WebFax
Massdistribution av fax via Web-klient

Ahmad Khalil
Kungliga Tekniska Högskolan

Examensarbete i
Teleinformatik

Handledare: Thomas Källander
INEO Konsult AB

Björn Lisper
PARADISlab
Kungliga Tekniska Högskolan

Examinator: Björn Lisper
PARADISlab
Kungliga Tekniska Högskolan

Stockholm den 18 juni 1999

Abstract

In today's society, most of the communication between people is carried out through telephony and Internet. It's amazing that fax still holds a strong position and the amount of fax in telephony communication is increasing. There is also an increasing need for making fax transmissions containing hundreds or thousands of faxes, e.g., when large companies want to send some information to their customers all around the world. Making such big fax transmissions will be very time consuming if it is made in a traditional way by using a standard paper fax machine.

We (at Ineo) have developed a powerful tool for big fax transmissions. The tool is called WebFax and this report shows that WebFax will make it easier, faster and cheaper to send big fax transmissions via a Web client. This report also contains a discussion on several of Microsoft's technologies, e.g., COM, ADO, MSMQ etc.

Innehållsförteckning

1 Introduktion 4
1.1 Marknad 4
1.2 WebFax 4
1.3 Tekniska krav på WebFax 6
1.4 Vad är COM? 6
1.5 Målet med detta examensarbete 6
2 Bakgrund 7
2.1 Företagsbakgrund 7
2.2 Tidigare tjänster på INEO 7
2.3 Andra faxtjänster på marknaden 7
2.3.1 Massfax 7
2.3.2 eFax 8
2.3.3 FaxAway 8
2.3.4 PeopleFax 9
3 Problemställning 10
3.1 Web-klient 10
3.2 Dokumentkonvertering 10
3.3 Meddelandehantering 10
3.4 Rapportering 11
3.5 Återställbarheten i systemet 12
3.6 Datalagring 12
4 Lösning 14
4.1 COM 14
4.1.1 Begrepp 14
4.1.2 Komponentbaserat system 16
4.1.3 COM vs CORBA 18
4.2 Dokumentkonverteraren 18
4.3 Databas 19
4.4 MSMQ och köer 20
4.5 ADO och dataåtkomst 22
4.6 Webben 23
4.6.1 Script 23
5 Resultat 25
6 Slutsatser 29
7 Referenser 30
8 Ordlista 31

1 Introduktion

I dagens samhälle sker en allt större del av kommunikationen mellan människor på elektronisk väg. Telefon, fax, epost och Webb, för att nämna några, står för en stor del av kommunikationen i dagens informationssamhälle. Att fax fortfarande står för en betydande del av teletrafiken kan tyckas förvånande, men faktum är att faxtrafiken inte minskar utan tvärtemot ökar [SS99]. Det finns också ett ökat behov av att göra massutskick (flera hundra/tusen) av fax, t.ex. när stora företag vill skicka information till en stor kundkrets runtom i världen. Att göra ett så stort utskick blir för kostsamt om det görs på det traditionella sättet med manuell pappersfax. WebFax är tänkt att göra det både snabbare, billigare och enklare att göra massutskick av fax via en Web-klient, vart man än befinner sig i världen. Det enda som krävs av användaren som ska använda WebFax är en Web-läsare och surfmöjligheter.

1.1 Marknad

Under de första veckorna av examensarbetet har jag bland annat undersökt vilka önskemål eventuella användare har. Jag, tillsammans med flera andra på INEO, har gjort flera kundbesök med avsikt att undersöka hur stor efterfrågan är samt vilka tjänster användarna är intresserade av. Med dessa kundbesök som underlag kan konstateras att det finns tillräcklig efterfråga för att introducera WebFax på marknaden. Vidare har det framgått att användarna hellre utnyttjar en Web-klient än använder sig av elektronisk post samt att eventuell dokumentkonvertering (till TIFF-format[1]) skall ske automatiskt. Användarna vill dessutom ha rapporter om hur varje, lyckad eller misslyckad, faxtransaktion gick. Rapporterna görs tillgängliga för användaren genom Web-klienten.

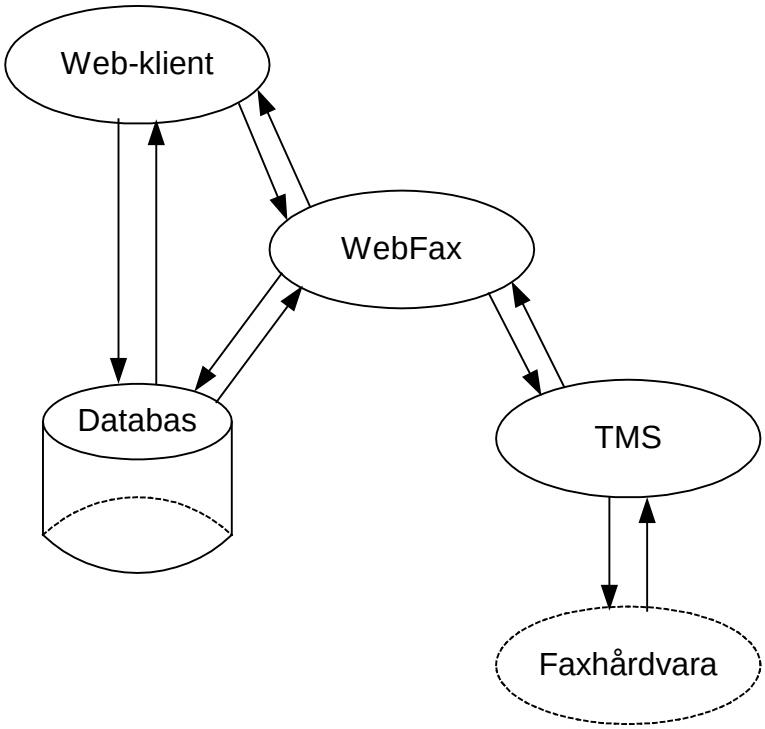
Det finns flera existerande tjänster och produkter ute på marknaden, men ingen av de jag undersökt har kombinationen "Web och massutskick" vilket är WebFax:s kännetecken. Det genomgående inslaget hos de existerande tjänsterna är utnyttjandet av epost istället för en Web-klient.

