

Utveckling av en röststyrd uppläsningstjänst för e-post

Maria Landberg
marland@stp.ling.uu.se

Examensarbete i datorlingvistik
Språkteknologiprogrammet
Institutionen för lingvistik och filologi · Uppsala universitet

6 december 2004

Handledare:
Mats Dahllöf, Uppsala universitet
Jaan Kaja, Icepeak AB

Sammandrag

Syftet med detta examensarbete är att utveckla en röststyrd applikation för uppläsning av e-post till Icepeak AB:s röststyrda telefonist Icepeak Attendant. Arbetet har bestått av att utveckla ett röststyrt gränssnitt för styrning av systemet samt att modifiera en redan befintlig preprocessor för bearbetning av den inkommande e-posten. Såväl det röststyrda gränssnittet som preprocessorerna är implementerade i C++.

I den utvecklade prototypen ringer användaren upp Icepeak Attendant och kan därefter få de önskade e-postmeddelandena upplästa i telefonen. Applikationen börjar med att läsa upp hur många olästa brev användaren har, vilka avsändarna är samt ärendet på breven. Användaren får sedan själv bestämma vilka brev som ska läsas upp och kan under uppläsningen när som helst avbryta uppläsningen, gå vidare till nästa eller föregående brev, radera ett brev, få information repeterad eller pausa uppläsningen. Användaren kan också söka efter en specifik avsändares e-post genom att säga namnet på avsändaren.

Uppläsningstjänsten har testats av tre användare för att få synpunkter på systemets nuvarande utformning och uppslag till vad som bör beaktas vid eventuella framtida förbättringar. I den mån det har varit möjligt har de problem som identifierats i utvärderingen tagits fasta på och ändrats i systemet. Utvärderingen visar dock att systemets funktioner generellt fungerar väl, systemet är lätt att använda och användarna har inga problem med att få de önskade breven upplästa.

Innehåll

Förord	ii
1 Inledning	1
1.1 Syfte	1
1.2 Disposition	1
2 Bakgrund	2
2.1 Liknande existerande lösningar	2
2.2 Det röststyrda användargränssnittet	2
2.2.1 Nuance mjukvaruplattform	2
2.2.2 Dialoger och taligenkänning	2
2.3 Standard för e-poststruktur	5
2.3.1 RFC 822 och MIME	5
2.3.2 HTML i e-post	7
3 Genomförande	8
3.1 Det röststyrda användargränssnittet	8
3.1.1 Översikt	8
3.1.2 Dialog	8
3.1.3 Utveckling av grammatik	9
3.1.4 Talsyntes	9
3.1.5 Implementering av dialogtillstånd	10
3.2 Preprocessor - en bearbetning av den inkommande e-posten	12
3.2.1 Den ursprungliga preprocessorn	13
3.2.2 Förändringar av den ursprungliga preprocessorn	13
4 Utvärdering	15
4.1 Utvärderingsmetod	15
4.2 Tillvägagångssätt	15
4.3 Användarnas synpunkter	16
5 Resultat	18
5.1 Förändringar av systemet utifrån utvärderingsresultatet	19
5.2 Diskussion	19
5.3 Utvecklingsmöjligheter	20
Litteraturförteckning	21
Bilagor	22

Förord

I denna uppsats redovisas ett examensarbete inom Språkteknologiprogrammet vid Institutionen för lingvistik och filologi, Uppsala universitet. Examensarbetet har utförts hos Icepeak AB i Stockholm under sommaren och hösten 2004.

Jag vill här uppmärksamma några personer som på olika sätt hjälpt mig under arbetets gång. Jag vill tacka min handledare på institutionen för lingvistik och filologi vid Uppsala universitet, Mats Dahllöf, som har kommit med värdefulla synpunkter kring uppsatsens innehåll och utseende. Ett tack går också till de på Icepeak AB som på olika sätt givit mig praktisk vägledning under arbetets gång; min handledare Jaan Kaja, Rehan Mirza, Hans Runehov och Ulrika Olsson samt Tim Hansen som gav förslag till ämnet. Slutligen vill jag också tacka min sambo och familj för uppmuntran och korrekturläsning samt de personer som ställt upp som testanvändare.

Tack!

Maria Landberg
Uppsala, november 2004

1 Inledning

Ökade krav på mobilitet och tillgänglighet av data i samhället skapar behov av nya mobila lösningar. Idag finns inte längre några hinder för att använda tal i användargränssnitt. Forskningen inom området har varit intensiv och datorkapaciteten är numera tillräcklig för att funktionella system skall kunna byggas till en rimlig kostnad. Fördelarna med röststyrda applikationer är många, de är effektiva, tillgängliga dygnet runt och kostnadseffektiva. De kan dessutom vara användarvänliga, att navigera i systemet med rösten är relativt smidigt för såväl nybörjaranvändare som avancerade användare. Alla kommandon är till exempel tillgängliga samtidigt utan att menyer måste gås igenom.

Den här uppsatsen beskriver utvecklingen av en prototyp för röststyrd e-postuppläsning. Användaren är med denna applikation inte längre hänvisad till en uppkopplad dator eller en liten handdator- eller mobiltelefonskärm för att kunna få tillgång till sin e-post. Den röststyrda tjänsten tillåter användaren att lyssna av sina meddelanden dygnet runt och överallt, även i de situationer han eller hon inte har ögon eller händer fria.

1.1 Syfte

Syftet med examensarbetet är att utveckla en röststyrd applikation för uppläsning av e-post till Icepeak AB:s röststyrda telefonist Icepeak Attendant. Systemet innefattar idag, förutom in- och utgående telefoni, även tjänster som hänvisning av telefon, bokning av konferensrum och bilar med mera.

I den utvecklade prototypen ska användaren kunna ringa upp Icepeak Attendant för att få e-postmeddelanden upplästa i telefonen. För att systemet skall bli funktionellt ska användaren kunna kontrollera uppläsningen genom röststyrda kommandon. Arbetet innefattar två delar som skall länkas ihop; utveckling av ett röststyrt gränssnitt mellan användare och system och vidareutveckling av en befintlig preprocessor för bearbetning av den inkommande e-posten. En mindre utvärdering ska också göras.

1.2 Disposition

Uppsatsen består av fem kapitel, där kapitel två utgör bakgrunden för de följande kapitlen. Uppbyggnad av röstgränssnitt och e-poststruktur presenteras här teoretiskt, kapitlet tar också upp några liknande existerande lösningar som idag finns på den svenska marknaden. Kapitel tre beskriver hur examensarbetet har genomförts samt kodningen av röstgränssnitt och e-postprocessor. Det följande kapitlet utgörs av en mindre användarutvärdering. Uppsatsen avslutas med resultat samt diskussion och framtida utvecklingsmöjligheter.

2 Bakgrund

2.1 Liknande existerande lösningar

Idag finns redan några e-postuppläsningssystem på marknaden, varav två som stödjer svenska språket kort kommer att presenteras här. Phoneticom AB:s 'Tele-post' och Ericsson AB:s 'OneBox' läser båda upp e-postmeddelanden över telefon med hjälp av talsyntes.

Phoneticoms tjänst läser upp användarens e-post och eventuella bilagor i telefonen och tillåter användaren att skicka e-post i form av ljudfiler samt att modifiera sin adressbok. Tjänsten är dock inte röststyrd utan styrs via telefonens knappsats (Phoneticom AB:s hemsida).

Ericssons 'OneBox' är ett system som tar hand om både e-post, 'voicemail' och fax. Systemet tillåter, förutom vanlig användning via dator, att användaren använder sig av en telefon för att lyssna av e-post och 'voicemail' (Ericsson AB:s hemsida).

2.2 Det röststyrda användargränssnittet

2.2.1 Nuance mjukvaruplattform

Den teknologi som använts för taligenkänning i uppläsningssystemens gränssnitt är utvecklad av Nuance Communications Inc. Nuance, som är en världsledande leverantör av produkter för taligenkänning och talarverifiering utvecklar, marknadsför och stödjer en mjukvaruplattform som möjliggör att information och tjänster för företag och telekomoperatörer kan nås via telefon. Teknologin stödjer närmare 30 olika språk, däribland svenska, norska och danska (www.nuance.com).

Tekniken går i korthet ut på att användarens röstkommandon tolkas och jämförs med de kommandon som definierats i en applikationsspecifik grammatik. Beroende på vad som matchas i grammatiken utför systemet olika åtgärder.

