid HTML också kan användas för VoiceXML. I det senare fallet skulle mycket vara vunnet vid en framtida implementering av VoiceXML i befintlig systemarkitektur på Sync, eftersom arbetet underlättas av att XML-filen ej behöver ändras. Datan är den samma, det är endast själva presentationsättet, vilket i dagsläget är webbaserat, som behöver förändras för att stödja även röstgränssnitt. Givetvis är ett delsyfte att vi kommer ur denna erfarenhet något klokare och mer bevandrade inom ämnet voice browsing med VoiceXML. Med andra ord, några lärdomar rikare att ta med i bagaget ut i arbetslivet.

1.5 Avgränsningar

När det gäller uppsatsens fokus har enbart de tekniska bitarna tagits upp rörande implementering av VoiceXML. Ingen tid har lagts på att undersöka lingvistik, fonetik och ljudklassificering. Ej heller utvärderingar av andra angränsande tekniker, metodiker och beröringspunkter rörande till exempel röstgränssnitt, talsyntes eller röststyrningstekniker har gjorts.

Endast tekniken för voice browsing har studerats, det vill säga VoiceXML. Varför just VoiceXML valdes beror på att det är en ny W3C-standard. Dessutom finns ytterligare en intressant anledning till varför vi valde VoiceXML och det är att även XML är en W3C-standard.

Som taltolkningsmiljö valdes IBM:s mjukvara ViaVoice TTS 5.0 SDK, vilken är den enda som använts. ViaVoice valdes dels på grund av att det är en IBM-produkt (Sync följer IBM:s *E-business framework* och därmed passar produkten in i utvecklingsmiljön) och dels efter att ha tagit del av en utvärdering, där ViaVoice framstod som den bäst lämpade (Keizer 2000).

Vi har utvecklat en prototyp för en väldigt liten och mycket avgränsad del av systemet. Inga användare av något slag deltog då det endast var en undersökning av teknisk natur. Tonvikt har lagts på att få undersökningens huvuddelar att fungera tillsammans i prototypen. Inga användartester eller undersökningar har genomförts angående användbarhet, processflöde vid prototypkörning eller dylikt.

1.6 Målgrupp

Uppsatsen riktar sig först och främst till studenter vars kunskapsbakgrund liknar systemvetenskapliga och datalogiska utbildningar. En fördel är att läsaren innehar en viss kunskap i märkspråk som till exempel HTML. Uppsatsen kan även tilltala andra människor med intresse för voice browsing eller informationsteknologi i allmänhet.

1.7 Val av termer

Voice browsing – Vi har valt att använda det engelska uttrycket ”voice browsing” då det, enligt oss, ej finns någon bra motsvarighet på svenska. Vi har varit i kontakt med Svenska datatermgruppen samt Svenska språknämnden (Ola Karlsson), men de hade inte några bra förslag. Anders Lotsson (journalist på Computer Sweden som också sitter med i Svenska

datatermgruppen) kände inte heller till någon bra översättning. Han svarade oss:

> *”[…] jag har inte satt mig in i VoiceXML, men om jag förstått det rätt handlar det i första hand om att användarna ska kunna prata i stället för att peka och klicka, i andra hand om att även innehållet på webbsidorna ska läsas upp med konstgjort tal. Röstnavigering täcker det första, tycker jag, men inte det andra. Röstsurfning skulle kanske duga, om det inte är för informellt.”*

Problematiken består alltså främst i att finna *ett* motsvarande ord.

Internet – Vi har valt att skriva internet med litet ”i”, detta efter att ha läst artikeln *Internet är som solen* (Lotsson 2000). Artikeln skrevs för Computer Sweden (CS) och tog upp läsarfrågor rörande varför journalister på CS skrev internet med litet i, när de tidigare propagerat för en stavning med versalt i.

Namnet *Internet* registrerades som varumärke, vilket också är anledningen till att det från början skrevs med stort i. Internet var ett egennamn. Numera menar Lotsson (2000) att internet har slutat vara ett egennamn och ger exempel på ord som dynamit, grammofon, masonit och makadam som är egennamn, eller egentligen varumärken som förvandlats till substantiv.

Det finns bara ett internet, som är en infrastruktur liksom järnvägen, posten och flyget vilka skrivs med liten begynnelsebokstav. *”Man säger ibland att det betraktas som en självklar del av språksamhällets liv*” (Lotsson 2000).

Motiveringen blir därmed att internet uppfyller båda kriterierna som diskuterats ovan (Lotsson 2000): *”(1) det används inte längre som ett egennamn och (2) det betraktas som en självklarhet. Det är som med solen. Vi räknar den till infrastrukturen och skriver ordet med litet s*.”

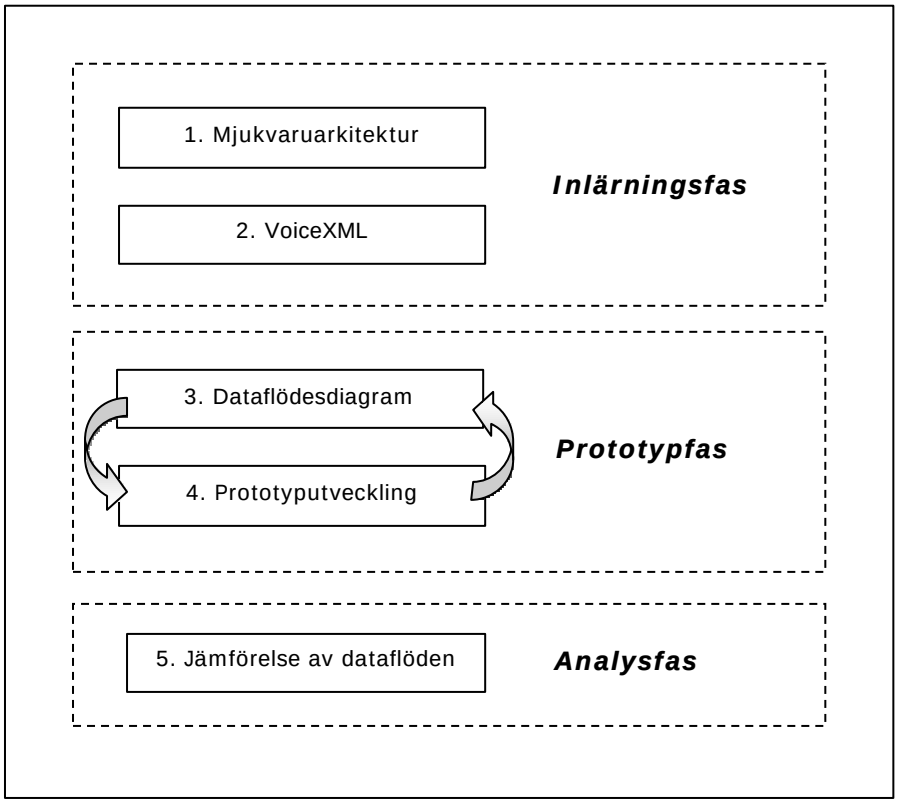
Märkspråk – Det finns många varianter på översättningen av det engelska *markup language* (till exempel uppmärkningsspråk, märkordspråk, taggspråk), vilket gjort det svårt att hitta en konsekvent svensk benämning. Vi har dock valt att följa Svenska datatermgruppens (2000) rekommendationer och kommer genomgående att använda termen *märkspråk*.

1.8 Metod

1.8.1 Metodmodell

Vår metod gick ut på att först få en förståelse för mjukvaruarkitekturen på Sync, sedan i tekniken för VoiceXML, för att på slutet se om, och hur, de två var möjliga att synkronisera.

Vi delade in vår metod i tre övergripande faser: inlärningsfas, prototypfas och analysfas (se figur 1.5).



Figur 1.5. Metodmodell för den empiriska undersökningen (egen referens).

1.8.1.1 Inlärningsfas

1. Denna fas delas upp i ytterligare två delar. Den första delen innebar att vi satte oss in i *mjukvaruarkitekturen* och datadialogen på Sync samt lärde oss deras arbetssätt för gränssnittsutveckling med mockups, dataflödesdiagram, HTML, XSL, XML och JSP. Detta arbetssätt är en del av Syncs metod för systemutveckling som heter SPINE (*Secure Parallell Iterative Network based Evolution*), vilken är en egenutvecklad

implementation av RUP-metoden (*Rational Unified Process,* Rational Software Corporation 2000).

2. Den andra delen innebar att vi lärde oss hur *VoiceXML* fungerar, hur det är uppbyggt, hur egna VoiceXML-applikationer (programmeringsspråk och syntax) utvecklas och så vidare. Denna information inhämtades främst på internet; på VoiceXML Forums (2000b), W3C:s (2000b) och Nuances (2000) webbsidor.

1.8.1.2 Prototypfas

I prototypfasen, vilken är den mer praktiska delen av vår metod, applicerade vi Syncs arbetsmetod för gränssnittsutveckling. Vi valde denna metod då den är inarbetad på Sync och fungerar väl för denna typ av utveckling.

3. Prototypfasen i sig bestod också av två delar, där första delen var litet av en planeringsfas där vi dels skissade mockups över dialogen och dels dokumenterade flödet i datastrukturen genom att utarbeta *dataflödesdiagram*.

   Mockups utgör enklare gränssnittsskisser som avser att på ett tydligt sätt visa hur det blivande användargränssnittet är tänkt att se ut (Löwgren & Stolterman 1998). Syftet med mockups kan variera och tack vare deras flexibilitet går det att uttrycka mycket med dem. Exempelvis kan de fungera som verktyg i designfasen att kommunicera med användaren hur gränssnittet ska se ut. De kan även användas vid kravspecifikationen för att finna användarbehoven (Löwgren & Stolterman 1998). Mockups fungerar även utmärkt för att testa ett systems funktionalitet och användbarhet innan det verkliga systemet byggs (Rational Software Corporation 2000). När Löwgren och Stolterman (1998) talar om gränssnittsskisser avser de skisser på papper. De finner nackdelar med denna metod, dels att det är arbetssamt att revidera skisserna och dels att systemets tänkta dynamik inte illustreras.

   Det finns även andra verktyg att tillgå. RUP-metoden (Rational Software Corporation 2000) nämner förutom pappersskisser även elektroniska ritverktyg (*bitmaps* på engelska) och interaktiva gränssnitt (*executables* på engelska). För vår undersökning använde vi oss utav pappersskisser vilka sedan renritades med hjälp av ritverktyget Microsoft Powerpoint. På Sync brukas interaktiva gränssnitt, vilka internt kallas ”dumma” mockups. Dessa är implementerade i HTML och visar främst gränssnittets utseende men även viss funktionalitet såsom navigering mellan sidorna. Varför de kallas dumma är för att de ligger helt utanför det verkliga systemet och därmed inte presenterar någon riktig data. Denna typ av mockups var ej aktuell i vår undersökning eftersom de inte skulle tillföra mer än pappersskisserna. Det vi avsåg

studera innefattade en liten och avgränsad del av systemet samt en hårt styrd dialog med endast en väg att gå utan valmöjligeter. Om det hade varit en större del av systemet med en mer komplex dialog som avsågs att undersökas, hade en interaktiv mockup varit mer intressant, då man på ett tydligt sätt kunnat redovisa gränssnittets olika valmöjligheter.

Syftet med våra mockups var att på ett tydligt sätt visa människa-dator-dialogen. Det var inte intressant att redogöra för hur gränssnittet exakt skulle komma att se ut, utan avsikten var att kommunicera testdialogens händelseförlopp.

Dataflödesdiagram beskriver hur information skickas i ett system. De visar på ett tydligt sätt hur datan passerar in och ut genom processer (Avison & Fitzgerald 1988, Preece et al 1994). Dataflödena visar också varifrån informationen kommer och vart den går (Andersen 1991). Diagrammen besitter förmågan att med samma notation visa både systemets processer och de mänskliga processerna, exempelvis representeras indata från användare respektive system på samma sätt (Preece et al 1994).

I vår undersökning utvecklades dataflödesdiagrammen för att få en överskådlighet av de olika komponenterna och dess relationer. Vi avsåg att på ett tydligt sätt visualisera strukturen samt dataflödet mellan de olika filerna. Vi utvecklade mockups och dataflödesdiagram för både HTML- och VoiceXML-lösningen. Detta för att erhålla ett bra jämförelsematerial inför analysen.

4. Den andra delen av fasen var av mer teknisk natur där vi utvecklade en *prototyp* i VoiceXML för att få en känsla för hur det fungerade i praktiken. Detta skedde iterativt och genom så kallad *trial-and-error* metod. Vi testade oss fram tills vi hittade en lösning som passade med XML samt med den existerande mjukvaruarkitekturen på Sync.

   VoiceXML-dataflödesdiagrammet samt prototypen utvecklades delvis iterativt, eftersom vi ej från början visste hur lösningen skulle komma att se ut. Dataflödesdiagrammet reviderades då efterhand i takt med prototyputvecklingen.

   Prototypen som utvecklades kommer ej att tas i bruk, den utvecklades som en test och kommer att *kastas bort*. Detta kallar Preece et al (1994) för *rapid prototyping*. Det är den inhämtade kunskapen och erfarenheten från utvecklingen och inte själva prototypen som är den viktiga slutprodukten. Både Alter (1996) och Turban och Aronson (1998) talar om en *kasta bort*-prototyp (*throwaway* på engelska) vars tanke är att uppnå en bättre förståelse för systemets prestanda och användarkrav. Med hjälp av prototypen ska det vara möjligt att testa idéer och

alternativ. När pilottestet genomförts ska prototypen kastas och en första design ta vid. Enligt Alter (1996) lämpar det sig att använda denna typ av prototyping vid jämförandet av alternativa designmöjligheter för delar av ett system.

Prototypen innehar full funktion men bara inom en liten avgränsad del av affärsapplikationen, vilket gör den till en så kallad *vertikal prototyp* (Preece et al 1994).

1.8.1.3 Analysfas

5. Analysen bestod främst i en jämförelse mellan de båda lösningarna i HTML respektive VoiceXML. Utifrån dataflödesdiagrammen diskuterades likheter och skillnader gällande dataflöde, uppbyggnad och struktur samt i förhållande till forskningsfrågorna. I samband med analysen utfördes även en kritisk granskning av det egna arbetet.

1.8.2 Datainsamling

Det finns ett visst problem med att välja ett så pass ungt ämne. Det fanns praktiskt taget inga publicerade böcker om voice browsing eller VoiceXML då vi sökte vår litteratur. Vi fann dock en hel del artiklar och information på internet. Vad det gäller XML och XSL fanns en del litteratur, men nästan enbart rena programmeringsböcker.

Förutom internet sökte vi information på bibliotek, i tidskrifter, dagspress och populär-vetenskapliga tidningar. Vi har även tittat på gamla uppsatser på Institutionen för Informatik gällande voice browsing och XML.

Genom praktiken på Sync parallellt med undersökningen inhämtades mycket av given information. Företaget fungerade som plattform för vår frågeställning och tester av VoiceXML.

Spontana och ostrukturerade samtal har kontinuerligt förts med vår handledare Carl-Johan Andersson på Sync. Genom dessa samtal har vi bland annat inhämtat information gällande Syncs mjukvaruarkitektur och arbetssätt. Andersson fungerar som teamleader för HCI-teamet på Sync, men även som teamleader och ansvarig för Strategic Development.

1.8.3 Källkritik

Vi ser en viss risk med tillförlitligheten på information vi funnit på internet. Det handlar ofta om icke-granskat material, speciellt på kommersiella hemsidor. Men även om informationen ej är akademiskt granskad har företag ofta civilrättsligt ansvar för det de lägger ut på sin webbsida. I vårt

fall handlar det dessutom om ganska stora och tunga företag och organisationer som till exempel Nuance, W3C och VoiceXML Forum, vilket, menar vi, ökar tillförlitligheten på materialet vi använt oss av.

Den ostrukturerade dialog vi haft med Carl-Johan Andersson har kanske inte varit den bästa datainsamlingsmetod ur akademiskt vetenskaplig synpunkt. Vi är medvetna om riskerna men de vägs upp av att tillvägagångssättet varit mycket praktiskt då vi haft konstant tillgänglighet till informationen.

1.9 Undersökningens kvalitet

1.9.1 Validitet

Innehållsvaliditet handlar om ifall alla aspekter av frågeställningen täckts in av den konkreta frågan (Svenning 1997). Utifrån de avgränsningar som vi gjort, tycker vi, att vi nått begränsade och undersökningsbara frågor att söka svar på.

Den inre validiteten handlar om ifall teori och empiri överensstämmer i sin kontext (Yin 1984). Kopplingen mellan den teoretiska och empiriska delen är erkänt svår, men utan en sådan koppling blir forskningen meningslös, enligt Svenning (1997). För att kunna stödja den inre validiteten krävs relevant och givande litteratur till teorin. Med andra ord, åligger det undersökaren att finna de fakta som erfordras för att styrka undersökningen och besvara ställda undersökningsfrågor. Då utbudet av publicerat material om VoiceXML än så länge är tämligen begränsat har det funnits vissa svårigheter att uppbringa erforderliga fakta. Vi tycker oss dock ha funnit tillräckligt med underlag för att ge en grund att stå på.

Den yttre validiteten handlar om projektet som helhet – om det finns möjligheter till generalisering (Svenning 1997). Det innebär att det ska vara möjligt att utifrån en specifik studie anta en allmän teori, följaktligen ska undersökningen i fråga vara representativ i anknytning till de urval som gjorts. Vår undersökning kan definitivt generaliseras till liknande situationer, det vill säga affärssystem som ska implementera VoiceXML i en liknande mjukvaruarkitektur som den på Sync. Det kan dock bli problem om mjukvaruarkitekturen ser annorlunda ut, möjligheten finns då att premisserna förändras radikalt. Samma argument kan även gälla för affärssystem som vill använda en annan teknik än VoiceXML vid implementation av voice browsing.

1.9.2 Reliabilitet

Med reliabilitet menas att en undersökning, med samma metoder och tillvägagångssätt, ska kunna göras om av någon annan och ändå resultera i att samma slutsatser dras (Yin 1994). En undersökning ska nå en så pass hög nivå att dess resultat kan anses vara tillförlitligt.

För denna uppsats undersökning, vars studie haft ett tydligt avgränsat område av teknisk karaktär, fanns inte så många alternativa lösningar till frågeställningen – antingen fungerar det eller så fungerar det inte. Med andra ord torde en ny undersökning vars premisser är de samma som för vår undersökning ge samma svar. Vi bedömer därför reliabiliteten säkrad, grundat på att dessa förbestämda distinkta variabler av teknisk natur inte kan ge annat utfall än svaret, ja det fungerade, eller nej det fungerade inte.

Enligt Yin (1994) kan man öka reliabiliteten genom att föra protokoll och eventuellt skapa en databas associerad med undersökningen. Skapandet av en databas gäller främst vid kvantitativa undersökningar då slutsatser ska deriveras ur en stor mängd kodad, analyserad och sammanfattad data. Databasen innehåller då de mätningar som görs, de mätvärden och metoder som används i analysen samt de slutsatser som dras under olika faser.

Att sätta upp en databas för vår undersökning har inte varit aktuellt. Utifrån våra bedömningar gjorde vi den resolutionen att arbetet med att sätta upp en dylik databas hade varit både tidskrävande och med föga egentliga förtjänster. Vi bedömde det dock som essentiellt att grundligt dokumentera vårt resultat både i skisser och förklarande text. Därför har vi använt oss av dokumentering som reliabilitetsstärkande verktyg. Tack vare dokumentationens detaljrikedom ökas reliabiliteten genom möjligheten att följa samt återupprepa undersökningen. Källkoden är även bifogad för den som eventuellt vill fördjupa sig i vår lösning.

1.10 Uppsatsens disposition

I kapitel två diskuteras XML och VoiceXML utifrån den litteratur vi studerat. Först ges en historisk bakgrund till webbens uppkomst, det vill säga behovet av ett forum för utbyte av information och multimediadokument samt sedermera även som en plattform för utveckling och formation av mjukvaruapplikationer. I kapitel två tas även XML och VoiceXML upp ur teoretisk synpunkt, bland annat vardera tekniks uppbyggnad och struktur, fördelar och vinster.

Den empiriska undersökningen beskrivs i kapitel tre. Inledningsvis presenteras Sync, företaget där undersökningen ägde rum, samt en

genomgång av deras systemarkitektur. Därefter redovisas resultatet av undersökningen, med bland annat mockups och dataflödesdiagram för HTML och VoiceXML. En jämförelse mellan lösningarna samt diskussion och analys av de båda teknikerna med gällande restriktioner rörande mjukvaruarkitekturen förs sedan. I kapitlet diskuteras även undersökningens resultat och revelationer – vad vi kommit fram till och hur resultatet nåddes. Diskussionen berör vår empiriska studie i förhållande till angivna forskningsfrågor. Vi ser på vilka likheter och skillnader som finns, eventuella förändringar och dess följder.

I kapitel fyra redogörs för vilka lärdomar och erfarenheter vi fått med oss i bagaget, både negativa och positiva. Det blir en sammanfattning av det sammanlagda arbetet med undersökningen, som en form av kritisk granskning av det egna arbetet och hanteringen av olika situationer. Vilka fallgropar, problem och andra hinder vi stötte på under arbetets gång och hur de hade kunnat undvikas eller fått en bättre lösning beskrivs.

I det femte kapitlet tas framtiden upp som en aspekt vid utvecklingen och utbredningen av röstteknologier. De variabler vilka spelar en betydande roll för att en ny teknologi ska etablera sig på marknaden och dessutom få så pass starkt stöd att den blir vanligt förekommande, eller ännu bättre blir en standard, diskuteras.

I kapitel sex redovisas kortfattat de slutsatser vi dragit gällande forskningsfrågorna. I kapitlet ges även en sammanfattning av arbetet och som avslutning lämnas några förslag till fortsatta studier inom området.

Som bilagor återfinns koden till de XSL-filer vi utvecklade samt till de motsvarande VoiceXML-filerna. Det finns även en ordlista innan bilagorna, där vanliga förkortningar, koncept och termer inom ämnet förklaras kortfattat.

Sist ligger en samlad lista med alla referenser vi använt oss av.

2 XML och VoiceXML i teorin

2.1 Historik

Internets World Wide Web (webben) har vuxit med en hastighet som saknar motstycke. Från att ha varit ett forum för utbyte av information och multimediadokument sker nu förändringar kring användarmönstret (Baiada 1997). Spetsorganisationer implementerar nya tillvägagångssätt för användning av internet, i synnerhet när det gäller intressanta fördelar av webben som plattform för utveckling och formation av mjukvaruapplikationer.

Webben uppfanns i början av 1990-talet vid CERN, det världsomryktade huvudkontoret för atomforskning i Schweiz. Det var forskarna på CERN som ville finna ett sätt att göra sina forskningsresultat, i form av uppställd data, ljudkommentarer och animerade bilder, tillgängliga för forskarkollegor runt om världen (Baiada 1997).

Webbteknologin formades efter klient-serverarkitekturen där innehållet, till exempel text, bild, digitala ljud- och videofiler, ligger på en server där det kan nås av användare. Materialet hämtas och presenteras av en webbläsare vilken iklär sig rollen som klient (Baiada 1997).

Bland dem som snabbt såg webbens kommersiella möjligheter var affärsanvändarna vilka tidigt skapade sina egna hemsidor på webben. Tillväxten av hemsidor ökade explosionsartat från 130 sidor i mitten av 1993 till uppskattningsvis 230 000 sidor i mitten av 1996 (Bass, Clements & Kazman 1998). Det växande antalet publicerade webbadresser i dagstidningarnas annonser, kollektiva brevhuvuden och TV-sända nyhets- och nöjesprogram är ännu ett tecken på webbens popularitet. Sammanslaget visar dessa publicerade webbadresser tydliga indikationer på webbens användningsområden och potential. Webbens varierande användningsområden har varit en förutsättning för dess expandering och är en förutsättning för att locka till sig nya användare.

Fram till idag har webben varit ett internetrelaterat kommersiellt forum där organisationer haft möjligheten att kommunicera med utomstående. Trots detta är den oftast förekommande applikationen på webben en applikation där man svarar på förfrågningar, till exempel sökmotorer. Andra typiska applikationer innebär annonsering, kommunikation mellan leverantörer och deras kunder samt sökning och mottagning av statisk kollektiv information. Den naturliga följden för all ny teknologi är att ju mer och ju längre den

används desto fler användningsområden upptäcks (Baiada 1997). Förutom att använda webben som ett forum för elektroniska publikationer för att kommunicera med allmänheten, genom att annonsera och ta emot order och så vidare, används nu webben även som en kanal för utveckling och implementering av *business-to-business* applikationer.

Webbens användningsområden är inte ensamma om att bryta nya marker. Även den teknologi som utvecklats för att bistå internetbaserad applikationshantering, används i nya sammanhang. Det senare fallet berör tjänster såsom till exempel kommunikation och handel via applikationer placerade på lokala nätverk.

2.2 XML

2.2.1 XML som utvecklingsverktyg

När man tänker på webbutveckling styrs tankarna osökt in på HTML och i viss mån dess mycket komplexa föregångare, SGML. Ett argument som dock talar till fördel för användandet av XML som utvecklingsverktyg är anledningen till den sistnämndes uppkomst, det vill säga meningen att göra rika strukturerade dokument användbara över webben. XML är dessutom formgiven för att kunna överbrygga kompatibilitetsproblem mellan olika system. Med andra ord stödjer XML integration mellan en katalogtjänst och andra datakällor, såsom tillämpningar och databaser (SIG Security 2000).

Med XML som generiskt språk, kan ägandet av data förbli oförändrat, detta innebär att respektive dataägare kan publicera egen information men även prenumerera på information från andra betrodda datakällor. Vidare gör kombinationen av tekniska funktioner och spridningen som XML fått tekniken till en viktig del i dagens IT-värld, bland annat vid standardisering av data och dokumentformat. XML kan fungera som bryggan mellan alla dataformat och se till att data är synkroniserat mellan applikationer och system (SIG Security 2000).

Tanken är att XML ska kunna fylla glappet mellan HTML och SGML, där SGML har möjlighet till strukturering, men är alltför komplicerat för att utveckla mjukvara med. HTML i sin tur, besitter enkelheten som krävs för att erhålla ett starkt mjukvarustöd men saknar all möjlighet till strukturering (Andréasson, Melvanius & Stenberg 1999). Vidare argument som talar för användandet av XML som utvecklingsverktyg vid internetteknikbaserade tjänster, tas upp i följande stycke om XML. Då ventileras inte bara fördelar med XML utan även vissa nackdelar med föregående tekniker såsom HTML och SGML. Till exempel varför HTML och SGML inte lämpar sig praktiskt för att göra strukturerade dokument tillgängliga över webben.

2.2.2 XML:s bakgrund

Man kan se XML som ett *utbyggbart märkspråk* som i likhet med HTML används för att skapa dokument avsedda att publiceras på internet och granskas i en webbläsare (Åström 2000). Dock är XML inte ett egentligt språk utan en standard för hur man *skapar* märkspråk. Med andra ord en standard för att notera i texter och dokument vilka logiska kategorier olika textavsnitt tillhör, till exempel rubrik, ingress och hypertextlänk. Liksom HTML använder sig XML av speciella märkord eller taggar för märkningen i dokumenten (Svenska datatermgruppen 2000). Standarden skapades av en arbetsgrupp som gick under namnet, SGML Working Group, vars arbete initierades av W3C sommaren 1996 och som sedermera presenterades i november samma år (Åström 2000). Idag är specifikationen ett viktigt verktyg för utvecklare av applikationer, system- och integrationsmodeller (SIG Security 2000).

2.2.3 Tanken bakom XML

När XML skapades var tanken att en standard skulle skapas som erbjöd både SGML:s möjlighet till struktur och HTML:s enkelhet (Andréasson, Melvanius och Stenberg 1999). Vid utvecklandet av XML lades extra vikt på själva utformningen. Meningen var att XML skulle användas och passa in i en distribuerad miljö som webben, där data, uppmärkt i XML, kan vara långt ifrån den maskin den kommer ifrån (Graham & Quin 1999).