Utöver Nuance så har även till exempel Loquendo (www.loquendo.com) och ScanSoft (www.scansoft.com), som köpt upp gamla Lernaut & Hauspie:s plattform, lösningar som stödjer svenska språket.

2.2.2 Dialoger och taligenkänning

För att göra ett röststyrt gränssnitt, Interactive Voice Response (IVR) system, behövs indata, utdata och en dialog. Eftersom tal är ett långsamt sätt att presentera information på är det viktigt att informationen är kort och koncis, samt att inte för mycket information presenteras. Att systemet ger korrekt och välvägd

återkoppling är också viktigt. Följdfrågor bör endast ställas efter kommandon som kan förstöra information eller liknande eftersom felhanteringsmeddelanden som blir alltför frekventa upplevs som irriterande (Acero et al 2001).

Eftersom användaren inte kan förlita sig på synen och möjligheten att gå tillbaka och läsa om eventuell otydlig information är ett intuitivt gränssnitt som guidar användaren genom systemets olika funktioner av största vikt. Ett viktigt led i utvecklingen av ett sådant system är att bestämma vilken information systemet ska ge användaren, vilken information systemet behöver från användaren och sist men inte minst, förutspå vilka kommandon användaren kommer utnyttja för att förmedla informationen. Dessa förberedelser är nödvändiga för att en optimal grammatik över kommandon i systemet skall kunna skrivas (Acero et al 2001).

Genom att göra ett flödesschema skapas en överblickbar bild av dialogen mellan användaren och systemet, vilket också underlättar testning på potentiella användare genom exempelvis Wizard-of-Oz metoden. Efter dessa steg är det viktigt att snabbt skapa en prototyp av systemet för att kunna fortsätta testerna och göra användarutvärderingar av systemet (Acero et al 2001).

Promptar

Information som ges från systemet kallas 'promptar' och består av förinspelade ljudsegment (wave-filer). De designas för att antingen forma en 'mixed-initiative dialogue' med öppna frågor ('Vad vill du göra nu?') som klarar av att hantera mycket information på samma gång eller en 'directed dialogue' som styr dialogen och endast ger användaren ett antal alternativ att välja på ('För att lyssna på nästa brev säg nästa, för att lyssna på föregående'... osv). I detta examensarbete används 'directed dialogue' eftersom promptarna blir tydligare. Detta underlättar för en användare eftersom det inte uppstår några tvivel om vilken information som efterfrågas av systemet. Denna typ av dialog underlättar även utformandet av grammatiken genom att antalet användarkommandon som skall förutspås minskar väsentligt. Dialogtypen upplevs dock som mindre naturlig i och med att användarens valfrihet minskar (Nuance 2001).

Promptarna spelas upp samtidigt som igenkänning av röstkommandon görs. Det innebär att information från systemet inte måste höras till slut innan användaren kan fortsätta. Tekniken som kallas 'barge-in' underlättar för vana användare eftersom användaren tillåts att ge kommandon under tiden systemet läser information (Nuance 2001).

Grammatik

Efter att dialogen mellan användare och system har skissats kan grammatikskrivandet påbörjas. En grammatik beskriver de fraser som Nuance har som grund för jämförelse med användarens röstkommandon. Detta skrivs i GSL (Grammar Specification Language).

Det finns två typer av grammatiker, statiska och dynamiska. Statiska kan delas upp i topp- och subnivågrammatiker där toppnivågrammatiken kan anropas från en applikation under körning eller från andra grammatiker. Subnivågrammatikerna kan endast anropas från andra grammatiker. Toppnivågrammatikens namn börjar med en punkt följt av minst en versal medan subnivågrammatikens namn endast måste inledas med minst en versal. Nedanstående exempel är en subgrammatik som består av grammatiknamnet på första raden, en referens till en annan subgrammatik på andra raden samt ett antal rader som motsvarar de tänkta användarkommandona.

```

EMAIL_REPEAT
((?HESITATION)
[
  repetera
  upprepa
  (?förlåt jag [hörde förstod] inte)
  (en gång till)
  (?om igen)
])

```

Syntaxen är relativt enkel; hakparenteser motsvarar disjunktioner, parenteserna konkatenering och frågetecknet betyder optionalitet. Det medför att bland annat följande kommandon accepteras:

```

repetera
jag hörde inte
förlåt jag förstod inte
igen

```

Subgrammatiken med namnet `HESITATION` på rad två fungerar som en så kallad 'filler', det vill säga en utfyllnad i förhållande till de kärnkommandon som finns definierade i grammatiken. Fillers kan bestå av ord som användaren lägger till för att göra ett kommando artigare eller som i detta fall består av diverse tvekljud från användaren medan han eller hon funderar. Dessa ligger i en speciell grammatik för att kunna återanvändas i fler grammatiker. Grammatiken `HESITATION` gör att även frasen *öhhh jag hörde inte* skulle accepteras av tjänsten. Även ordet *förlåt* i grammatiken ovan är en filler eftersom den inte innehåller någon för systemet viktig information.

Beroende på vilken av grammatikerna som aktiveras av användarens kommandon så fyller Nuance en variabel som kallas 'slot' med olika värden. Om användaren skulle säga något av kommandona i ovanstående exempel får alltså slot-variabeln `email` värdet `repeat`.

```

.NAVIGATE_EMAIL
((?HESITATION)
[
  EMAIL_REPEAT          {<email "repeat">}
  EMAIL_READ            {<email "read">}
  [ ... ]
])

```

Dynamiska grammatiker används i de fall där det inte är lämpligt att explicit definiera alla möjliga kommandon i en grammatik, exempelvis tidsangivelser, orter eller när användarens kommandon beror av något utanför systemet. En dynamisk grammatik behöver inte kompileras utan skapas när den behövs och varar i detta fall bara så länge som samtalet varar. Variabeln `name` i exemplet nedan får det värde som returneras av `DynamicSenderName` och fyller sedan slot-variabeln `sender_name` med värdet. För en utförlig beskrivning av dynamiska grammatiker hänvisar jag till Olsson (2004).

```
EMAIL_READ
[
  [ ... ]
  DynamicPersonalName:name
]

{<sender_name &name>}
DynamicSenderName:dynamic
```

För att testa en grammatik efter kompilering finns ett antal kommandon att tillgå. Jag har använt två; en parser som visar de grammatiker som aktiveras av ett givet kommando och ett genereringsverktyg som slumpmässigt genererar kommandon som accepteras av systemet och som därmed underlättar arbetet med att åtgärda övergenereringar (Nuance 2001).

2.3 Standard för e-poststruktur

E-postsystem är uppdelade i ett antal individuella delar:

- 'E-postreläet', förflyttar ett e-postmeddelande mellan två olika komponenter.
- Registret, tillåter användaren att kolla upp andra e-postadresser etc.
- Postkontoret, lagrar e-posten så att en användare som inte är uppkopplad i det ögonblick som brevet anländer kan ta emot det senare.

För att ovanstående delar ska kunna kommunicera behövs ett gemensamt språk, ett protokoll som både sändande och mottagande komponent kan tyda. Det innebär att innehållet kodas innan det kan skickas och avkodas när det tagits emot. För att skicka e-post används i första hand två typer av protokoll, RFC¹ 822 och MIME (Rhoton 2000).

2.3.1 RFC 822 och MIME

RFC 822 är standardformatet för e-post. Benämningen är egentligen 'Standard for the Format of AR-PA Internet Text Messages', men RFC 822 används vanligtvis. Det finns dock problem med RFC 822, standarden tillåter inte 8-bitars tecken vilket innebär att man till exempel inte kan skriva svenska tecken. Engelska är ett av mycket få språk som endast använder 7-bitars tecken. Standarden klarar inte heller av att skicka flera filer och särskilja dem samt tillåter inte att exekverbara filer skickas. För att åtgärda dessa problem har ytterligare några RFC:er tillkommit med nummer 2045 till 2049. Dessa RFC:er går under samlingsnamnet MIME - Multipurpose Internet Mail Extensions (Rhoton 2000, Palme 1995).

¹RFC - Request for Comment. Alla föreslagna internetstandarder kallas till en början för RFC:er när de remitteras för utredning. Även efter att de gått igenom som standarder refereras de ofta till genom RFC-numret.