XML är utformat att stödja *egenrådig* markering för att XML ska kunna användas till att märka data från vitt skiljda applikationer, genom att använda ett universellt format (Graham & Quin 1999). Vidare ärvde XML en annan viktig funktion av SGML, nämligen den att tillåta egendefinierade märkord. Med andra ord har XML möjligheten att strukturera upp data genom att skapa en intern struktur med talande märkord, ett viktigt särdrag som skiljer XML från HTML (Andréasson, Melvanius & Stenberg 1999). Medan HTML har en fast uppsättning fördefinierade taggar med vilka man kan skapa rubriker, stycken och lägga in bilder, har XML fördelen att utvecklaren själv kan skapa erforderliga taggar (Åström 2000). Utvecklaren väljer själv deras namn och funktion.

Meningen är inte att XML ska ersätta HTML utan snarare fungera som ett komplement till exempelvis HTML (Åström 2000). Man kan säga att XML inte bara består av en teknik utan det krävs flera olika komponenter för det ska fungera. Det vill säga, koda kan man bara göra i *tillämpningar* av XML.

2.2.4 XML:s uppbyggnad och struktur

Eftersom XML separerar innehåll från presentationen underlättas utvecklingen. I en fil sparar man informationen som ska visas tillsammans med de olika märkorden och i en annan fil talar man om hur informationen ska formateras. Man kan se det som att XML utgörs av fyra olika delar (Andréasson, Melvanius & Stenberg 1999):

1. *Strukturdokumentet* eller *Document Type Definition (DTD)* som det egentligen heter utgör kanske den mest centrala delen inom XML, vars syfte är att beskriva en XML-applikation. Det är i DTD-filen strukturen för den information man vill visa byggs upp. Namn, information och regelverk för de olika märkorden och hur de får användas tillsammans är en del av DTD-filens innehåll (Åström 2000). Dessa dokumentdefinitioner sparas i en separat fil med ändelsen *.dtd.* Huvudsyftet är att DTD:n ska garantera att de olika filerna följer de regler som satts upp. Dock är användandet av DTD:n valfritt då meningen är att de endast ska skapas när behov av egna finns.

2. *Informationsdokument* – Själva XML-dokumentet beskriver innehållet till skillnad från HTML-dokumentet, där märkorden endast talar om textens placering såsom rubriker, rader och kolumner (Åström 2000). För XML-dokumentet och de egendefinierade märkorden är situationen annorlunda. Som angavs i punkten innan ska de egendefinierade märkordens namn vara talande och ge en tydlig fingervisning om textens innehåll. Exempel på märkord kan vara:

   ```
   <LIBRARY>
       <AUTHOR>Douglas Adams</AUTHOR>
       <TITLE>Liftarens guide till galaxen</TITLE>
       <TITLE>Restaurangen vid slutet av universum</TITLE>
       <TITLE>Livet, universum och allting</TITLE>
       <TITLE>Adjöss och tack för fisken!</TITLE>
       <TITLE>I stort sett meningslös</TITLE>
   </LIBRARY>
   ```

   Märkorden som finns definierade i DTD-filen beskriver innehållet snarare än hur det ska visas. Av koden framgår det klart och tydligt att innehållet avser böcker. Denna funktion förenklar förståelsen och läsningen inte endast för människa utan även för maskin (Andréasson, Melvanius & Stenberg 1999). En tjänlig tillämpning av funktionen är vid sökningar på internet. Framtidens sökningar kan komma att bli mer exakta eftersom det är enklare för en dator

att ta fram rätt data då det är lättare att förstå vad en XML-fil innehåller (Åström 2000).

3. *Presentationsdokument* – Vid användandet av XML är samtliga taggar skapade, namngivna och definierade av utvecklaren själv och ingen ledning ges för hur dessa ska visas. Med andra ord krävs det ytterligare en fil som hanterar informationsdelen för hur de olika taggarna i XML-filen ska formateras (Åström 2000). Den fil som ger möjligheten att kunna presentera samma information på flera olika sätt, kallas för stilmall. Man brukar tala om två olika typer av stilmallar så kallade, *eXtensible Stylesheet Language (XSL)* och *Cascading Style Sheets (CSS)*. I likhet med DTD:n sparas båda som separata filer med ändelsen *.xsl* respektive *.css.*

   CSS går även under namn som exempelvis stilmall, kaskadmall och formatmall och är en teknik som fanns redan före XML (Åström 2000). Med CSS-mallar kan man helt och hållet bestämma hur de olika elementen ska se ut när det gäller till exempel typsnitt och bakgrundsfärg. I början av XML-filen kan man tala om vilken stilmall den ska använda. Stilmallen kan vara extern, alltså sparad i en separat fil vars namn och absoluta eller relativa sökväg man anger i inledning av XML-filen. Likväl kan stilmallen vara intern, med andra ord är det möjligt att skriva CSS-mallen direkt i XML-filen, det viktiga är då att man vet med säkerhet att man vill använda stillmallen till en enda XML-fil. I annat fall är det bättre att lägga filen separat (Åström 2000).

   XSL är en teknik som utvecklades i samband med XML och är anpassad för att formatera XML-filer. Liksom XML är XSL en W3C-standard framarbetad av *W3C XSL Working Group*. En XSL-mall är egentligen ingenting annat än en XML-applikation och består av två delar; *XSL Transformations (XSLT),* ett språk att transformera XML-filer, och *XSL Formatting Objects (XSLFO),* ett XML språk att specificera formateringssemantik (W3C 2000c). De byggs med andra ord upp av en samling element med bestämda namn och funktioner. XSL-filens uppgift är att plocka ut datan ur en XML-fil och infoga den bland till exempel HTML-element. Resultatet hamnar i ett så kallat *källträd* som då ersätter XSL-elementen med de hämtade värdena tillsammans med HTML-elementen. Det ser ut som vanlig HTML-kod vilket är vad webbläsaren visar. Jämfört med CSS-mallar finns för XSL-filer lite andra möjligheter att formatera XML-filer, bland annat den att återanvända samma text från XML-filen på olika ställen på till exempel en webbsida (Åström 2000).

XSLT möjliggör den första användbara egenskapen, nämligen möjligheten att använda XSL till att transformera XML till exempel HTML. XSLFO gör det även möjligt att med XSL formatera XML direkt utan att gå via till exempel HTML (W3C 2000c). Valfrihet av användandet av XSL-filer gäller på liknande grunder som för DTD-användandet, man ska endast skapa sina egna XSL-filer vid särskilda behov, till exempel om man vill presentera samma information fast på flera olika vis (Andréasson, Melvanius & Stenberg 1999).

4. *Länkning* – Tekniken är egentligen uppdelad i två delar; *XML Linking Language (XLink)* och *XML Pointer Language (XPointer)*. De fungerar ofta tillsammans då genom att XLink används till att skapa en länk till ett dokument och XPointer för att peka ut exakt vart den ska leda. XLink tillåter infogning av element i XML-filer för att skapa och beskriva länkar mellan olika dokument, men även mellan olika platser inom ett och samma dokument (Åström 2000). XPointer används som en fragmentidentifierare för URI-länkar som lokaliserar en källa, till exempel en XML-fil. XPointer är baserat på *XML Path Language (XPath)* och stödjer adressering av interna strukturer i en XML-fil. Det tillåter förflyttning av dokumentträd och val av dess innehåll baserat på alternativa attribut, såsom element typ, attributvärden, egenskapsinnehåll och relativ position (W3C 2000a). Detta betyder att ett dokument, vilken länken leder till, inte behöver innehålla någon egen information om länken (Åström 2000).

2.2.5 Fördelar och vinster med XML

Summerar man fördelarna med att använda XML, ser man att det flexibla sätt på vilket XML organiserar data gör att det lämpar sig väl för integration mellan olika applikationer och system. Detta för att XML kan anpassas till olika dataformat och datastrukturer (SIG Security 2000). Möjligheten att definiera egna märkord gör syntax begriplig samt tolkbar för både människa och dator. Språket är utbyggbart och dynamiskt och med hjälp av de egendefinierade märkorden erhålls en beskrivning av XML-filens innehåll. Andra vinster med XML är att det tillsammans med XSL är möjligt att återanvända samma data på flera olika ställen i ett dokument. Samt att det vid presentation av data är möjligt att använda en och samma stilmall till olika dokument.

2.3 VoiceXML

VoiceXML är ett XML-schema (W3C 2000b), vilket innebär att VoiceXML nyttjar XML:s sätt att uttrycka struktur, innehåll och semantik i dokument (W3C 2000d).

VoiceXML kan både anses vara ett språk samt ett Application Programming Interface (API), enligt Nuance (2000). Det påminner lite om HTML på det sätt att märkord används för att förmedla innehållet. Precis som webbläsaren tolkar HTML-sidan och återger en visuell presentation på skärmen, läser VoiceXML-tolkaren VoiceXML-sidan och återger märkningen som en dialog (Nuance 2000).

Ett av fundamenten med VoiceXML är dess enkelhet. Det är inget fullfjädrat programmeringsspråk och är beroende av andra tekniker för att bli kraftfullt (Parfitt 2000). Precis som HTML är VoiceXML lätt att använda och lära sig och vänder sig därmed till en publik med liten eller ingen programmeringserfarenhet.

2.3.1 Tanken bakom VoiceXML

VoiceXML bygger på arbetet att sätta en gemensam standard för tidigare teknologier såsom Motorolas VoXML och IBM:s SpeechML för tjänster vars interaktion sker via ett röstgränssnitt (Wireless Developer Network 2001). Standarden skapades av VoiceXML Forum vilket är en industriorganisation som grundades av fyra stora och etablerade företag; AT&T, IBM, Lucent och Motorola (W3C 2000b). Det huvudsakliga målet med VoiceXML är att ta tillvara på kapacitet och kraft från webbaserad teknik och innehållstransportering för att tillföra detta vid utvecklande av interaktiva röstapplikationer. Samtidigt är syftet att befria utvecklare av röstapplikationer från lågnivåprogrammering (VoiceXML Forum 2000a).

2.3.2 VoiceXML:s omfattning

VoiceXML beskriver en människa-dator interaktion vilken sker via rösten. I denna interaktion kan det ingå (VoiceXML Forum 2000a):

- Utdata eller svar med syntetisk röst (text-till-tal)
- Utdata med ljudfiler
- Igenkänning av talad indata
- Igenkänning av tonvalsindata (DTMF)
- Inspelning av talad indata
- Telefonitjänster såsom vidarekoppling och frånkoppling

Utdata kan alltså presenteras på två olika sätt för användaren; dels genom så kallad text-till-tal och dels genom ljudfiler (Parfitt 2000).

Text-till-tal innebär att syntetiskt tal genereras från text, vilken läses maskinellt. För varje år ökar kvaliteten på dessa system. Även om syntetiska röster genererade från text-till-tal-system fortfarande lätt kan särskiljas från naturligt eller förinspelat tal är begripligheten hög (Parfitt 2000). Man förstår alltså vad som sägs, även om det inte är lika trevligt som med mänsklig röst.

Ljudfiler. I många situationer kan information som ska presenteras för användaren beredas i förväg. Då kan förinspelade ljudfiler av mänsklig röst användas. Fördelen med detta är att de naturliga nyanseringar som förekommer i det mänskliga talet finns med, vilket ger ett mer tilltalande intryck. Nackdelar med denna lösning är dels att det krävs stora mängder lagringsutrymme och dels oförmågan att presentera godtyckliga meddelanden till användaren. Dessutom kan tidsmässiga förseningar uppstå om stora ljudfiler försöker laddas på tungt trafikerade nätverk (Parfitt 2000).

Det finns även möjlighet att spela in och sedan spela upp ljudinformation från användaren eller från en annan källa (VoiceXML Forum 2000a).

Information från användaren kan förekomma i två olika former; som text vilken har översatts från talad indata via en automatisk röstigenkänningsserver eller som tolkad tonvalsindata (Parfitt 2000).

2.3.3 Arkitekturmodell över VoiceXML

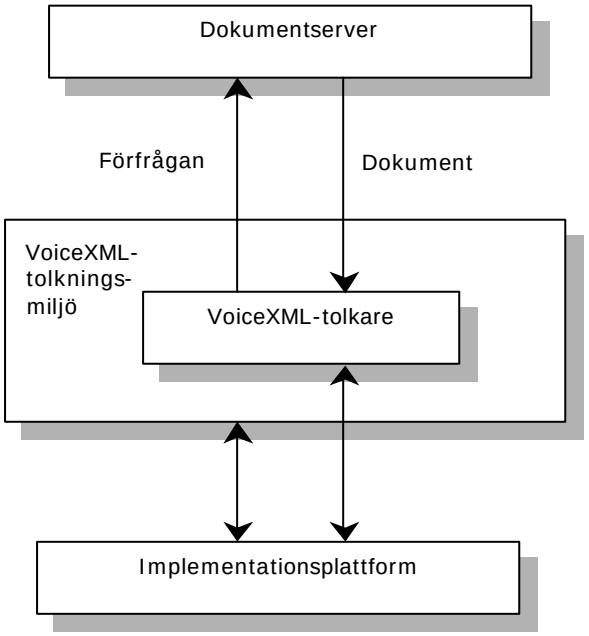
Enligt VoiceXML Forum (2000a) består arkitekturen för VoiceXML av:

- dokumentserver,
- VoiceXML-tolkare,
- VoiceXML-tolkningsmiljö samt
- implementationsplattform.

VoiceXML möjliggör integration av röstservice med dataservice genom så kallad klient-server teknik. En röstservice kan ses som en sekvens av interaktiva dialoger mellan en användare och en implementationsplattform. Dialogerna tillhandahålls av dokumentservrar, vilka kan vara externa och ligga utanför implementationsplattformen. Dokumentservrar underhåller den övergripande servicelogiken, utför databas- och ärvda systemoperationer samt producerar dialoger. Ett VoiceXML-dokument specificerar varje interaktiv dialog som behandlas av en VoiceXML-tolkare. Användarens indata påverkar dialogtolkningen och samlas till förfrågningar vilka skickas till en dokumentserver. Dokumentservern kan sedan i sin tur svara med ett

annat VoiceXML-dokument för att fortsätta användarsessionen med ytterligare dialoger (W3C 2000b).

I figur 2.1 nedan illustreras arkitekturen:



Figur 2.1. Arkitekturmodell över VoiceXML (W3C 2000b, egen översättning).

En dokumentserver (till exempel en webbserver) behandlar via *VoiceXML-tolkningsmiljön* förfrågningar genererade av klientapplikationen, en så kallad *VoiceXML-tolkare*. Servern genererar i sin tur VoiceXML-dokument som svar vilka bearbetas av VoiceXML-tolkaren.

VoiceXML-tolkningsmiljön kan avlyssna användarindata parallellt med VoiceXML-tolkaren. Till exempel kan en särskild VoiceXML-tolkningsmiljö alltid lyssna efter en speciell fras (*escape phrase* på engelska) som tar användaren till en avancerad personlig assistent, medan en annan tolkningsmiljö lyssnar efter fraser som ändrar användarpreferenser som till exempel ljudvolym eller text-till-tal karaktäristik (W3C 2000b).

Implementationsplattformen kontrolleras av VoiceXML-tolkningsmiljön och VoiceXML-tolkaren. Till exempel kan VoiceXML-tolkningsmiljön bära ansvaret att upptäcka och svara på inkommande samtal i en interaktiv röstresponsapplikation men också att skaffa fram det initierande VoiceXML-dokumentet, medan VoiceXML-tolkaren leder dialogen efter svaret. Implementationsplattformen genererar händelser som svar till användaraktiviteter (till exempel indata genom tal eller tecken samt

frånkoppling) och systemhändelser (till exempel förfallotid). Några av dessa händelser hanteras av VoiceXML-tolkaren själv, specificerade av VoiceXML-dokument, medan andra hanteras av VoiceXML-tolkningsmiljön (W3C 2000b).

2.3.4 VoiceXML:s uppbyggnad och struktur

Ett VoiceXML-*dokument* får ändelsen *.vxml* och utgörs av en eller flera *dialoger*. En *applikation* kan bestå av ett eller flera VoiceXML-dokument som tillsammans bildar en *interaktion* (även kallad *session*) mellan användare och system. Användaren befinner sig hela tiden i en dialog åt gången. Varje dialog bestämmer nästkommande dialog och övergångarna specificeras genom URI-länkar, vilka definierar nästa dialog eller dokument som ska användas. Interaktionen avbryts då en dialog ej specificerar en efterföljare, eller om interaktionen med vilje avbryts av antingen användaren, ett dokument eller tolkningsmiljön (VoiceXML Forum 2000a).

Ett exempel på en interaktion mellan människa (M) och dator (D) kan se ut så här:

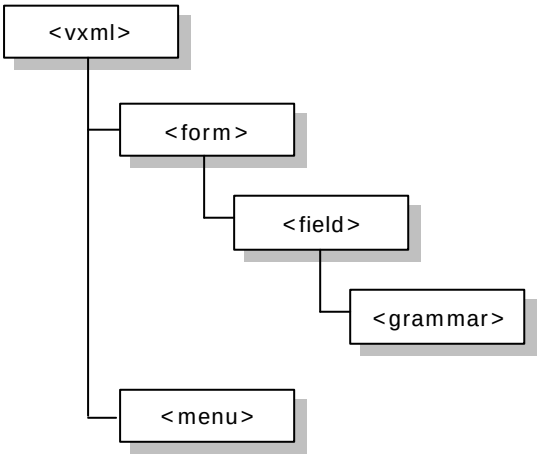
D: Vill du ha nyheter, väder eller TV-tablå?
M: Väder.
D: Ange landskap och stad.
M: Skåne och Lund.
D: Vädret i Lund är soligt till halvklart och 15 grader Celsius.
M: TV-tablå.
D: Ange dag och tidpunkt.
M: Idag klockan 21.00.
D: På kanal 1 visas Curry nam-nam, på kanal 2 visas Aktuellt. Önskas fler kanaler?
M: Nej, tack. Avsluta.

Det finns två olika typer av dialoger; *formulär* och *menyer*. Formulär presenterar information, samlar in data samt sköter en viss felhantering (Parfitt 2000). Menyer erbjuder olika valmöjligheter och en ny dialog exekveras baserat på det val som användaren gör. Formulär innehåller *inmatningsfält* och varje sådant fält kan associeras med VoiceXML-*grammatik*, vilka styr de tillåtna inmatningsvärdena.

Grammatik är en mängd regler som specificerar vad rösttolkaren kommer att känna igen. En bra grammatik använder sig av ord och fraser som användaren känner till. Det är viktigt att grammatiken på ett riktigt sätt reflekterar vad uppgiften avser samtidigt som det sker med så mycket flexibilitet och naturlighet som möjligt (Parfitt 2000). Grammatik kan även specificeras på formulärnivå och då kan flera fält fyllas i utifrån endast ett

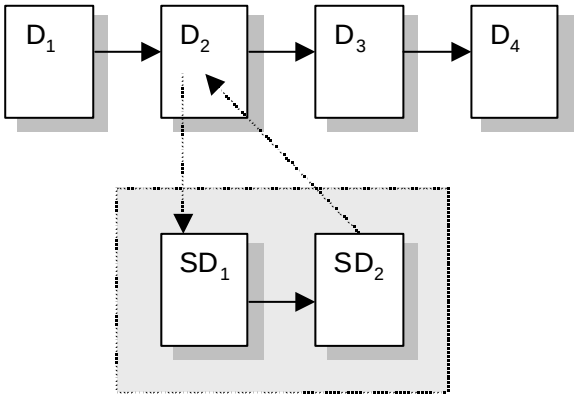
yttrande från användaren (W3C 2000b). Det finns även färdiga grammatikbibliotek att använda sig av (Martin 2000).

I figur 2.2 nedan illustreras de olika komponenter som kan ingå i ett VoiceXML-dokument.



Figur 2.2. Övergripande struktur av ett VoiceXML-dokument (Martin 2000, egen modifikation).

En *subdialog* kan liknas vid ett funktionsanrop, vilken erbjuder möjligheten att starta en ny interaktion och sedan gå tillbaka till ursprungsläget. Det vill säga information om lokala variabler, grammatik och status sparas. Subdialoger kan komma väl till användning då man vill skapa dialoger som kan delas mellan flera VoiceXML-dokument i en applikation, alternativt om man vill skapa ett dialogbibliotek som sedan kan återanvändas av flera applikationer (W3C 2000b).



Figur 2.3. Exekveringsflödet när en serie dokument (D) transfereras till en subdialog (SD) och sedan tillbaka (VoiceXML Forum 2000a).

VoiceXML kan även hantera händelser (*events* på engelska), vilka ej går under den typiska formulärfunktionaliteten. Händelser kastas av implementationsplattformen vid olika situationer som till exempel då en användare ej svarar, svarar otydligt, ber om hjälp etcetera. Tolkaren kastar också händelser då den hittar semantiska fel i VoiceXML-dokument (VoiceXML Forum 2000a). Händelser kan användas på en mängd kreativa sätt för att förhöja användarupplevelsen (Parfitt 2000).

Exempel på typiska händelser kan vara *help*, *noinput*, *nomatch* och *error*. Härnäst ges ett exempel på en situation där grammatik och händelser samverkar och avlyssnas på olika nivåer. På dokumentnivå finns en grammatik specificerad som känner igen kommandona *cancel* och *help*, vilket innebär att användaren när som helst kan avbryta sessionen eller be om hjälp. Tolkningsmiljön avlyssnar indatakanalen och kastar en *noinput*-händelse då en viss förbestämd tid har förlöpt utan att den noterat någon röst. Denna tid är typiskt satt på tio sekunder och kan definieras som en systemparameter. Tolkningsmiljön kastar en annan händelse, *nomatch*, ifall ljudnivån på indatan är tillräcklig för att utlösa igenkänningsfunktionen men då inget matchande resultat returnerats.

Länkar stödjer mixad initiering (*mixed initiative* på engelska) vilket tillför flexibilitet och kraftfullhet till röstapplikationer. Mixad initiering innebär att användare och system alternerar i att bestämma vad som ska hända. Om det som användaren anger matchar länkens grammatik går sessionen över till länkens destination (URI). En länk kan också kastas som en händelse för att gå vidare till en ny destination (VoiceXML Forum 2000a).

2.3.5 Fördelar och vinster

VoiceXML är ett förhållandevis enkelt och standardiserat märkspråk. Enligt Nuance (2000) finns det därmed många fördelar med detta språk:

Lägre skicklighetskrav, större utvecklarkrets – VoiceXML är ett lättförståeligt skriptspråk som kommer att tilltala den breda systemutvecklarkretsen. Eftersom språket saknar den avancerade konstruktion samt tillhörande inlärningskurva som andra moderna programmeringsspråk ofta har, kommer även ovana programmerare att kunna utveckla med VoiceXML.

Portabel standard – VoiceXML Forum har föreslagit standarden VoiceXML 1.0 vilken har erkänts och antagits av W3C. Standarden kommer medföra att språket uppnår stor genomslagskraft.

Möjlighet för Voice Service Providers (VSP) – VoiceXML tillför telefoni och röstigenkänning till webben. Vidare möjliggör VoiceXML avskiljning av röst- och telefoniresurser från applikationen. Detta kommer att nära

uppkomsten av VSP:er, vilka agerar som värd för röstapplikationen (precis som Internet Service Providers (ISP) agerar som värd åt webbapplikationer).

Tillvaratagande på investeringar i webbinfrastruktur – Webbtekniken har krävt investeringar i infrastruktur och i applikationsutveckling (som till exempel distribuerade *N-tier* webbläsare, webbservrar, applikationsservrar och databasservrar). VoiceXML passar väldigt väl in i denna modell och får därmed mycket gratis.

2.3.6 Begränsningar

Att VoiceXML är ett enkelt och förhållandevis lätt utvecklingsspråk medför också vissa begränsningar. Enligt Nuance (2000) stödjer VoiceXML ej sofistikerade dialoger, återanvändning av kod eller röstverifiering. Språket har även en mycket begränsad händelse- och undantagshantering (*exception handling* på engelska) jämfört med andra kraftfullare språk. Ju mer komplexa och sofistikerade dialoger som avses utvecklas desto högre prestationer krävs och detta förhållande ökar exponentiellt. VoiceXML Forum (2000a, sidan 7) poängterar också detta:

> *”While VoiceXML strives to accommodate the requirements of a majority of voice response services, services with stringent requirements may best be served by dedicated applications that employ a finer level of control.”*

Dock kan VoiceXML med fördel aktighet kombineras med andra tekniker och programmeringsspråk, såsom till exempel XML eller javaobjekt, för att öka kraftfullheten (Parfitt 2000, Nuance 2000).

Även om VoiceXML blivit en W3C-standard är det i sig ingen garanti för att det är den teknik som kommer att slå igenom som framtidens voice browsing-teknik. Det är mycket som talar för att det *kan* bli voice browsings motsvarighet till webbens HTML. Men det är ett så pass nytt utvecklingsspråk och ny standard att det ändå mycket väl kan bli ett annat språk eller en annan teknik som slår igenom och vinner marknad. Risken finns då det ständigt sker förändringar, speciellt inom marknaden för internetteknik, där det hela tiden dyker upp nya språk och standarder.

3 XML och VoiceXML i praktiken

Uppsatsens arbete har sin grund i den praktik vi gjorde på Sync under sista terminen på vår magisterutbildning hösten 2000. Det var på Sync som all undersökning av VoiceXML och härledd dokumentering skedde. Företaget fungerade som ett resurscenter men också som skådeplats och testlabb för våra försök att testa VoiceXML i en systemstruktur där XML-filer utgör en viktig del av företagets mjukvaruarkitektur. Nedan följer en närmare presentation av företaget.

3.1 Sync Business Process Integration AB

Sync Business Process Integration AB (Sync) är ett kunskapsföretag i IT-branschen verksam inom affärsprocess integration. Företaget har cirka 60 anställda med huvudkontoret beläget i centrala Malmö och ett marknadskontor i centrala Stockholm. Verksamhet är i full gång för att etablera ytterligare marknadskontor utomlands (Ottosson 2001).

Sync utvecklar lösningar och standardiserade produkter baserade på webb- och integrationsteknologi för skapande av affärsportaler. Företaget tar fram koncept, metoder och produkter för att effektivisera de horisontella affärsprocesserna i företag, utan organisationsförändringar och med hög kostnadseffektivitet. Syftet är att utnyttja internet, befintliga affärssystem och infrastruktur för att skapa förutsättningar för kunden att samordna och förenkla sina affärsprocesser med sina affärspartners (Ottosson 2001).

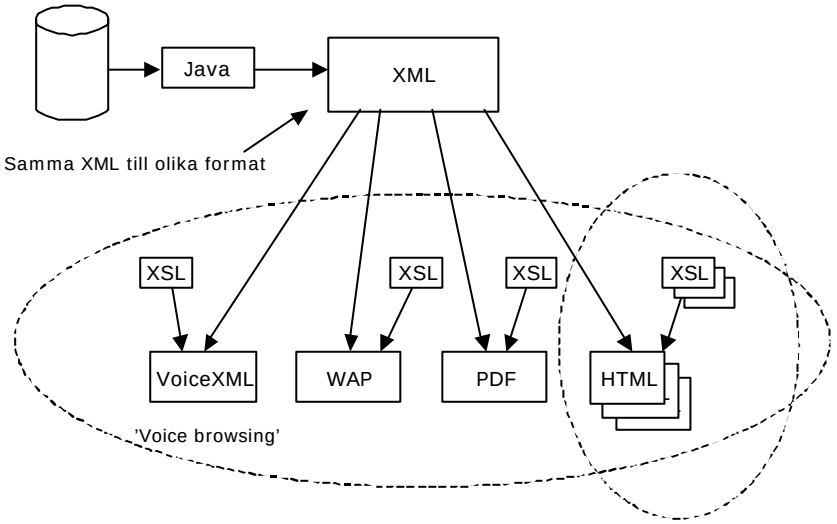
Företagets marknads- och kundsegment är multinationella företag med flera tillverkande, distribuerande och säljande enheter. Det rör sig främst om större industri- och tjänsteföretag där det finns ett stort behov av att konsolidera och samordna (synkronisera) globala verksamheter samt att knyta leverantörer och kunder närmare sig (Ottosson 2001).