RFC 822 - struktur

RFC 822-brev består huvudsakligen av en så kallad 'header' och en så kallad 'body', separerade av en blankrad. Header-sektionen består av ett antal header-fält av typen 'namn: värde'. Det finns ett tjugotal definierade fält varav tre är obligatoriska: From, Date samt en av To/Cc/Bcc. Listan är dock inte sluten, egna kan läggas till vilket ofta sker exempelvis i MIME-applikationer. Dessa 'headers' brukar inledas med ett 'X-' för att undvika krockar med eventuella andra standarder. Body-sektionen är mycket enkel, enda kravet är att 7-bitars ASCII-tecken ska användas och att raderna inte överstiger 80 tecken (Rhoton 2000).

```
Date           : 27 Aug 04 0932 PDT
From           : Kalle Karlsson <KKarlsson@adress.se>
Subject        : RFC syntax
Sender         : LLarsson@adress2.se
Reply-To       : Lisa_Larsson@adress3.se
To             : OlleO@adress4.com
Cc             : Stina <StinaS@adress4.com>
Comment        : Lisa Larsson är bortrest, jag tar hand om hennes
                  e-post under tiden.
X-special-action: Exempel på användardefinierade headers.
Message-ID     : <1234.567.xzXZ-@Host>
```

MIME - struktur

MIME skapades för att komma tillrätta med de problem som fanns i RFC 822. Det nya formatet skulle vara kompatibelt med RFC 822, medge maximal läslighet för användaren och tillhandahålla utvecklingsbara och intuitiva definitioner av body-sektionen. Formatet bygger i grunden på RFC 822 men har förbättrats. Ett antal 'headers' har lagts till utöver de tidigare definierade. Dessa ignoreras av de e-postläsare som inte stödjer formatet. Nytt är också att formatet definierar en struktur för body-sektionen. Icke-MIME läsare klarar inte av att parse en MIME-body men kan visa 'body'n' i ett oformaterat textformat vilket gör att användaren trots allt har möjlighet att läsa texten (Rhoton 2000).

MIME-brev är precis som RFC 822-brev uppdelade i 'header' och 'body' med hjälp av en blankrad. I 'headern' hittar man de obligatoriska fält som nämndes ovan samt ett extra, 'Content-Type', som tas upp nedan. 'Body'n' delas upp i olika delar som åtskiljs med hjälp av gränsmarkeringar som definieras i fältet 'boundary'. Första delen är ett slags prolog som innehåller information om typen på brevet till icke MIME-läsare. MIME-läsare ignorerar däremot detta textstycke. Efter den kommer en eller flera delar som inleds med en rad om typ, 'Content-Type' och kodningsformat, 'Content-Transfer-Encoding'. Content-type-fältet anger vilken typ av brev som skickats, till exempel 'text/plain' för textmeddelanden, eller att brevet består av flera delar, till exempel 'multipart/mixed' eller 'multipart/alternative'. Eftersom RFC 822 endast klarar 7-bitars tecken konverterar MIME 8-bitars data till 7-bitars med hjälp av två olika format, 'Base64' och 'QuotedPrintable'. Den senare används oftast för text. Fältet för kodningstyp specificerar vilket format som använts om konvertering skett. Avslutningsvis kan en epilog som innehåller liknande information som prologen finnas, också denna ignoreras av MIME-läsare. Brevet avslutas med en 'boundary' med två extra avslutande bindestreck, se exempel på sida 7 (Rhoton 2000, Palme 1995).

2.3.2 HTML i e-post

I och med att MIME-formatet introducerades blev det möjligt att skicka HTML-formaterade e-postmeddelanden. Genom att i header-fältet 'Content-type' specificera att brevet är i HTML-format (text/html) vet e-postprogrammet hur koden ska tolkas och kan därmed presentera brevet som avsändaren tänkt sig. Dock finns det fortfarande program som inte klarar av att visa HTML-formaterade sidor, exempelvis Pine. För att komma till rätta med detta problem skickas HTML-brev tillsammans med ett vanligt textmeddelande. Mottagarens e-postprogram väljer vid mottagandet att visa den version som stöds. I nedanstående exempel ses ett sådant brev. I headerfältet Content-Type ses att brevet är av typen multipart/alternative, det vill säga är tillgängligt i både HTML- och text-format. Efter det kommer prologen och de två delarna, en av text-typ (text/plain) där endast den rena meddelandesträngen finns och en som är av HTML-typ (text/html) där samma textsträng ligger insprängd i HTML-taggar. För att HTML-delar skall kunna identifieras som just HTML inleds de med en slags tagg (<!DOCTYPE HTML PUBLIC "[...] ">) som deklarerar typen. (Berners-Lee & Connolly 1995, Borenstein & Freed 1993)

```
...
MIME-Version: 1.0
Content-Type: multipart/alternative;
                boundary='''-----=_NextPart_000_1234_01ABCD23.4E5F6G7H
```

This is a multi-part message in MIME format.

```
-----=_NextPart_000_1234_01ABCD23.4E5F6G7H
Content-Type: text/plain; charset="iso-8859-1"
Content-Transfer-Encoding: 7bit
```

Ett HTML mail.

```
-----=_NextPart_000_1234_01ABCD23.4E5F6G7H
Content-Type: text/html; charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

<!DOCTYPE HTML PUBLIC "-//W3C//DTD W3 HTML//EN">
<HTML>
<HEAD>
<META content=3Dtext/html; charset=3Diso-8859-1 =
http-equiv=3DContent-Type>
<META content=3D'"MSHTML 4.72.3110.7"' name=3DGENERATOR>
</HEAD>
<BODY>
<DIV>Ett <STRONG>HTML</STRONG> mail.</DIV>
<DIV> </DIV></BODY></HTML>
```

```
-----=_NextPart_000_1234_01ABCD23.4E5F6G7H--
```

3 Genomförande

3.1 Det röststyrda användargränssnittet

3.1.1 Översikt

De huvudsakliga delarna i det röststyrda gränssnittet som kommer att tas upp i de följande kapitlen är:

- Dialog
- Grammatik
- Talsyntes
- Implementation

Dialogen är helt beroende av systemets funktioner i och med att funktionerna bestämmer vilken information som systemet ger och vilka kommandon som en användare därmed kan tänkas vilja bruka. Dessa styr i sin tur grammatiken som definierar de kommandon som systemet skall känna igen. Syntesen används av systemet för sådan information som inte kan spelas in i form av promptar. Funktionerna i systemet motsvarar tillstånd i implementeringen som är gjord i C++.

3.1.2 Dialog

För att kunna styra en applikation som denna krävs ett antal grundläggande funktioner. Användaren måste kunna hoppa mellan breven, välja vilka brev som skall läsas upp samt avbryta uppläsningen. En hjälpfunktion är också nödvändig för ovana användare. Utöver dessa grundläggande funktioner ska också möjlighet att radera, repetera och söka brev samt att pausa uppläsningen finnas. Dessa funktioner styr dialogen i och med att de bestämmer vilken information systemet ska ge och därmed de kommandon som systemet skall reagera på.

Dialogen mellan system och användare börjar med att antalet nya brev, namnet på avsändarna, ämnet på breven samt ankomsttiden för breven presenteras. Generellt gäller att användaren hoppar mellan breven genom kommandona *nästa*, *föregående*, *första* och *sista*, får ett brev uppläst genom kommandot *läs* samt kan radera ett brev genom att säga *radera*. Hjälp fås genom kommandot *hjälp*. Kommandot *avbryt* ger användaren möjlighet att avsluta tjänsten, söka efter en specifik avsändare genom att säga användarnamnet eller återgå till presentationen av de nya breven. Repetition av informationen fås genom *repetera* och systemet avslutas genom *avsluta*. Om systemet inte uppfattade eller förstod användarens kommando spelas en ursäkt upp och systemet antingen loopar tillbaka och läser upp den aktuella informationen igen eller den aktuella hjälpinformationen. Under uppläsning av ett brev kan användaren

pausa uppläsningen, uppläsningen återupptas då användaren uttryckligen ber om det. Ett fullständigt flödesschema över dialogen och de tillgängliga kommandona finns i bilaga A.

3.1.3 Utveckling av grammatik

Grammatiken för röststyrning av applikationen är relativt enkel. Det enda som ska kännas igen, förutom de kommandon som behövs för att styra funktionerna i systemet och ingången till applikationen från huvudmenyn, är avsändarnamn. Eftersom det inte är möjligt att definiera dessa i förväg används en dynamisk grammatik.

Grammatiken för applikationen består av en toppnivågrammatik (.NAVIGATE_EMAIL) som definierar ett antal subgrammatiker som motsvaras av de funktioner som skall finnas i systemet. Subgrammatikerna ser ut som i exemplet på sida 4.

```
.NAVIGATE_EMAIL
( (?HESITATION)
[
    EMAIL_MAIN_MENU          {<email "main_menu">}
    hjälp                    {<email "help">}
    EMAIL_REPEAT              {<email "repeat">}
    EMAIL_REMOVE              {<email "remove">}
    EMAIL_PAUS                {<email "paus">}
    EMAIL_CANCEL              {<email "cancel">}
    EMAIL_PAUS                {<email "paus">}
    EMAIL_NEXT                {<email "next">}
    EMAIL_PREV                {<email "prev">}
    EMAIL_FIRST               {<email "first">}
    EMAIL_LAST                {<email "last">}
    EMAIL_READ                {<email "read">}
    EMAIL_CONTINUE            {<email "continue">}
    EMAIL_SEARCH              {<email "search">}
    EMAIL_POS_CONFIRMATION    {<email "pos_confirmation">}
    EMAIL_NEG_CONFIRMATION    {<email "neg_confirmation">}
]
)
```

Ingången till själva applikationen och den dynamiska grammatiken ligger i andra systemgrammatiker. Avsändarnamnen hämtas från e-postmeddelandena och läggs i en array innan de omvandlas till en dynamisk grammatik i det tillstånd där de först behövs. Detta görs genom två funktioner där den första loopar igenom arrayen och lägger till olika tecken, tabbar och nyradstecken för att skapa en GSL-grammatik med korrekt syntax, den andra initierar sedan grammatiken. Utvecklingen av den dynamiska grammatiken beskrivs utförligt i Olsson (2004).

3.1.4 Talsyntes

För uppläsning av information rörande e-posten och själva e-postmeddelandena, som man i förväg ej kan spela in som promptar, används TTS (Text-To-Speech), det vill säga talsyntes. Mjukvaran som används i denna prototyp är BrighSpeech från Babel Technologies S.A. Babel Technologies är en global aktör

inom talteknologiområdet och tillhandahåller både programvaror för TTS och ASR (Automatic Speech Recognition). Produkten BrightSpeech är konkateneringsbaserad med så kallad 'Unit Selection' av talenheter. Rösten som används är kvinnlig och kallas för 'Emma'.

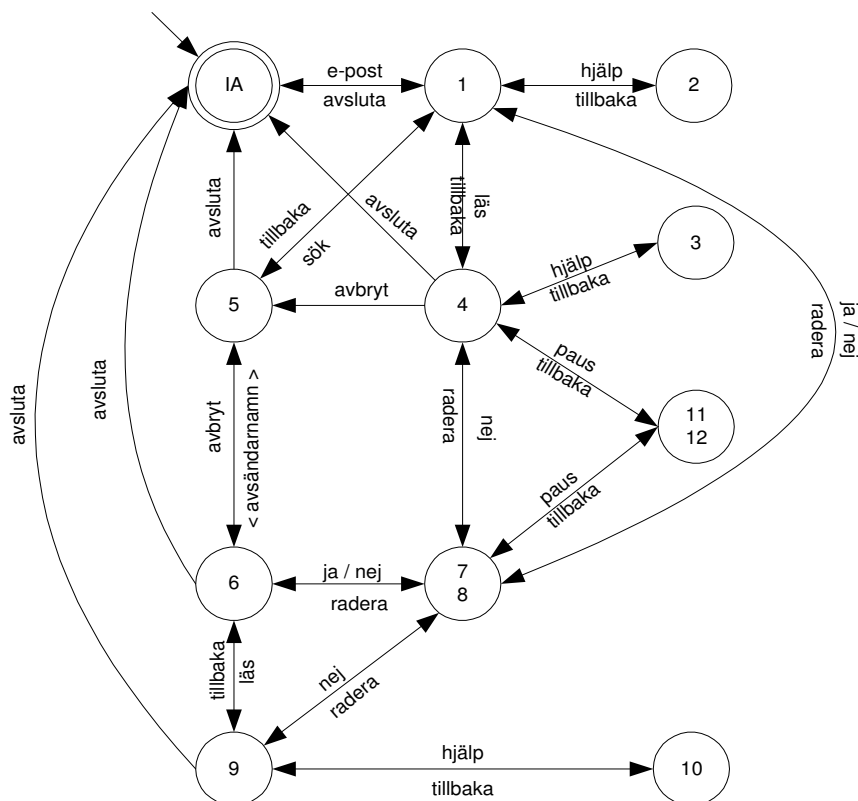
I BabTTS finns möjlighet att använda sig av Microsoft SAPI 4-taggar för att i realtid kunna påverka syntesen, pauser av olika längd kan läggas in och hastigheten på vissa sekvenser kan exempelvis modifieras. Taggarna som består av ett antal tecken omslutna av 'backslash'-tecken läggs i texten som skall syntetiseras. Exemplet:

Det här \Pau=2000\ är en paus

resulterar i att meningen kommer att syntetiseras med en paus på två sekunder mellan här och är (BabelTechnologies 2003).

3.1.5 Implementering av dialogtillstånd

Det röststyrda gränssnittet kan jämföras med en finit automat där de olika funktionerna i applikationen motsvaras av ett tillstånd. Starttillståndet för den här applikationen kommer man till via Icepeak Attendants huvudmeny som också utgör sluttillståndet. Övergångarna mellan de olika tillstånden styrs med hjälp av användarens kommandon som beskrivs i stycke 3.1.2.



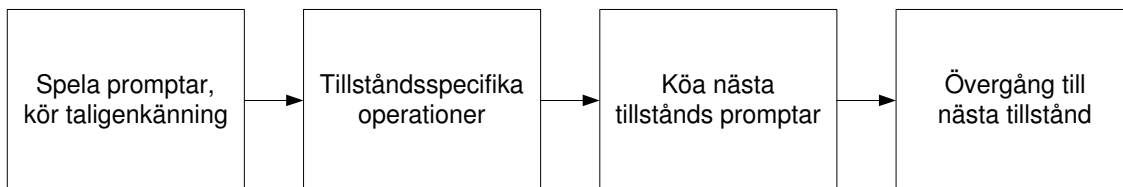
Figur 3.1: Dialogtillstånd

1. emailStartState - användaren får uppläst hur många nya brev som finns i inkorgen, från vilka avsändare med uppgift om ämne, datum etc.
2. emailHeaderFunctionsState - hjälpfunktion för tillstånd 1.
3. emailMailFunctionsState - hjälptillstånd för tillstånd 4.
4. emailReadState - hela brevet läses upp.
5. emailPreFinishState - användaren har möjlighet att söka efter en specifik avsändare genom att säga avsändarnamnet.
6. emailSearchMailState - de funna breven från tillstånd 5 presenteras.
7. emailDeleteState - brev raderas efter användarens bekräftelse.
8. emailFoundDeleteState - funna brev raderas efter användarens bekräftelse.
9. emailReadFoundMailState - de funna breven från tillstånd 6 läses upp.
10. emailFoundFunctionsState - hjälptillstånd för tillstånd 9.
11. emailPausState - paustillstånd för tillstånd 4.
12. emailFoundPausState - paustillstånd för tillstånd 9.

Förutom dessa tillstånd finns också en så kallad pre- och en så kallad postfunktion där prefunktionen initierar variabler och liknande. Postfunktionen tar hand om så kallade STN-fall (Silence To long Nothing understood), det vill säga de fall där systemet inte kunnat uppfatta något kommando under ett visst antal sekunder.

Kodning av dialogtillstånden

Kodningen är gjord i C++ med hjälp av de fördefinierade funktioner, så kallade makron, som finns i systemet. Vart och ett av tillstånden ovan motsvaras av en funktion i koden. Varje funktion är uppbyggd på liknande sätt, först bestäms vilken grammatik som skall användas, i detta fall `.NAVIGATE_EMAIL`, sedan spelas promptar och TTS upp samt igenkänning görs. Eventuella funktionsspecifika operationer görs innan nästa tillstånds promptar och TTS köas och systemet slutligen hoppar till nästa tillstånd.



Figur 3.2: Tillståndsstruktur. Översatt bild från Olsson (2004), med tillåtelse.