3.2 Syncs mjukvaruarkitektur

Sync använder sig av en, enligt Andersson (2000), ganska unik systemarkitektur för sin utveckling av användargränssnitt och interaktionskanaler. De arbetar med XSL-filer för att stödja olika språk och stilar i sina gränssnitt. Meningen är att det ska gå lätt och smidigt för kunden

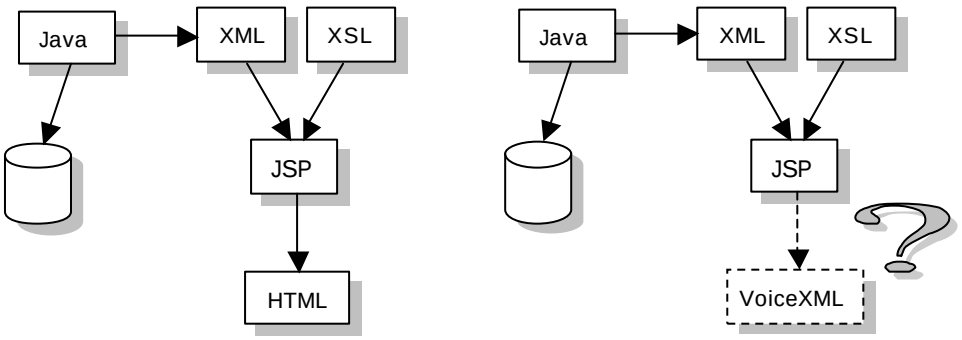
att ändra språk från till exempel svenska till engelska på applikationens textsträngar men också att ändra utseendet för applikationen. I det senare fallet kan det vara aktuellt vid både produktutvecklingen av applikationen och individuella inställningar hos kunden. Ett exempel på stilstöd sett ur produktsynpunkt kan vara att erbjuda respektive kundföretag ett företagsanpassat gränssnitt med vederbörande företags färger och logotyp. Som exempel på individuella inställningar kan den information som presenteras i applikationen vara olika beroende på inloggad användare. Detta utan att det krävs extraprogrammering och konfigurering för Syncs programmerare (Andersson 2000). Arbetssättet är uppbyggt så att *samma* XML-fil ska kunna användas oberoende av språk och stil.

Man kan säga att Syncs modell för gränssnittshantering antar två dimensioner (Andersson 2000); dels stöd för olika format (interaktionssätt) och dels stöd för olika utseende inom samma format. Den förstnämnda dimensionen är den som går horisontellt i figur 3.1. Denna demonstrerar att samma XML-fil förser olika format med information. Detta för att man ska slippa gå in och koda om för varje nytt format. Den andra dimensionen illustreras vertikalt i figuren och visar att det kan förekomma olika utseende inom samma format, det vill säga olika stilar och språk.



Figur 3.1. Illustration över Syncs mjukvaruarkitektur med de två dimensionerna (enligt Andersson 2000).

Vi går lite djupare och tittar på hur arkitekturen ser ut för webbgränssnittsutvecklingen. Systemet är i grund och botten utvecklat i java som interagerar med en databas. En JSP-fil anropar metoder för att skapa en XML-fil vilken sedan tolkas tillsammans med en XSL-fil, varpå en HTML-fil genereras. Se figur 3.2.



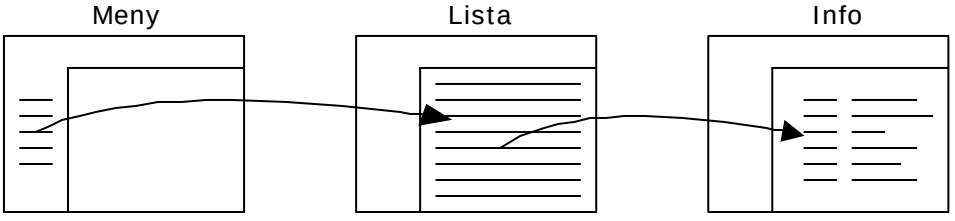
Figur 3.2. En mer detaljerad modell över arkitekturen (egen referens).

3.3 Mockups

Den enkla och väl avgränsade delen av systemet vi valde att utföra vår undersökning på innebar att ur ett register plocka fram telefonnumret till en viss handelspartner (*trading partner* på engelska). Användaren anger ett namn på en handelspartner som indata och som utdata returneras telefonnumret.

3.3.1 HTML-mockup

Mockupen i HTML består av traditionella webbsidor där navigering sker genom hypertextlänkar. Den enkla dialogen består av tre webbsidor vilka huvudsakligen innehåller meny, lista och information där flödet mellan sidorna flyter i just den ordningen. Figur 3.3 beskriver mockupen i HTML där pilarna visar navigeringen mellan de tre webbsidorna.



Figur 3.3. Mockup över HTML-dialogen (egen referens).

Den första sidan visar en meny till vänster med olika alternativ. Här klickar användaren på en länk i menyn för att gå vidare. I detta fall väljer användaren alternativet *Trading partner*. Nästa sida som öppnas visar en lista över alla handelspartners som finns i registret. Användaren väljer igen

genom att klicka på en viss handelspartner i listan. Ytterligare en webbsida öppnas vilken innehåller information om den valda handelspartnern. Här kan användaren bland annat utläsa handelspartnerns telefonnummer.

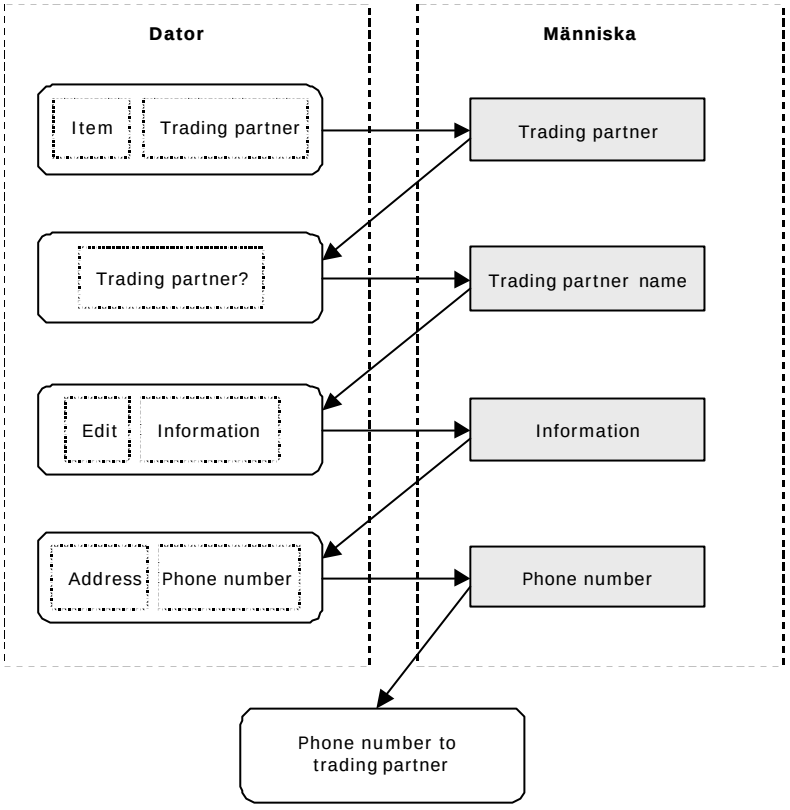
Dialogen mellan dator (D) och människa (M) ter sig alltså som följande:

M: Användaren väljer *Trading partner* bland alternativen i menyn.
D: En lista över alla handelspartners som finns i registret returneras.
M: Användaren väljer en handelspartner genom att klicka på ett namn i listan.
D: Information om handelspartnern, bland annat telefonnummer, visas.

3.3.2 VoiceXML-mockup

Dialoger i röstgränssnitt kan naturligtvis se olika ut beroende på sammanhang och situation. Datorns förfrågningar kan till exempel ställas på olika sätt. Antingen kan alternativen användaren har att välja mellan läsas upp, eller kan en fråga ställas där användaren själv får ange ett svar. Det senare exemplet kan vara mer praktiskt om det finns många olika alternativ. Om inte användaren förstår vad som ska anges kan en hjälpfunktion läggas in där systemet förklarar mer utförligt för användaren vad som avses.

I figur 3.4 illustreras en enkel mockup över VoiceXML-dialogen, där människa och dator kommunicerar via den mänskliga rösten respektive text-till-tal.



Figur 3.4. Mockup över VoiceXML-dialogen (egen referens).

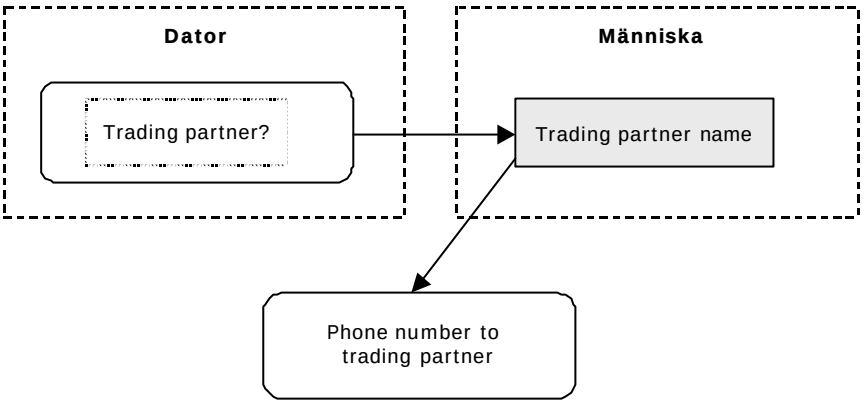
Ett exempel på hur denna dialog skulle kunna se ut mellan dator (D) och människa (M):

D: Choose *Item* or *Trading partner*.
M: Trading partner.
D: What trading partner?
M: Volvo Construction Equipment.
D: Choose *Information* or *Edit*.
M: Information.
D: Choose *Address* or *Phone number*.
M: Phone number.
D: The phone number to Volvo Construction Equipment is 020-878845.

Detta exempel visar på hur en människa-dator-dialog skulle kunna se ut med ett röstgränssnitt i Syncs affärsapplikation. Vi valde dock att implementera en förenklad och avskalad version, då det i denna undersökning endast avsågs att testa tekniken. Dialogen behövde inte heller vara användarvänlig eller innehålla fel- och undantagshantering, då denna bara var till för oss

själva att testa tekniken och systemet, det vill säga att rätt parametrar kom in och ut.

Nedan visas en mockup över den förenklade dialogen (se figur 3.5) där bland annat alla valmöjligheter är bortplockade. Kvar är det väsentliga, det vill säga att användaren anger ett namn på en handelspartner och att telefonnumret returneras.



Figur 3.5. Förenklad mockup över VoiceXML-dialogen (egen referens).

Nedan följer den dialog vi implementerade och konversationen ter sig som följande:

D: What trading partner?
M: Volvo Construction Equipment.
D: The phone number to Volvo Construction Equipment is 020-878845.

Vid datainmatningen lade vi in en liten hjälpfunktion, mest för att visa på hur det kan se ut. Om användaren säger *help* istället för att ange ett namn på en handelspartner returneras följande:

D: Please speak the trading partner name for which you want the phone number.

3.4 Dataflödesdiagram

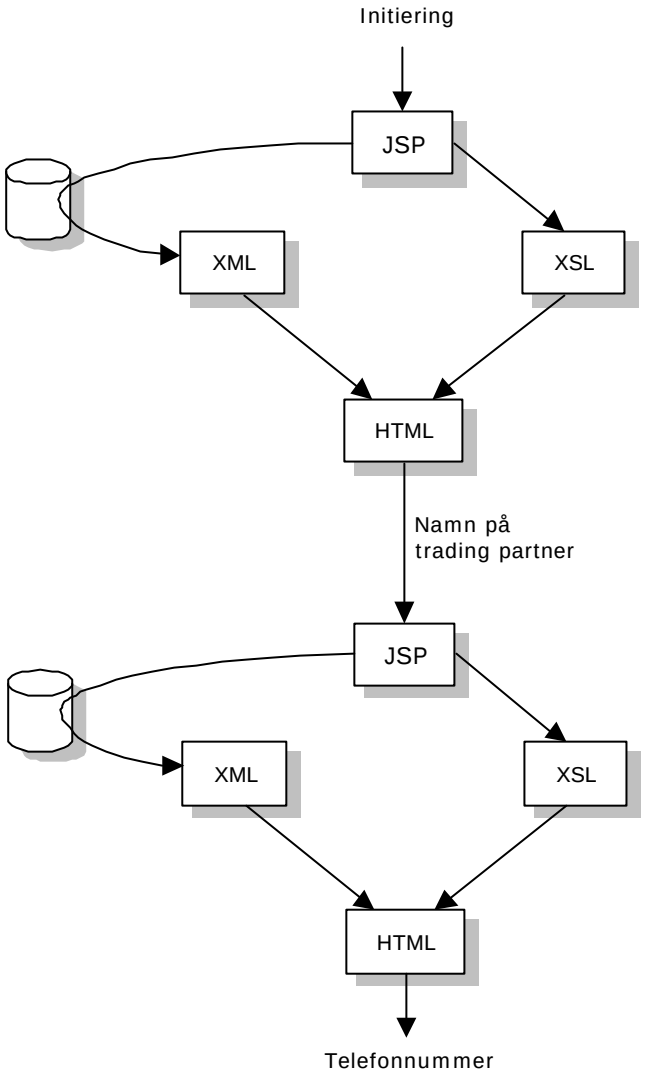
Med hjälp av dataflödesdiagrammen visualiseras den tekniska lösningen på ett tydligt sätt. Det är utifrån dessa vi redogör för våra resultat.

Vi utvecklade inte några JSP- eller XSL-filer för HTML-lösningen då hela Syncs webbgränssnitt är utvecklat enligt denna modell, så det fanns många exempel att tillgå. Dessutom arbetade vi mycket med detta under vår praktik så vi var väl insatta i hur det fungerade.

Datastrukturen består av fyra huvuddelar; JSP, XSL, XML samt antingen HTML eller VoiceXML. JSP- och XSL-filerna är de filer som skapas manuellt medan XML- och HTML- respektive VoiceXML-filerna skapas under exekvering.

3.4.1 HTML-dataflödesdiagram

Nedan skildras dataflödet för HTML-lösningen (se figur 3.6). Den första nivån i dialogen, där man väljer *Trading partner* i menyn, valde vi att utelämna. Detta för att hamna på samma nivå som den implementerade förenklade mockupen i VoiceXML-lösningen.