De promptar som ska spelas upp förbereds, det vill säga läggs i en så kallad kö, i det föregående tillståndet och spelas sedan upp i det aktuella tillståndet samtidigt som igenkänning av röstkommandon görs (`recognize`). Tekniken, 'barge-in', innebär att ett brev eller en prompt inte måste höras till slut innan användaren kan fortsätta. Eventuella användarkommandon som ges sparas i en variabel `recResult`.

Systemet gör en tolkning av yttrandet och hypotesen återfinns i en variabel `pRecResult`. `pRecResult`-variabeln jämförs med de fördefinierade kommandona i grammatiken, det vill säga i detta exempel de fraser i grammatiken som refereras till av 'sloten' `repeat` (se kodexemplet på sida 4) och `help`. Om skillnaden mellan hypotesen och kommandot är noll anses hypotesen vara korrekt.

```
grammar = ".NAVIGATE_EMAIL" + m_grammar_suffix;
setGrammar( grammar );
recError = recognize( ivrChannel, &recResult, stateMachine, state );

[ ... ]

if( strcmp( pRecResult, "repeat" ) == 0){
    TRY_QUEUE_PROMPT( ivrChannel, "ljudfil_namn" );
    [ ... ]
    TRY_SET_NEXT_STATE( state );
    return;
}
else if( strcmp( pRecResult, "help" ) == 0){
    ERR_REPORT_AND_IGNORE( ivrPlaybackQueueTTS( ivrChannel,
                                                "en sträng att göra TTS på" ), [ ... ] );
    TRY_QUEUE_SILENCE( ivrChannel, SHORT_SILENCE );
    [ ... ]
    TRY_SET_NEXT_STATE( hjälpState );
    return;
}
[ ... ]
```

Beroende på vilken 'slot' som fyllts så köas olika promptar (`TRY_QUEUE_PROMPT`), strängar som ska syntetiseras (`ivrPlaybackQueueTTS`) och pauser (`TRY_QUEUE_SILENCE`). Slutligen sker en övergång till ett annat eller samma tillstånd (`TRY_SET_NEXT_STATE`). För alla ovanstående åtgärder utom för syntetiseringen används fördefinierade makron som skrivs med versaler.

3.2 Preprocessor - en bearbetning av den inkommande e-posten

I och med att 'rå', obehandlad, e-post innehåller mycket teknisk information som inte är intressant för en vanlig användare och som aldrig presenteras i grafiska e-postprogram måste denna även plockas bort i en röstapplikation. Detta görs med hjälp av en preprocessor.

Bearbetningen går ut på att sortera bort sådan information som inte är intressant för användaren att lyssna på och som därmed inte skall skickas vidare till talsyntesen. I detta fall är det viktigt att komma ihåg att information som kanske är naturlig att presentera i ett grafiskt gränssnitt inte nödvändigtvis lämpar sig för ett IVR-system (se stycke 2.2.2). Information om hur tung e-postfilen är presenteras i ett grafiskt program, eftersom användaren enkelt kan bortse från den informationen om den inte anses vara intressant. I ett IVR-system så är det däremot inte möjligt för användaren att bortse från information på ett liknande sätt och därför presenteras sparsamt med information.

Svårigheten i att skapa en preprocessor för e-post ligger i att e-postmediumet är föränderligt, typer och format förändras fort och standarderna är få. Flera av delarna är dessutom språkspecifika och delvis

möjliga att påverka för användaren. Exempelvis kan man i vidarebefordrade e-postmeddelanden se något av följande rader innan själva meddelandet:

```
----- Original Message -----  
----Ursprungligt meddelande-----
```

vilket ibland kan följas av antingen header-fält eller något av följande rader:

```
Maria Landberg wrote:  
NAMN@ADRESS.SE wrote:  
At 01:52 PM 5/22/01, you wrote:  
At 01:52 PM 5/22/01, namn@adress wrote:  
At 01:52 PM 5/22/01, Maria Landberg wrote:
```

Problemet ligger i att avsändaren av brevet kan välja att ta bort hela eller delar av rader, vilket försvårar identifiering av det vidarebefordrade meddelandet.

3.2.1 Den ursprungliga preprocessorn

Den ursprungliga preprocessorn innehåller både språkberoende och språkoberoende delar. De språkberoende delarna behandlar till exempel header-fält, bilagor, indenteringar, tabeller etc. De språkberoende delarna hanterar exempelvis tid, datum och olika markeringar för ursprungliga och vidarebefordrade brev etc. Dessa delar är kodade i flex som är ett verktyg för att generera program, 'scanners', som känner igen lexikala mönster i text. Filerna som flex tar som indata beskriver de 'scanners' som skall genereras och består huvudsakligen av reguljära uttryck och C-kod. Utdatan består av C-filer som vid kompilering skapar exekverbara filer. Det är dessa filer som vid körning 'scannar' sin indata efter förekomster som matchar de reguljära uttrycken som definierades i första steget. Om matchning sker så exekveras den motsvarande C-koden (Paxson 1995). Eftersom en del av ett multipart-brev i sin tur kan bestå av ett e-postmeddelande är det nödvändigt att använda rekursiva metoder för att reda ut strukturen hos brevet.

Det första som sker i preprocessorn är att brevet läses in från fil och görs om till en string-vektor. Header-fälten läses och om 'content-type'-headern är av typen 'multipart' delas brevet upp i sina beståndsdelar, det så kallade förordet ignoreras. Header-fälten i varje del läses och olika objekt skapas beroende på typen. Varje del behandlas sedan med avseende på sin typ. Om delen består av en bilaga, det vill säga har typen 'jpeg', 'ms-word' med flera, sparas innehållet. Om typen är 'message/rfc822' innehåller delen ett komplett e-postmeddelande och rekursion sker. Om delen är av typen 'text/plain' fortsätter den språkliga bearbetningen av innehållet för att identifiera stycken, header-information, signaturer med mera. Flex-funktionerna anropas från processorn för att identifiera språkspecik information. Informationen som hittas sparas i vissa fall i diverse objekt eller vektorer men behandlas inte vidare utan skrivs endast ut i terminalen.

3.2.2 Förändringar av den ursprungliga preprocessorn

Preprocessorn är ursprungligen utformad för brev av typen 'text/plain' och 'multipart'. I dag sänds även en stor del av e-postmeddelandena i HTML-format och har alltså typen 'text/html'. För att även dessa brev skall kunna läsas upp har en funktion som rensar bort HTML-taggar lagts till i preprocessorn. Som

visades i kapitel 2.3.2 börjar HTML-brevets 'body' med en `<!DOCTYPE HTML PUBLIC -//W3C//DTD W3 HTML//EN`-tagg eller liknande (se exemplet på sida 7). Genom att stega igenom bodyn och ta bort de strängar som ligger emellan '`<`' och '`>`' från och med denna tagg elimineras alla HTML-taggar. Naturligtvis är detta inte en helt säker metod, kodningsinformation som inte ligger i taggar tas ej bort och eventuell text i e-postmeddelandet som markerats inom '`<`' och '`>`' försvinner.

För att kunna bygga ihop preprocessorn med det röststyrda gränssnittet sparas det som senare skall läsas upp från e-postmeddelandet som objekt av en skapad e-postklass i en pekararray. Informationen läggs in och hämtas genom olika 'set-' och 'get'-funktioner. Ett stort problem har varit att den ursprungliga preprocessorn var skriven i C++ med standarder och funktioner från programbiblioteket 'Standard Template Library' (STL) medan röstgränssnittet är utvecklat i Microsoft Visual C++ som använder Microsofts utökade bibliotek 'Microsoft Foundation Classes' (MFC). För att få de två delarna att passa ihop har de funktioner som exporteras från preprocessorn, det vill säga som utgör gränssnittet mot röstgränssnittet fått skrivas om ett antal gånger. Funktionerna kan inte innehålla några objekt som inte stöds av MFC vilket exempelvis har inneburit att raden `'using namespace std'`¹ tagits bort och därmed att standardbiblioteket 'string' inte kan användas. Alla variabler av klassen 'string' har därför skrivits om till fält av typen 'char'.

Preprocessorn har sedan gjorts om till ett DLL-bibliotek (Dynamic Link Library) som inkluderas i röstgränssnittet. I biblioteket specificeras vilka funktioner i vilka klasser som ska exporteras, det vill säga kunna anropas utifrån ett annat projekt, i detta fall av röstgränssnittet. Ursprungligen skapades ett bibliotek som inkluderades statiskt, det vill säga genom att bibliotekets sökväg lades till i utvecklingsmiljöns inställningar. Detta skapade ett flertal mycket diffusa problem med oklart ursprung. Biblioteket inkluderas nu istället dynamiskt i det ögonblick som funktionerna behövs genom ett antal funktioner i koden.

¹Alla deklarerationer som finns i standardbiblioteken görs i en så kallad namnrymd som heter 'std'. Raden `'using namespace std'` gör standarddeklarerationerna direkt tillgängliga i programmet (Skansholm 2000).

4 Utvärdering

4.1 Utvärderingsmetod

En mindre användarutvärdering, en så kallad användbarhetstestning har gjorts för att få reda på potentiella användares synpunkter på applikationen. Syftet med utvärderingen är att hitta de mest uppenbara problemen i applikationen och i den mån det är möjligt åtgärda dessa.

Användbarhetstestning är ett mer eller mindre formellt experiment där en person försöker använda systemet för att genomföra ett antal testuppgifter. Interaktionen mellan användare och system studeras, liksom var eventuella fel uppstår och hur många samt var användarna kan behöva extra hjälp (Löwgren & Stolterman 1998). Att metoden har valts beror på att det är en relativt enkel och billig metod för att få reda på viktiga synpunkter. Enligt Nielsen (2000) finner man med endast en testanvändare nästan en tredjedel av de problem som kan finnas i ett system. Med två eller tre testanvändare så hittas en del nya problem men de flesta utgörs ändå av överlappningar, det vill säga problem som redan har hittats av en tidigare testanvändare. Med tre testanvändare hittas uppskattningsvis 65 procent av problemen och med fem testanvändare cirka 85 procent. 100 procent av problemen bör hittas med 15 testpersoner men Nielsen menar ändå att fler än fem testanvändare är onödigt eftersom varje ny användare tillför så lite som inte redan har påpekats av någon tidigare. Det ultimata vore enligt Nielsen att ha fem testpersoner som testat systemet i tre omgångar. De problem som uppdagades i en testning kan då åtgärdas innan nästa för att säkerställa att de verkligen åtgärdats och att de inte leder till några nya oförutsedda problem.

I en testning av detta slag är statistiska resultat mindre intressanta eftersom testet är för litet. Om några användare har problem kan det därför vara klokt att göra ändringar (Löwgren & Stolterman 1998). Man får inte heller i allt för stor utsträckning endast fokusera på användargränssnittet och huruvida användaren lyckas genomföra testuppgifterna, eftersom detta kan leda till att man bortser från användarens totala intryck av systemet. Detaljer kan, oavsett hur små de kan te sig för en utvecklare, uppfattas som mycket störande för en användare och därmed göra att systemet inte kommer att användas (Preece et al 1994). Andra faktorer att ta hänsyn till är att tester, exempelvis denna, i de flesta fall sker i ett 'socialt vakuum', det vill säga inte i den naturliga miljö där systemet är tänkt att användas. Därmed kan det vara svårt att avgöra hur lämpligt systemet är inom det tänkta användningsområdet. Användbarhetstester är dessutom ofta korta och visar därmed endast på systemets styrkor och svagheter vid användarens första kontakt med systemet (Löwgren & Stolterman 1998).

4.2 Tillvägagångssätt

Tre personer har deltagit i användarutvärderingen:

- En man i 50-55 års åldern med datorvana och liten vana av att använda röststyrda tjänster.

- En man i 25-30 års åldern med datorvana och viss vana av att använda röststyrda tjänster som till exempel tidtabellsupplysning.
- En kvinna i 20-25 års åldern med datorvana och liten vana av att använda röststyrda tjänster.

Testpersonerna har fått pröva tjänsten med hjälp av telefon och har sedan besvarat frågor om hur de upplevde tjänsten och systemupplägget. Frågorna redovisas nedan. Ett exempel på en dialog mellan system och användare finns i bilaga B. I och med att systemet idag inte är ihopkopplat med något e-postprogram har testningen skett genom att ett antal specificerade filer läses in i uppringningsögonblicket.

- Hur upplevde du uppläsningstjänsten? Var det lätt eller svårt att använda systemet?
- Gav systemet för lite eller för mycket information? Borde något läggas till eller tas bort?
- Vad var mindre bra med systemet och hur skulle det kunna förbättras?
- Skulle du vilja lägga till eller ta bort någon funktion i systemet?
- Skulle du kunna tänka dig att använda tjänsten? När skulle tjänsten i så fall användas?

4.3 Användarnas synpunkter

Systemet upplevdes som enkelt att använda av alla användarna, men den syntetiska rösten upplevdes som ganska ansträngande att lyssna på. En av användarna ansåg att man vände sig relativt snabbt men två upplevde det som irriterande att lyssna på längre brev. Ytterligare ett problem var att systemet i ett flertal fall inte uppfattade användarnas kommandon, vilket upplevs som frustrerande när det sker upprepade gånger. Exempelvis tolkade systemet en användares kommando 'läs' som 'nej'. Barge-in funktionen upplevdes däremot som positiv av alla användarna i och med att användaren då själv kan styra hur mycket information han eller hon vill höra.

Vad gäller hjälpfunktionen så gick åsikterna bland användarna isär. En användare ansåg att hjälpen innehöll ganska mycket information, men i och med att det är lätt att lyssna om på kommandona eller avbryta uppläsningen så upplevdes inte det som ett problem. Dessutom tror inte användaren att hjälpfunktionen kommer att användas speciellt mycket. Så fort som användaren har satt sig in i systemet så borde inte funktionen behövas. En annan testanvändare störde sig på att man utan att ha begärt det kan hamna i hjälpfunktionen (genom STN-fall eller oriktiga kommandon). Det är irriterande i de fall då användare säger ett giltigt kommando men blir missförstådd av systemet. Hjälpfunktionen borde användaren endast komma till om man uttryckligen begärt det. Vidare ansåg användaren att prompten som inleder hjälpfunktionen skulle kunna förenklas. Användaren ansåg att det inte är nödvändigt för systemet att informera om att kommandona inte kan användas innan hjälpfunktionen avslutas. Just detta tyckte en tredje användare var bra information som kan undanröja eventuella missförstånd som kan uppkomma för en nybörjaranvändare.

Pausfunktionen bör finnas tillgänglig överallt i systemet, inte bara i uppläsningstillstånden. När pausfunktionen avbryts och uppläsningen återupptas vore det också önskvärt att uppläsningen skulle återupptas exakt där pausningen skedde och inte, som idag, från början av det aktuella brevet.

Två av användarna upplevde användningen av kommandona 'avbryt' och 'tillbaka' som något oklar. Idag leder kommandot 'avbryt' alltid till tillståndet 'emailPreFinishState' där användaren har möjlighet att avsluta tjänsten, söka efter en avsändare eller återgå till huvudmenyn medan kommandot 'tillbaka' leder till

det föregående tillståndet. Den tredje testanvändaren upplevde inte några problem med dessa kommandon.

Vad gäller nya funktioner i systemet tyckte två av användarna att många funktioner i ett system inte alltid är det bästa för alla användare. En osäker användare kan bli förvirrad och störas av för många olika kommandon. Idag är systemet lätt att ta till sig just i och med att det har få funktioner. Det vore dock önskvärt att även kunna få bilagor upplästa eller i vart fall filnamnen på eventuella bilagor. Om användaren inte är intresserad kan han eller hon lätt avbryta uppläsningen genom att hoppa till nästa brev. En av användarna föreslog också att man skulle kunna söka på endast för- eller efternamn, inte bara på det fullständiga namnet på en avsändare, eftersom en användare skulle kunna ha glömt bort efternamnet på en avsändare som han eller hon inte känner så väl. Ytterligare en intressant utveckling skulle naturligtvis vara en svarsfunktion där användaren direkt i telefonen skulle kunna diktera ett svar.

Vidare frågade en av användarna hur kopplingen mellan e-postprogram och uppläsningstjänst bäst skulle kunna utformas; om en användare har allt för många brev i sin inkorg blir uppläsningstjänsten ohanterlig och därmed mindre använd. Speciellt sökfunktionen måste i så fall förfinas eftersom det tar lång tid att lyssna igenom många brev innan de eftersökta hittas.

Vad gäller tjänstens potentiella användning så trodde alla användarna att det visst skulle finnas en marknad. Rörliga användare utan tillgång till dator borde vara intresserade av liknande tjänster.