Figur 3.6. Dataflödesdiagram över HTML-lösningen (egen referens).

En mer ingående förklaring av flödet sker nedan då vi beskriver dataflödesdiagrammet samt resultatet för VoiceXML-lösningen (se kapitel 3.4.2).

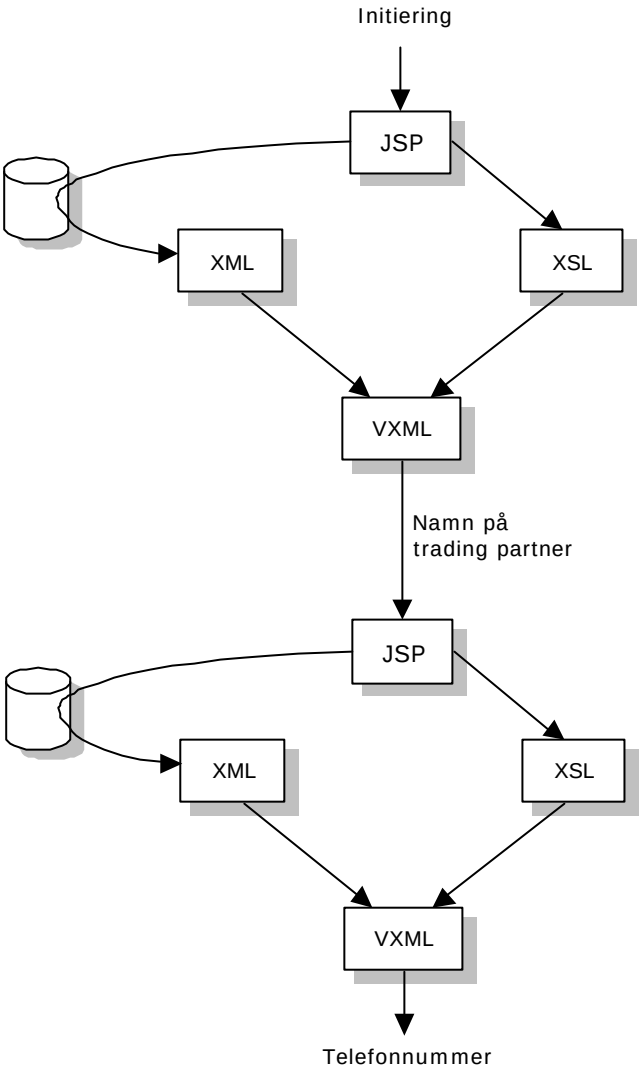
3.4.2 VoiceXML-dataflödesdiagram

Till vår lösning skapade vi två JSP-filer och två XSL-filer. XML- och VoiceXML-filerna skapades under exekvering. Vi går inte närmare in på hur de olika filerna exakt ser ut eller kodats, utan koncentrerar oss på deras uppgifter i lösningen.

- **JSP**-filens uppgift är att generera en XML-fil med data från databasen samt att initiera tolkningen av XSL-filen.
- **XML**-filen innehåller metadata och tillgodoser XSL-filen med data.
- **XSL**-filens uppgift är att styra hur den hämtade datan från XML:en ska presenteras.
- **VoiceXML**-filen är *slutprodukten* som genereras, vilken är själva gränssnittet mot användaren och som sköter dialogen med densamme.

Man kan se lösningen som två loopar där exekveringen sker uppifrån och ned (se figur 3.7). Först anropas en JSP-fil vilken hämtar data från databasen och genererar en XML-fil, som i detta fall innehåller alla handelspartners i registret samt information om dessa. JSP-filen initierar även en XSL-fil, som tolkas med XML-filen, vilket i sin tur resulterar i en VoiceXML-fil. VoiceXML-filen exekveras och användaren anger då ett namn på en handelspartner som indata, vilket läggs i en variabel.

Denna variabel skickas sedan vidare som parameter till nästa loop och då till nästa JSP-fil, vilken hämtar data om just denna handelspartner ur databasen och lägger i en XML-fil. JSP-filen initierar nästa XSL-fil vilken tolkas med XML-filen och nästa VoiceXML-fil genereras och exekveras. Som utdata fås telefonnumret till rätt handelspartner, vilket presenteras för användaren genom text-till-tal.



Figur 3.7. Dataflödesdiagram över VoiceXML-lösningen (egen referens).

Med hjälp av XML-filens egendefinierade märkord underlättas sökningen efter den efterfrågade datan, vilken definierats av XSL-filen, antingen genom de indata som angivits av användaren eller genom annan erhållen data förvärvad under processen.

3.5 Resultatsdiskussion och analys

Ur de dataflödesdiagram som presenterades i resultatsbeskrivningen ovan deriveras följande resultatsdiskussion.

3.5.1 Jämförelse av HTML- och VoiceXML-lösningarna

Undersökningen visar att kombinationen XML och VoiceXML fungerar bra tillsammans. Likaså besvarades följdfrågan med samma positiva svar, det vill säga, samma XML-fil som används för HTML fungerar lika väl för VoiceXML. XML-filerna behövde med andra ord inte ändras eller struktureras om på något sätt för att det skulle fungera. Inte heller behövde vi gå in i databasen för att ändra något. Undersökningen visade att VoiceXML även fungerar i en kontext med JSP- och XSL-filer, vilket illustreras i figur 3.7, där en VoiceXML-fil genereras som slutprodukt efter ett samarbete mellan JSP, XSL och XML. Om lösningen verkar bekant beror det på att det vi just beskrev även fungerar som en beskrivning över Syncs mjukvaruarkitektur. Det betyder att vår lösning för VoiceXML passar in i Syncs mjukvaruarkitektur vilken innefattar JSP, XSL och XML. Minns figur 3.1 över Syncs mjukvaruarkitektur där kombinationen XML och XSL resulterar i HTML.

Gör man dessutom en jämförelse mellan HTML- och VoiceXML-lösningen påvisar undersökningen att de båda lösningarna är väldigt lika. Ser man till de båda figurerna 3.6 och 3.7 ter sig lösningarna identiska så när som på slutprodukten som är en VoiceXML-fil respektive en HTML-fil. Funktionellt sätt är de även identiska. Både HTML- och VoiceXML-lösningarna utnyttjade en användbar egenskap som XSL-tekniken erhåller, det vill säga transformationsspråket XSLT. Lösningarna använder XSL till att transformera XML till HTML respektive VoiceXML. I HTML-lösningen hamnade resultatet i ett källträd som ersatte XSL-elementen med de hämtade värdena tillsammans med HTML-elementen. Källträdets innehåll ser ut som vanlig HTML-kod. I VoiceXML-varianten erhålls istället ett källträd med VoiceXML-kod. En JSP-fil initierar hela processen och bygger ett XML-träd av databasens innehåll, genom vilken XSL-filen hämtar de värden som sedan presenteras via motsvarande gränssnittformat, det vill säga det filformat som gäller för HTML respektive VoiceXML.

Den marginella skillnaden ligger inte i funktionaliteten utan i de filer vardera lösning nyttjar. Då menar vi inte skillnaden på presentationsformaten, webb- kontra röstgränssnitt, utan de filer som föregår de genererade resultaten. Vi nämnde tidigare att ingen förändring krävdes av XML-filerna för att det skulle fungera. Det som dock behövde ändras, eller egentligen mer korrekt skapas nya, var XSL-filerna. Med andra ord ligger skillnaderna i XSL-filerna för HTML respektive VoiceXML.

Anledningen till varför nya XSL-filer var tvungna att byggas, beror på att det är XSL-filerna som sköter presentationsformatet i aktuell mjukvaruarkitektur. Det är just XSL-filerna som definierar vilken information som ska visas och hur den ska presenteras, det vill säga vilket gränssnittsformat som används. De XSL-filer som fanns färdiga för Syncs

produkt var helt anpassade till att presentera informationen i ett webbgränssnittsformat, vilket inte var meningen vid vår undersökning. Därför skapade vi nya XSL-filer eftersom behovet fanns att presentera samma information fast på ett annat sätt. Vi ville endast generera VoiceXML-filer som presenterade resultatet i ett röstgränssnitt, utan att blanda in ett extra attribut som ett webbgränssnitt med röststöd hade inneburit.

Vinster som följer med XML är rena fördelar vår lösning dragit nytta av. Förutom XSL-filens uppgift att definiera och generera presentationsformatet genom att hämta data ur XML-filen och infoga den bland till exempel VoiceXML-element, använder vi oss av de fördelar XML-filerna ger genom de egendefinierade märkorden. Vi drog nytta av de av Sync egendefinierade märkorden, detta framför allt vid sökningen efter begärd data. Sökningen förenklades genom att XSL-filen direkt gick till den nivå, det vill säga de märkord i XML-trädet där eftersökt data låg. Dessutom var det möjligt att återvinna samma data på mer än ett ställe i dokumentet.

XSL-filen gjorde det även möjligt att med hjälp av XML-filen, dynamiskt skapa grammatiken för VoiceXML-filen så att VoiceXML-tolkaren hade möjlighet att känna igen alla alternativa handelspartners som fanns lagrade i databasen och därmed fungerade som möjlig indata.

3.5.2 Språk- och stilstöd

I Syncs mjukvaruarkitektur har XSL-filen en betydande plats. Faktum är att det är XSL-filen som bestämmer hur och vilken data som ska presenteras. Med andra ord bestäms utseendet på resultatet, i Syncs fall en HTML-sida, och informationen på densamma, av innehållet i XSL-filen. Sync använder sig av XSL-filer med anledning av just det erhållna språk- och stilstödet. Med hänsyn till att det är på Sync undersökningen ägt rum, kan det vara intressant att ta upp frågan om språk- och stilstöd även för VoiceXML. Precis som i mjukvaruarkitekturen på Sync, har XSL-filerna även i vår lösning en betydande plats, för att inte säga central roll.

Förutom att likheterna mellan arkitekturen för det HTML-baserade gränssnittet och det VoiceXML-baserade gränssnittet är mer än slående, kan man tänka sig att andra fördelar som gäller för HTML-lösningen även fungerar som vinningar för VoiceXML-lösningen. Ett exempel kan vara språk- och stilstöd, då framför allt språkstödet. En applicering av språkstödjande XSL-filer avsedda att generera VoiceXML-filer torde vara en utmärkt tillämpning, där XSL-filen hämtar rätt strängar beroende på användarens individuella inställningar och presenterar dem på det språk användarens preferenser påbjuder. När det gäller stilstöd kanske det kan vara en fråga om en förvald röststil för operatorn. Användaren har till

exempel kanske valt att utdatan ska presenteras av en äldre mansröst eller en klingade barnröst.

3.5.3 Andra möjligheter

Vi ser även andra möjliga lösningar, där VoiceXML är en del av arkitekturen men övriga delar består av andra tekniker än dem vi tagit upp i vår undersökning. Det kan även finnas andra alternativa lösningssätt såsom olika variationer på vår lösning. En möjlig variation kan till exempel vara en detaljrikare testprototyp eller en testprototyp vars kapacitet vidare spänner över tänkt systems olika egenskaper. Snyggare lösningar är också tänkbara variationer. Då menar vi snyggare lösningar både ur programmerings- och funktionalitetshänseende, samt kanske även ur användbarhetssynpunkt. Nu gavs det inte utrymme att förbättra och förfina vår lösning, utan vi koncentrerade oss på att genomföra undersökningens primära mål.

Vi tror att ett webbgränssnitt med integrerat röststöd är en möjlighet, mycket på grund av just likheterna mellan HTML- och VoiceXML-lösningen. Fungerar de båda varianterna i samma kontext torde det vara möjligt att slå ihop dem och på så vis få en webbapplikation med stödjande röstgränssnitt. Faktorer som pekar på en integrerad lösningsmöjlighet i detta fall är delvis XML-teknikens flexibla sätt att organisera data, men också XML:s möjlighet att anpassas till olika dataformat och datastrukturer.

4 Lärdomar och erfarenheter

4.1 VoiceXML

VoiceXML i sig är relativt enkelt att lära sig, åtminstone i sådan utsträckning att man kan skriva en fungerande applikation som går att köra med grammatik, subdialoger, menyer och så vidare. Dock är det viktigt att poängtera att det krävs mycket mer praktiskt arbete där fler scenarier ingår för att verkligen kunna göra ett bra program. För att erhålla en maximal lösning kan det även vara viktigt att titta på andra tillvägagångssätt och möjligheter, som till exempel kompletterande tekniker.

Genom våra tester har vi fått känna av VoiceXML-teknikens många sidor, både för- och nackdelar. En framträdande sida av VoiceXML är, som vi tidigare nämnt, dess enkelhet. Denna enkelhet har dock ett pris i form av begränsad prestanda. Med andra ord är VoiceXML inte särskilt kraftfullt ur programmeringssynpunkt och kan därmed inte stå för sig självt utan behöver stödjande tekniker. Det är inte möjligt att med en VoiceXML-fil hantera en sträng med jämförelseoperationer om inte användaren själv tidigare har angett strängen som indata. Det vill säga, en VoiceXML-fil kan inte ta indata, genomföra jämförelser med lagrad data för att sedan presentera tillhörande post. För att det ska vara möjligt måste kompletterade teknik till.

Möjligheten finns att andra lösningar än den vi redogjort för är lika bra eller kanske ännu bättre. Vid en förestående reell implementation av röststödda applikationer vore det fel att gå miste om en mer avgörande lösning. Det naturliga vid en dylik implementation, är att välja en applikation vars egenskaper och utförbara händelser hanteras med bästa tänkbara lösning för att erhålla högsta möjliga prestanda, underhållbarhet, användbarhet och så vidare.

4.2 Arbetsprocessen

Utomstående personer som inte varit inblandade praktiskt i arbetet fungerade som bollplank där nya influenser införskaffades, speciellt då vi fastnade eller allmänt stötte på problem. Till exempel kunde personer med annan kunskap och kompetens på företaget belysa problemet ur ett annat perspektiv.

Vi kom på oss själva med att tänka huruvida en lösning var snygg eller inte, eller om det finns andra lösningar som *såg* snyggare ut (ur programmeringssynpunkt). Det vill säga inlärda tankegångar i rollen som till exempel programmerare. Vi tyckte det tog emot att släppa detta inlärda tankesätt för att endast fokusera på att tänka problemlösande, intuitivt och enkelt. Huvudsyftet för denna undersökning var trots allt att svara på forskningsfrågorna, inte att ge en så snygg lösning som möjligt. Att bygga snygga prestandaeffektiva lösningar blir aktuellt först vid en förestående implementation.