5 Resultat

Uppsatsen beskriver utvecklingen av en röststyrd uppläsningstjänst för e-post för Icepeak AB:s röststyrda telefonist Icepeak Attendant. Tjänsten börjar med att presentera hur många olästa brev användaren har, vilka avsändarna är samt ärendet på breven. Användaren får sedan själv bestämma vilka brev som ska läsas upp och kan under uppläsningen när som helst avbryta eller pausa uppläsningen, gå vidare till nästa eller tillbaka till föregående brev, radera ett brev eller få information repeterad. Användaren kan också söka efter en specifik avsändares e-post genom att säga namnet på avsändaren.

Applikationen består av två större delar, ett röststyrt system och en preprocessor för den inkommande e-posten. Arbetet med röstgränssnittet har huvudsakligen bestått av följande delar:

- Funktionalitet
- Dialog
- Promptar
- Grammatik
- Implementation

Efter att ha bestämt de funktioner som skall ingå i systemet utformades en möjlig dialog mellan användare och system som promptar och grammatik sedan skapas utifrån. För att slippa skriva om grammatiken i ett senare skede planerades dialogen noggrant. Teknologin som används för röstigenkänningen kommer från Nuance Communications Inc. och går i korthet ut på att det som användaren säger jämförs med de kommandon som definieras i grammatiken.

Systemets funktioner implementeras i C++ och motsvaras av ett antal tillstånd som man med hjälp av röstkommandona hoppar emellan. Tillstånden har alla samma grundstruktur; promptar spelas upp, tillståndsspecifika operationer görs, nästa tillstånd promptar köas innan transitionen till nästa tillstånd slutligen görs.

Arbetet med preprocessorn har bestått i att modifiera en redan befintlig processor. Den ursprungliga preprocessorn tog inte vara på den funna informationen och behandlade inte e-post av HTML-typ. Preprocessorn tar nu vara på den information i e-postfilen som skall läsas upp, sparar den som ett objekt av en e-postklass och exporterar de funktioner som röstgränssnittet behöver för att kunna läsa upp e-postmeddelandet. Preprocessorn kompileras som ett DLL-bibliotek och instansieras dynamiskt i röstgränssnittet. En enkel funktion för HTML-rensning har också lagts till.

5.1 Förändringar av systemet utifrån utvärderingsresultatet

I den mån det har varit möjligt har de problem som identifierats i utvärderingen (se kapitel 4.3) tagits fasta på och ändrats i systemet.

En av användarna upplevde det som irriterande att hamna i hjälpfunktionen utan att ha begärt det, de andra ansåg att funktionen inte kommer användas då en användare lärt känna systemet. Systemet har därför ändrats så att användaren endast kommer till hjälpfunktionen efter att uttryckligen ha begärt det. Pausfunktioner har lagts till i fler av tillstånden. Metoder för att hålla reda på var i texten syntesen befinner sig vid pausningen är ett större projekt men skulle förmodligen kunna lösas om tid fanns. Sökning efter en speciell avsändares e-post kan nu även ske med såväl för- som efternamn, förutom med det fullständiga namnet. Vad gäller uppläsning av bilagor så skulle bilagenamnet kunnat läsas upp efter en relativt enkelt komplettering. Om systemet skall kompletteras med en möjlighet att läsa upp bilagorna i sin helhet krävs en större arbetsinsats, vilket också skulle kräva mycket av syntesen.

Exemplet med de två kommandona 'avbryt' och 'tillbaka' visar på problemet med många kommandon och valmöjligheter, det är svårt att förutse hur användare tolkar kommandon och deras användning. Det är svårt att komma till rätta med detta eftersom en användare inte upplevde några problem med dessa kommandon. I och med att det inte är klart hur dessa kommandon ska användas för att upplevas som självklara för alla användare så har den ursprungliga versionen behållits.

Redan i en relativt lugn kontorsmiljö upplever alla testanvändare ibland problem med att bli förstådda av systemet, detta väcker naturligtvis frågan hur väl tjänsten skulle fungera i en annan typ av miljö som en bil eller vid en bullrig arbetsplats. För att försöka förbättra taligenkänningen av de ord som systemet hade svårt att uppfatta har en fonetisk transkription lagts till. Genom denna åtgärd är det möjligt att styra exakt vilket/vilka uttal som systemet skall förknippa med ett kommando. Hur väl detta fungerar i en annan miljö är svårt att förutsäga, det är en fråga som bör tas i beaktning vid en eventuell vidareutveckling av applikationen. Användarnas kommentarer kring kvaliteten på syntesen bör också tas i beaktning. Om applikationen utvecklas vidare för att kunna hantera en användares hela inkorg och kanske även bilagor vore det önskvärt om kvaliteten på syntesen kunde förbättras.

5.2 Diskussion

Även om dialogen har planerats omsorgsfullt och under utvecklingen testats med hjälp av Wizard-of-Oz metoden har det varit svårt att förutse hur kommandon kommer att tolkas av en användare och vilka som är naturliga för användare att nyttja. Man kan i viss mån styra en användare genom att utforma prompterna på ett otvetydigt sätt och därmed ge tillräcklig ledning för att systemet skall kunna användas. Det innebär dock inte nödvändigtvis att alla användare intuitivt känner vilket kommando som skall användas.

E-post som medium kan vara ganska komplicerat att arbeta med, formaten är många och förändras ofta. Beroende på vilken portal som ett HTML-brev skickas från så kan exempelvis taggar se olika ut, och i vissa fall inte vara uppbyggda som klassiska taggar. Att bygga en HTML-rensning för alla dessa är tidskrävande och kräver förmodligen kontinuerlig uppdatering. Ett annat problem med HTML-brev är att de kan innehålla så mycket annat än bara text som måste specialbehandlas i en applikation som denna, bilder och tabeller till exempel. Vad som gör arbetet ännu svårare är att avsändaren har möjlighet att själv påverka och modifiera brevet. Hur ska man exempelvis kunna skilja ett besvarat meddelande från det aktuella ifall ingen indentering eller 'original message'-rad finns kvar?

De problem som uppkommit under implementationen har i huvudsak berott på att röstgränssnittet är

kodat i Microsoft Visual C++ som använder sig av MFC-biblioteket, medan den redan befintliga preprocessor som arbetet utgick ifrån använde sig av STL-biblioteket. Ifall att dessa två delar skulle ha varit mer kompatibla hade mycket tid kunnat sparas för att istället läggas på förbättringar av andra delar av systemet.

Utvärderingen visar dock att systemets funktioner fungerar väl, systemet är lätt att använda och användarna har inga problem med att få de önskade breven upplästa. Systemet fyller därmed sin funktion. De förslag på mindre förändringar som framkommit har i den mån tiden tillåtit ändrats enligt testanvändarnas synpunkter.

Den allvarligaste kritiken som framkom under utvärderingen och det som upplevdes som mest störande av användarna var att systemet ibland hade svårt att förstå användarnas kommandon. Kvaliteten på syntesen fick också kommentarer. Detta är faktorer som kan påverka användandet av tjänsten, speciellt som tjänsten främst är tänkt att användas i miljöer där dator och/eller uppkoppling inte är tillgänglig och som kan tänkas vara bullrigare än en kontorsmiljö.

5.3 Utvecklingsmöjligheter

Applikationen är idag inte ihopkopplad med något e-postprogram. Systemet läser in ett antal e-postmeddelanden från fil för att användartestning ska kunna genomföras. Innan man i ett senare skede kopplar till ett e-postprogram bör man tänka igenom hur väl systemet skulle fungera för användare med mycket stor inkorg. Ska alla brev kunna läsas upp, endast nya brev eller ska användaren kanske kunna välja vilka brev som ska vara tillgängliga för avlyssning via telefonen.

Utvecklingsmöjligheterna för denna tjänst är många, såväl röstgränssnitt, preprocessor som talsyntes skulle kunna förbättras och vidareutvecklas. En förbättrad HTML-rensning som klarar av att ta hand om bilder, tabeller och annan 'icke-text' på ett bättre sätt är exempelvis önskvärd innan systemet kan tas i bruk. En språkidentifieringsfunktion vore också önskvärd eftersom talsyntesen som används endast är gjord för svenska språket. Brev skrivna på engelska borde läsas upp med engelsk syntes och om inte det är möjligt sållas bort.