Huruvida en undersökning är framgångsrik eller inte, har nog sin grund i undersökarens inställning till uppgiften. Vi tror att mycket är vunnet om man som undersökare är lyhörd och öppen för nya infallsvinklar och idéer. Med andra ord, att man under processen är villig att ändra sin egen inställning och uppfattning om uppgiften och dess hantering.

5 Framtiden

Sedan mitten av 1990-talet har många marknadsledande företag använt system med stöd för röst vid olika servicetjänster på webben, som till exempel vid flyginformation och realtids aktiehandel. Denna röststödda elektroniska handel (*voice-enabled e-commerce* på engelska) kallas V-Commerce (Nuance 2000). Idag sammanförs olika individuella röststyrda applikationer och skapar ett nytt nätverk, så kallat *Voice Web*. Ett växande antal företag börjar i allt högre hastighet att förena sina olika tjänster (som enhetlig meddelandehantering, långdistanssamtal eller shopping) och internetbaserade innehåll (till exempel aktiekurser, filmer och gula sidorna) samt gör dem tillgängliga från vilken telefon som helst. Då Voice Web förvandlar telefoner till direkta marknadsföringskanaler, vilka samtidigt personifierar och lokaliserar shoppingvanor, förväntas intäkterna för V-Commerce att överstiga 30 miljarder dollar (USD) år 2005 (The Kelsey Group 2000 enligt Nuance 2000).

Tillgängligheten av standardiserad teknologi och standardiserade produkter kommer att öka utvecklingen av Voice Web-tjänster på samma sätt som industristandarder möjliggjorde den snabba tillväxten av webben, enligt Nuance (2000).

Frågan är då varför tekniken inte haft större genomslag med tanke på att det finns en del röstteknologi redan på marknaden. En av anledningarna är att det är en fråga om trend. Teknologier blir trender baserade på faktorer som dess tillänglighet, mängden marknadspropaganda från fabrikanter och medias bevakning av dessa teknologier. Moderna erövrare inkluderar multimedia i början av 1990-talet och internet 1995 (Keyes 1997).

Andra anledningar är de begränsningar och svårigheter som finns med röstgränssnitt. Kunin (2000) vill gå så långt som att påstå att även de värsta *visuella* gränssnitten fyller en viss funktion, medan samtliga *röstgränssnitt*, förutom de allra bästa, kraftigt frustrerar och förvirrar användare. Kunin (2000) gör jämförelser mellan webbdesign och röstdesign och menar att det senare kräver mer expertis samt är svårare att utveckla. Dels på grund av användarens brist på tolerans mot systemet, men också eventuell motvilja från användarens sida att bruka dylika gränssnitt.

Det kan hända att en del människor upplever det som besvärande att prata med datorer, vilket kan ge en känsla av obehag och misstro. Istället för ett tillförlitligt system som inger förtroende, kan resultatet bli en vilsen användare som motsträvigt för en, enligt sig själv, komplicerad människa-

dator-interaktion. Det krävs en medvetenhet om de faktorer som påverkar kommunikationen mellan människor och datorer för att ett gränssnitt ska bli riktigt bra (Rosander & Magnusson 1998).

Det finns en viss tolerans mot datorsystem. I fönsterbaserade gränssnitt accepteras problem som till exempel för sakta eller för snabba dubbelklickningar, marginella felklickningar och så vidare. För en röstapplikation skulle en feltolkning ej accepteras. De företag som försöker ändra konsumenternas förväntningar kommer inte att lyckas eller accepteras på marknaden. Det är snarare företagen som får anpassa sig och sitt system efter förväntningarna. Röstteknologin måste uppnå en tillförlitlighet som är genomgående hög (98-99+ %). Det måste vara tillräckligt robust för att tåla ljudstörningar som till exempel på ett kontor eller i ett hem, där det aldrig är tyst. Idag finns dock effektiva ljudfilter (Keyes 1997).

Keyes (1997) trodde trots allt, vid tidpunkten för bokens publicerande, att det fanns stora möjligheter för röstrelaterade teknologier att bli nästkommande trend. Mycket beroende på lösta problem med att olika människor talar på olika sätt, olika dialekter och så vidare. Även professor Robin Cooper, datalingvist vid Göteborgs universitet (Claesson 2000), uppmärksammar problematiken kring lingvistiska frågor. Han betonar vikten av språkteknologi vid utveckling av röstsystem. Cooper menar, enligt Claesson (2000), att det krävs humanistisk forskning för att tekniken ska närma sig människan. Teknikutvecklare måste ta hjälp av språkvetare för att få igång samtal mellan människa och maskin. Det är språkvetarna som har kunskap om språkets komplexitet. Parfitt (2000) är av samma mening. Han säger (Parfitt 2000 sidan 48):

> *”Some of the most proficient writers of VoiceXML dialogs that I’ve worked with have come from a background in linguistics, audio engineering or the broadcast industry.”*

Enligt Parfitt (2000) krävs kunskap om vad som gör att den mänskliga dialogen fungerar för att kunna skriva bra VoiceXML-dialoger.

Telia startade den 11 januari 2001 en automatisk nummerupplysning, 118 888 AutoSvar ”Den talande telefonkatalogen”, som är ett samarbete mellan Respons AB och Philips Speech Processing (Telia 2001). Tjänsten är gratis under de första två månaderna, då det fortfarande ses som en utvecklingsperiod där varje inkommande samtal medverkar till justeringar av teknik och övrig utformning. AutoSvar kan ses som ett exempel på en fortskridande teknik och ett led i problemhanteringen av dialektala och liknande språkliga brydsamheter. I sitt pressmeddelande uppger Telia (2001) att:

> *”Den talande telefonkatalogen känner igen uttalet från de flesta privatpersoner över hela Sverige och svarar med att läsa upp telefonnumret.”*

Utöver dialektala differenser finns även svårigheten med olika språk. En förutsättning för att en röstapplikation överhuvudtaget skall vara intressant är att den stödjer det språk användaren själv känner sig bekvämast med att använda. För att få en känsla för vilken nivå befintlig röstteknologi befinner sig på, då inte bara språkligt sett utan även ur prestandasynpunkt, kan Nuance 7.0, vilket är Nuances röstigenkänningsmjukvara, tjäna som exempel. I skrivande stund stödjer Nuance 7.0, 20 olika språk, samt har en igenkänningstillförlitlighet på 96% (Nuance 2001). Nuance 7.0 tillhandahåller, enligt dem själva (Nuance 2001):

> *”unparalleled accuracy, scalability and reliability for voice-driven applications via the telephone.”*

Som ytterligare argument menar Keyes (1997) att processprestandan inte längre utgör något problem eftersom problematiken med att tolka hela fraser, istället för bara enstaka ord, är löst. En förutsättning är dock att medföljande restriktioner åtföljs, något som i sig inte är specifikt, då det för tangentbord och mus finns än fler restriktioner. Vidare anledningar till en ökad genomslagskraft är den teknologi som åtföljer *artificiell intelligens* (AI), då datorerna kan lära sig nya fraser och ord efterhand. Fraser och ord som med andra ord inte behöver vara lagrade från början (Keyes 1997).

Keyes (1997) talar dessutom om sensationen runt voice browsing, vars teknologi påverkas av marknadsförväntningar satta av filmindustrins Hollywood. Han menar att den vanliga användaren fortfarande har Star Trek-tankesättet där man endast behöver tala rakt ut i luften, som till en annan människa. Men han tror samtidigt också att det i framtiden kommer att bli en realitet. Förutom röstigenkänning som standard i persondatorer, talade Keyes (1997) till exempel om automatiserade hem och fordonsnavigering med rösten som kontrollverktyg. Han såg en framtid där användaren själv kan interagera med filmer och annan underhållning som om det skulle ha varit i en riktig studio. Ser man tillbaka på det som Keyes (1997) förutspådde 1997, ser man tydliga indikationer på att åtminstone delar av hans profetia förverkligats.

6 Slutsatser och sammanfattning

Sammanfattningsvis besvarar resultatet av vår undersökning ställda forskningsfrågor. Det vill säga, kombinationen XML och VoiceXML fungerar bra tillsammans och det är fullt möjligt att använda samma XML-fil för HTML och VoiceXML. Likheterna mellan HTML-lösningen och VoiceXML-lösningen är väldigt stora vilket medför att skillnaderna endast är marginella.

Möjligheten att besvara forskningsfrågorna blottlades vid vårt praktiska arbete där otaliga tester och försök blandades med en hel del faktasökande i fackböcker och diverse annan litteratur. Syncs mjukvarumiljö visade sig passa väl som plattform för vår undersökning och mycket av den kunskap vi förvärvat inhämtades via företagets kompetens.

Använda tekniker har samtliga bidragit med sina fördelar och vi tycker att vi lyckats ta vara på dem när tillfälle givits. Många av dessa fördelar har även varit en förutsättning och ett argument till varför vald teknik använts. Som exempel kan vi nämna använda modelleringstekniker för mockups och dataflödesdiagram. Dessa tekniker gav uttryck för dataflöden och processdialoger, hur de skulle se ut och hanteras. Med hjälp av modellerna kunde vi på ett tydligare sätt förmedla tankar och idéer även för varandra. De gav oss en gemensam bild över undersökningens fokus då vi tillsammans, kreativt men kritiskt, tog fram dem under ett antal diskussioner. Dessutom fungerade de även som resultatuppsamlare av analyser och diskussioner som kom upp och som pågick under arbetet. De användes även vid jämförelsefasen som underlag för analys av slutresultatet. Modellerna var enkla och gav en tydlig bild över respektive lösnings uppbyggnad och struktur.

Trial-and-error som undersökningsmetod är svårplanerad och kan därför verka lite ostrukturerad men det är en användbar och ofta brukad metod vid programmering. Det går inte att komma ifrån att det var en teknik som passade vår uppgift bra. Hela undersökningens mål var att testa och se om det gick att kombinera teori med praktik. Det vill säga, vi testade en funktion för att se om den fungerade. Fungerade det inte fick vi göra på ett annat sätt för att återigen testa och köra. Varje problem som dök upp var vi tvungna att hantera innan vi körde nästa test. Arbetsprocessen gick helt enkelt ut på att vi provade oss fram till lösningen.

Förutom att det varit en enorm utmaning och oerhört intressant att få arbeta med en relativt ny och oprövad teknik, gjorde det uppgiften än mer

spännande när målet var att få densamma att passa in i en fördefinierad kontext. Det finns mycket kvar att lära men inhämtad kunskap har varit tillräcklig för att kunna lösa uppgiften.

Undersökningen har visat oss VoiceXML:s många fördelar men också begränsningar. Tekniken har nog en bit kvar att gå men vi tror att språket har en framtid vid utveckling av röstbaserade gränssnitt. Avslutningsvis ser vi andra områden som kan vara intressanta att studera vidare, bland annat hur och i vilken utsträckning VoiceXML går att kombinera med andra interaktionstekniker, till exempel WAP. Ett annat intressant område är hur ett användbart röstgränssnitt fungerar, vilka faktorer som är viktiga att tänka på vid utvecklingen, till exempel användbarhetsaspekter som skiljer mellan visuella gränssnitt och röstgränssnitt.

Ordlista

API (*Application Programming Interface*) – Ett gränssnitt som gör det möjligt att i program och insticksmoduler utnyttja funktioner för vissa tjänster som finns tillgängliga i ett annat program eller i en funktionssamling.

CGI skript (*Common Gateway Interface*) – En skriptfil som exekveras på en webbserver som svar på en användarförfrågan. Skriptet används för åtkomst av dynamiska data, såsom data i en databas.

CSS (*Cascading Style Sheets*) – En stilmalls typ som används för att specificera presentationsformat och layout för strukturerade dokument, såsom HTML- eller XML-dokument. CSS är *inte* en dialekt av XML.

DTD (*Document Type Definition*) – Används för till exempel SGML- och XML-dokument och är en beskrivning av textens struktur i en viss dokumentkategori. DTD innehåller till exempel namn, information och regelverk för de så kallade märkorden och hur de får användas tillsammans.

DTMF (*Dual Tone Multi Frequency*) – Även kallad tonval (*touch-tone* på engelska) där inmatning sker med siffertangenterna på telefonen.

HCI (*Human-Computer Interaction*) – Även kallad Människa-Dator Interaktion (MDI) på svenska. HCI är en disciplin vilken behandlar design, utvärdering och implementering av interaktiva datorsystem avsedda för människor.

HTML (*Hyper Text Markup Language*) – Ett standardiserat märkspråk, baserat på SGML, för strukturering av information på bland annat hemsidor och i e-post. HTML tillåter till exempel länkar till andra hemsidor, så kallad hypertext, definiering av typsnitt, grafik och andra detaljer. HTML är en implementering av SGML.

HTTP (*HyperText Trasfer Protocol*) – Ett kommunikationsprotokoll särskilt utformat för distribution av hypertextdokument över internet. Protokollet förmedlar datapaket från en webbserver till en webbläsare.

Java – Ett plattformsoberoende enkelt, objektorienterat programmeringsspråk utvecklat av Sun Microsystems. Språket är utformat speciellt för att skriva distribuerade webbapplikationer.

JavaScript – Ett skriptspråk utvecklat av Netscape Inc. JavaScript kan inkluderas i ett HTML-dokument och exekveras av webbläsaren när dokumentet laddas.

JSP (*Java Server Page*) – Används för att dynamiskt sätta in data vid skapande av webbinnehåll. Innehållet är vanligtvis strukturerad text såsom till exempel HTML och XML. Genererar HTML-sidor i flykten.

N-tier – En applikation som kan delas upp i ett antal nivåer av logiska lager genom att använda en återanvändbar komponentbaserad lösning. Dessa logiska nivåer kan arbeta i olika konfigurationer, genom att använda ett antal fysiska system och på så sätt tillhandahålla obegränsad flexibilitet och skalbarhet till dynamiska affärskrav.

PDF (*Portable Document Format*) – Ett filformat som gör det möjligt att skicka dokument över internet så de ser likadana ut, oavsett datorutrustning. Med vanliga HTML-dokument kan man endast ge en rekommendation till hur dokumentet ska se ut hos mottagaren, men när det gäller blanketter, broschyrer med mera vill man kunna få ett exakt återgivande av originaldokumentet.

SGML (*Standard Generalized Markup Language*) – En standard för hur man specificerar ett märkspråk för dokument. SGML är inte i sig själv ett dokumentspråk, utan en definition av hur man specificerar ett sådant. Både HTML och XML är definierade som instanser av SGML.

Text-till-tal (*Text-To-Speech (TTS)* på engelska) – Med text-till-tal menas att text omvandlas till syntetiskt tal via ett TTS-system.

URI *(Uniform Resource Identifier)* – Även kallad *Uniform Resource Locator* (URL). URI är en utbyggbar adresseringsteknik för webben, och i skrivande stund, den enda i sitt slag. Det finns tre användbara typer av URI:er: URL, partiella URL:er och *Uniform Resource Names* (URN). Fungerar genom strängar som identifierar olika källor på webben som till exempel dokument, bilder, nedladdningsbara filer, tjänster och elektroniska postlådor. Strängarna gör källorna tillgängliga via en mängd varierande namngivnings scheman och accessmetoder som till exempel HTTP och FTP (*File Transfer Protocol*).

ViaVoice – En röstapplikation utvecklad av IBM.

Voice browsing – Ett nytt interaktionssätt där man kan använda tal som indata i samband med bland annat internetbaserad teknik och till exempel få syntetiskt tal som utdata.

VoiceXML (*Voice eXtensible Markup Language*) – En ny W3C-standard som används bland annat för att göra information på internet tillgängligt via röst och telefon. Men fungerar även på vanliga standardapplikationer som till exempel MS Word för att addera röstnavigeringsstöd.

VoiceXML Forum – Ett forum grundat av fyra företag; AT&T, IBM, Lucent och Motorola, för att utveckla VoiceXML.

W3C (*World Wide Web Consortium*) – En organisation vilken försöker främja användandet av internationella standarder vid mjukvaruutvecklingen av internetprodukter. Många gemensamma standarder som exempelvis XML och HTML, har utvecklats och utvecklas W3C överinseende. Idag är nästan alla stora företag som arbetar mycket med internet medlemmar, däribland Microsoft och Netscape (Åström 2000).

WAP (*Wireless Application Protocol*) – Ett protokoll för trådlös överföring av data som sedan kan visas i till exempel ett mobiltelefonfönster.

WWW (*World Wide Web*) – Internets världsvida HTML-baserade informationssystem. Webben är hypertextbaserad och man navigerar genom att klicka på ord eller fraser, varvid man kommer till en ny informationssida.

XML (*eXtensible Markup Language*) – Ett utbyggbart språk för märkning av text. XML kan genom denna förmåga användas för att beskriva och strukturera information eller data. Detta medför att en dator kan tolka innebörden i informationen genom att man vid inmatningen beskrivit vad det är. XML kan ses som en förenklad variant av SGML speciellt anpassad för användning på Internet. Precis som VoiceXML är en XML W3C standard.

XSL (*eXtensible Stylesheet Language*) – Ett stilmallsspråk och W3C standard som används för att formatera XML till presentationsformat. Med andra ord en typdeklaration utformat att specificera regler för hur innehållet av en XML-dokument ska omstruktureras (transformera en instans av ett XML-dokument till en annan instans) och för att specificera presentationsformatet för ett XML-dokument.

XSLT (*eXtensible Stylesheet Language Transaction*) – XSLT är ett språk att transformera ett XML-dokument till ett annat XML-dokument. XSLT är utvecklat för att användas som en del av XSL. XSL specificerar ett XML-dokuments utseende genom att använda XSLT för att beskriva hur dokumentet transformeras till ett annat XML-dokument som använder formateringsvokabuläret.

Bilagor

Bilaga I – Tradpart_a.xsl

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="vxml" indent="yes"/>
<xsl:strip-space elements="*"/>
<xsl:template match="/">

<vxml version="1.0">

    <form id="TradingPartnerInput">
        <block>Welcome to the Sync application.</block>
            <field name="trad">
            <prompt>What trading partner?</prompt>
            <grammar>
                <!--  Add all trading partner names to grammar -->
                <xsl:for-each select="/RESULTS/TRADPAADR_TD/ROW">
                    <xsl:apply-templates select="TDNAME" />&#124;
                </xsl:for-each>
            </grammar>
            <catch event="help">
                Please speak the name of the trading partner
                for which you want the phone number.
            </catch>
            </field>
        </form>
</vxml>

</xsl:template>
<xsl:template match="*">
<xsl:apply-templates/>
</xsl:template>
<xsl:template match="text()">
<xsl:value-of select="."/>
</xsl:template>
</xsl:stylesheet>
```

Bilaga II – Tradpart_b.xsl

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="vxml" indent="yes"/>
<xsl:strip-space elements="*"/>
<xsl:template match="/">

<vxml version="1.0">

  <form id="TradingPartnerAnswer">
      <block>
          <xsl:variable name="lsTRAD" select="'Volvo Construction Equipment'" />
          <xsl:for-each select="/RESULTS/TRADPAADR_TD/ROW">
              <xsl:variable name="lsTDNAME">
                  <xsl:apply-templates select="TDNAME"/>
              </xsl:variable>
              <xsl:if test="$lsTRAD = $lsTDNAME">
                  <xsl:variable name="lsTDPHONE">
                       <xsl:apply-templates select="TDPHONE" />
                  </xsl:variable>
                  <prompt>The phone number to <value expr="'{$lsTDNAME}'"/>
                            is <value expr="'{$lsTDPHONE}'"/>.
                  </prompt>
              </xsl:if>
          </xsl:for-each>
      </block>
  </form>
</vxml>

</xsl:template>
<xsl:template match="*">
<xsl:apply-templates/>
</xsl:template>
<xsl:template match="text()">
<xsl:value-of select="."/>
</xsl:template>
</xsl:stylesheet>
```

För tydlighetens skull gavs variabeln lsTRAD värdet av strängen ’Volvo Construction Equipment’ för att påvisa att handelspartnerns namn ligger i denna variabel. Annars ska detta värde komma som en inparameter ifrån JSP-filen.

Bilgaga III – Tradpart_a.vxml

```
<?xml version="1.0" ?>
<vxml version="1.0">
  <form id="TradingPartnerInput">
        <field name="trad">
            <prompt>What trading partner?</prompt>
            <grammar>TransAdr| eTekram AB| eTekram AB| eTekram AB| eTekram AB|
                Semper AB|Findus AB| Scan AB| Mat Gross AB| Snabb Gross AB|
                Kalles Snabblivs AB| Nisses Handelsbod AB| Semper AB|
                Findus AB| Scan AB| Mat Gross AB|Snabb Gross AB| Snabblivs AB|
                Mat Gross AB| Snabb Gross AB| Snabblivs AB| Nisses Handelsbod AB|
                TransAdr|Pan-Man Inc.| Pan-Man Inc.| Pan-Man Inc.| Mat Gross AB|
                Matgross AB| Nisse| Kalle| Olle| Volvo Construction Equipment|
                TransAdr|TransAdr| TransAdr|TransAdr| TransAdrXX| TransAdr|
            </grammar>
        <catch event="help">Please speak the name of the trading partner
            for which you want the phone number.
        </catch>
      </field>
  </form>
</vxml>
```

Bilaga IV – Tradpart_b.vxml

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

    <form id="TradingPartnerAnswer">
      <block>
            <prompt>The phone number to <value expr="'Volvo Construction Equipment'" />
            is <value expr="'020-878845'" />.</prompt>
       </block>
    </form>

</vxml>
```

Referenser

Andersen E S (1991), *Systemutveckling – principer, metoder och tekniker*. Andra upplagan. Studentlitteratur, Lund, Sverige.

Andleigh P K och Thakrar K (1996), *Multimedia systems design*. Prentice Hall Inc, Upper Saddle River, New Jersey, USA.

Andréasson C, Melvanius A och Stenberg F (1999), *XML – möjligheter och problem för företag*. Kandidatuppsats INF99-038, Institutionen för Informatik, Lunds Universitet, Sverige.

Alter S (1996), *Information system – a management perspective.* Andra upplagan. The Benjamin/Cummings Publishing Company Inc. Menlo Park, California, USA.

Avison D E & Fitzgerald G (1988), *Information Systems Development – Methodologies, Techniques and Tools.* Blackwell Scientific Publications, Oxford, England.

Baiada M (1997), *World Wide Web Application Development* (Ur Keyes 1997).

Bass L, Clements P & Kazman R, *Software Architechture in Practice*. Addison Wesley Longman Inc, Reading, Massachusetts, USA.

Claesson P (2000), Talande maskiner kräver humanistisk forskning. Ur *Computer Sweden*, 3 november 2000, Årgång 18, Nr 105. Tryck & Media, Sverige.

Fluckiger F (1995), *Understanding network multimedia applications and technology*. Prentice Hall, Padstow, Storbritannien.

Graham I S & Quin L (1999), *XML Specification Guide*. John Wiley & Sons Inc, New York, USA.

Kelsey Group, The (2000), *Foundations of Speech Portals*. Report February 2000. Enligt Nuance (2000).

Keyes J, red (1997), *Ultimate Multimedia Handbook*. Andra upplagan. The McGraw-Hill Companies Inc, New York, USA.

Keizer G (2000), *CNET Review (10/17/00) IBM ViaVoice Pro 8.0*. WWW-dokument, 2000-10-19. URL http://www.cnet.com/software/0-3731-8-2938624-4.html?tag=st.sw.3731-8-2938624-3.DIR.3731-8-2938624-4.

Kunins J (2000), *Ask The Expert – Topics: VoiceXML and Voice Applications*. WWW-dokument, 2000-12-10. URL http://www.wirelessdevnet.com/channels/voice/expert/.

Lotsson A (2000), Internet är som solen. Ur *Computer Sweden*, 13 oktober 2000, Årgång 18, Nr 96. Tryck & Media, Sverige.

Löwgren J & Stolterman E (1998), *Design av informationsteknik – materialet utan egenskaper*. Studentlitteratur, Lund, Sverige.

Nuance (2000), *SpeechObjects™ & VoiceXML*. Menlo Park, Kalifornien, USA. White Paper. PDF-dokument, URL http://www.nuance.com.

Nuance (2001), *Nuance 7.0, Powerful Speech Recognition for Transactions over the Telephone*. WWW-dokument, 2001-01-17. URL http://www.nuance.com/index.htma?SCREEN=nuance7.

Martin D (2000), *Adapting Content for VoiceXML*. O'Reilly & Associates, Inc. WWW-dokument, 2000-08-23. URL http://www.xml.com/lpt/a/2000/08/23/didier/index.html.

Parfitt R (2000), Voice interaction for the Web – Speech recognition applications are making major breakthroughs. *XML Journal*, Volym 1 Utgåva 5. SYS-CON Publications Inc, New Jersey, USA.

Preece J, Rogers Y, Sharp H, Benyon D, Holland S & Carey T (1994), *Human-Computer Interaction*. Addison-Wesley Longman Ltd, Essex, England.

Rational Software Corporation (2000), *Rational Unified Process*. Version 2001.03.00. IBM Corporation, USA.

Rosander C & Magnusson L (1998), *Rösten – det naturliga kommunikationsmedlet?* Magisteruppsats INF98-039, Institutionen för Informatik, Lunds Universitet, Sverige.

SIG Security (2000), *E-delegring – roller och rättigheter*. Studentlitteratur, Lund, Sverige.

Svenning C (1997), *Metodboken*. Andra upplagan. Lorentz Förlag, Sverige.

Svenska Datatermgruppen (2000), *Term- och språkmaterial version 18, 14 september 2000*. WWW-dokument, 2000-11-22. URL http://www.nada.kth.se/dataterm/.

Telia (2001), *Den talande telefonkatalogen är här*. 2001-01-11. Arkiv för pressmeddelanden. WWW-dokument, 2001-01-17. URL http://www.telia.se/bvo/omdirigering/tvaramar.jsp.html?refoid=23241&body=http%3a%2f%2fhan16ns.telia.se%2fTelia%2fThk%2fThkPre52.nsf%2fvNyhetEfocus%2fB8723510F6994173412569D1003A84E2%3fOpenDocument+.

Turban E och J E Aronson (1998), *Decision Support Systems and Intelligent Systems.* Femte upplagan. Prentice-Hall Inc, Upper Saddle River, New Jersey, USA.

VoiceXML Forum (2000a), *Voice eXtensible Markup Language – VoiceXML*. Version 1.00, 7 Mars 2000. PDF-dokument.

VoiceXML Forum (2000b), *Voice eXtensible Markup Language*. WWW-dokument, 2000-10-24. URL http://www.voicexml.org.

W3C (2000a), *XML Pointer Language (XPointer) Version 1.0 W3C Last Call Working Draft 8 January 2001*. WWW-dokument, 2001-01-16. URL http://www.w3.org/TR/WD-xptr.

W3C (2000b), *Voice eXtensible Markup Language (VoiceXML™) version 1.0, W3C Note 05 May 2000*. WWW-dokument, 2000-12-03. URL http://www.w3.org/TR/voicexml.

W3C (2000c), *W3C XML Pointer, XML Base and XML Linking*. WWW-dokument, 2000-11-27. URL http://www.w3.org/XML/Linking.html.

W3C (2000d), *XML Schema Part 0: Primer. W3C Candidate Recommendation 24 October 2000*. WWW-dokument, 2001-01-15. URL http://www.w3.org/TR/xmlschema-0/.

Wireless Developer Network (2001), *An Introduction To VoiceXML*. WWW-dokument, 2001-01-10. URL http://www.wirelessdevnet.com/channels/voice/training/voicexmloverview.html

Yin R K (1984), *Case Study Research – Design and Methods*. SAGE Publications Inc, Beverly Hills, USA.

Åström P (2000), *XML - Extensible Markup Language*. Elanders Graphics Systems AB, Göteborg, Sverige.

Muntliga referenser

Andersson C-J (2000), Teamleader för HCI-teamet på Sync Business Process Integration AB. Även teamleader och ansvarig för Strategic Development.

Ottosson A (2001), VD för Sync Business Process Integration AB.