Vidare kan man tänka sig att även funktioner som 'skicka', 'svara', 'svara alla' och 'vidarebefordra' kan ingå, något som också testanvändarna önskade sig. Användaren skulle kunna diktera brev över telefonen som skickas till mottagaren som ett vanligt e-postmeddelande. En annan möjlig utveckling är att möjliggöra uppläsning av vissa bilagor, beroende på format. Detta kräver dock mycket av talsyntesen som är en viktig del av tjänsten. Om talsyntesen är bristfällig blir tjänsten automatiskt mindre användarvänlig. Att lägga till Microsoft SAPI taggar för att styra talsyntesens pauser, intonation, gruppering etc. är ytterligare en utvecklingsmöjlighet.

Idag görs inget med informationen om eventuella kopior (Cc) etc. Detta skulle kanske i vissa fall kunna vara intressant information som skulle kunna presenteras på befallning, likaså telefonnummer och webbadresser som kan finnas insprängda i ett brev. Andra utvecklingsmöjligheter vore att expandera vanliga förkortningar och tagga upp mindre vanliga för specialbehandling i syntesen. Vad som dock bör beaktas inför en framtida utveckling av systemet, men kanske också rent generellt, är att försöka hålla systemet relativt enkelt. Som utvärderingen visade anser testanvändarna att styrkan i den utvecklade applikationen ligger i att på ett enkelt sätt kunna använda tjänsten. Utvärderingen visar också att det är användarvänligheten och interaktionen mellan användare och system som är det viktigaste för användaren, inte att tjänsten har så många funktioner som möjligt.

Litteraturförteckning

Acero, A. et al (2001). *Spoken language processing*. Upper Saddle River: Prentice Hall.

BaBTTS User Guide Version 1.2.(2003). Babel Technologies S.A. Stockholm.

Berners-Lee, T. & Connolly, D. (1995). *RFC 1866*. [Elektronisk].
<http://www.faqs.org/rfcs/rfc1866.html> [Oktober 2004].

Borenstein, N. & Freed, N. (1993). *RFC 1521*. [Elektronisk].
<http://www.faqs.org/rfcs/rfc1521.html> [Oktober 2004].

Ericsson AB:s hemsida: *OneBox Unified Messaging*. [Elektronisk].
<http://www.ericsson.com/enterprise/products/onebox.shtml> [September 2004].

Grammar Developer's Guide (2001). Nuance Communications Inc. Menlo Park.

Löwgren, J. & Stolterman, E. (1998). *Design av informationsteknik*. Lund: Studentlitteratur.

Nielsen, J. (2000). *Why you only need to test with five users*. [Elektronisk].
<http://www.useit.com/alertbox/20000319.html> [September 2004].

Olsson, U. (2004). *Developement of a voice-driven dictation application*. Examensarbete Uppsala Universitet. http://stp.ling.uu.se/~matsd/thesis/arch/2004_olsson2.pdf [September 2004].

Palme, J. (1995). *Electronic Mail*. Norwood: Artec House.

Paxson, V. (1995). *Flex, version 2.5. A fast scanner generator*. [Elektronisk].
http://www.gnu.org/software/flex/manual/html_mono/flex.html#SEC3 [Oktober 2004].

Phoneticom AB:s hemsida: *Phoneticom Tele-Post*. [Elektronisk].
<http://www.phoneticom.com> [Juli 2004].

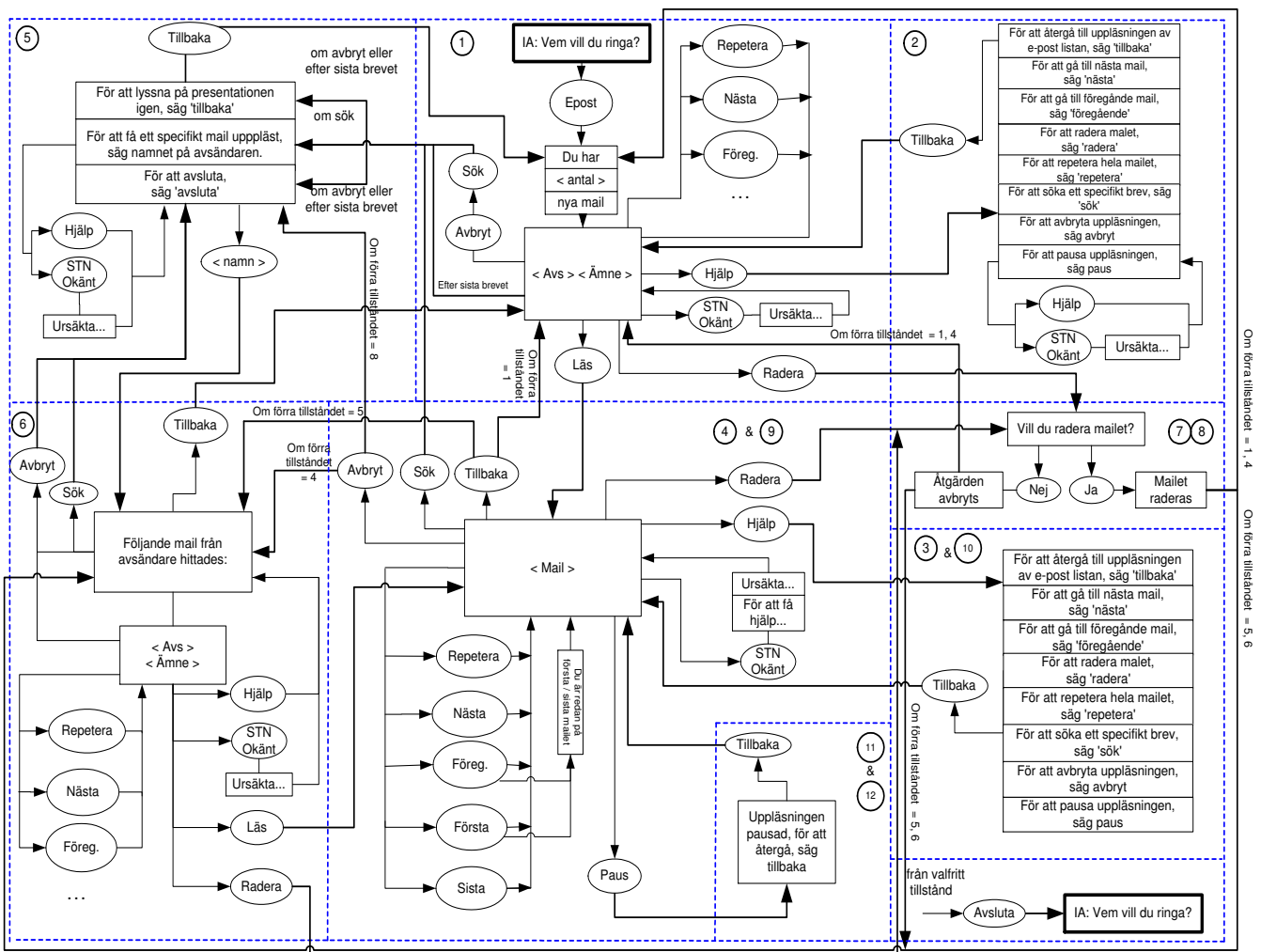
Preece, J. et al (1994). *Human-Computer interaction*. Harlow: Addison Wesley.

Rhoton, J. (2000). *Programmer's Guide to Internet Mail*. Woburn: Digital Press.

Skansholm, J. (2000). *C++ Direkt*. Lund: Studentlitteratur.

Bilagor

A Dialogfödesschema



B Dialogexempel

- **"Vem vill du ringa?"**

- E-post.

- **"Du har 3 nya meddelanden:"**

Stina Student, Exjobb, onsdag den 27 oktober 2004 kl. 12.35.20. En bilaga finns.

"Vill du få brevet uppläst eller gå till nästa?"

- Läs upp.

- *Hej, skickar över [...]*

- Sök.

- **"För att söka efter ett specifikt brev, säg namnet på avsändaren."**

- Johan Jobbare.

- **"Ett brev från avsändaren hittades."**

Johan Jobbare, Måndagsmötet, onsdag den 27 oktober 2004 kl. 14.25.30

"Vill du få brevet uppläst eller gå till nästa?"

- Läs.

- *Har tyvärr fått förhinder, kan ej på måndag. Kan vi ses på tisdag klockan 12.00 istället? Johan*

- Radera.

- **"Vill du radera meddelandet, säg ja eller nej."**

- Ja.

- **"Meddelandet raderas."**

- **"Du har [...]"**

- Avsluta.

- **"Vem vill du ringa?"**

Textförklaring:

Fet stil = promptar

Fet och kursiv stil = TTS

Vanlig stil = användarkommandon