1.2 WebFax

WebFax, som består av flera applikationer samt **COM** (Component Object Modelling) objekt, ingår som en del i ett större system (Figur 1) för förmedling av stora faxtransaktioner. Systemet består av följande delar:

- Web-klient
- Databas
- WebFax
- TMS[2] (TeleMedia Service)

[1] Tiff (Tagged Image File format, version 3) är det filformat som faxar världen över använder.
[2] TMS är en av INEO utvecklad komponent som abstraherar telefonihårdvara...



Figur 1: Systemet som WebFax ingår i.

Användaren ska kunna utnyttja WebFax genom att med hjälp av användarnamn och lösenord logga in på en av INEO tillhandahållen Web-klient. Användaren ska därefter kunna börja utnyttja WebFax genom att bl.a. välja vilket/vilka dokument som skall skickas till av användaren vald sändlista/sändlistor. Administration av både enskilda mottagare och grupper av mottagare (sändlistor) ska ske via Web-klienten. Användaren får då själv se till att sändlistornas innehåll, som sparas i en databas hos INEO, är korrekt. Användaren ska också ha möjlighet att specificera en omförsöksstrategi i fall en mottagare inte nås vid första försöket. Användaren ska också ha möjlighet att ange med vilken prioritet faxen skall skickas samt vilken nätoperatör som skall användas. Dessa tjänster är endast en del av vad WebFax kommer att kunna erbjuda sina användare.

WebFax tar emot förfrågningar om faxtransaktioner från Web-klienten, utför eventuella dokumentkonverteringar för att sedan skapa enskilda faxtransaktioner för varje mottagare. Dessa transaktioner läggs sedan i en kö för vidarebefordring mot TMS (TeleMedia Service) som utför själva faxutskicket. INEO klarar av stora faxutskick tack vare den stora linjekapacitet som INEO förfogar över. Vidare registrerar systemet kontinuerligt alla utskick i en databas för att möjliggöra för användaren att se utskicksrapporter som görs tillgängliga via Web-klienten.

1.3 Tekniska krav på WebFax

Som det redan har framgått, ingår WebFax i ett system som har många komponenter skrivna av olika utvecklare i olika programmeringsspråk. Detta ställer stora krav på att WebFax ska kunna samexistera i systemet oberoende av vilket språk de andra delarna av systemet är skrivna i. Vidare ställs det krav på att WebFax skall vara lätt att vidareutveckla då nya tjänster skall läggas till. Ett annat krav är att WebFax skall vara driftsäker, övriga komponenter i systemet skall inte påverkas av instabilitet i WebFax och vice versa.

Krav på WebFax:

- Objektorienterat
- Komponentbaserat
- Språkoberoende
- Versionsvänlig (nya versioner på komponenter ska vara bakåtkompatibla).
- Helautomatiserat system

1.4 Vad är COM?

Svaret på frågan, vad är COM, blir olika beroende på vem man frågar. Enkelt utryckt är COM "*en specifikation för att skriva återanvändbar mjukvara som körs i komponentbaserade system",* [COMSPEC]. Följer utvecklare COM-specifikationen (Microsoft) blir det enklare för de att skriva komponenter som kan interageras i ett system med olika komponenter från olika leverantörer. COM är språkoberoende till den grad att programmerare kan utveckla komponenter i många olika programmeringsspråk som finns på marknaden idag, t.ex. C, C++, Visual Basic, Java, Delphi o.s.v. Vidare gör COM det mycket enklare att sköta versionshanteringen då nya versioner av en komponent utvecklas. Gamla komponenter ersätts helt enkelt med nya. Detta medför inga problem gentemot gamla klienter till COM-komponenten, då COM-specifikationen säger att nya funktionaliteter läggs till i en COM-komponent medan gamla funktionaliteter måste bibehållas. Det enda COM kräver av programmeringsspråket som används är att det klarar av pekare.

En mer detaljerad beskrivning av COM kommer längre fram i kapitel 4.1.

1.5 Målet med detta examensarbete

Det är svårt att fastställa vad som skall ingå i mitt examensarbete på grund av svårigheten att veta hur lång tid varje del av projektet tar. Efter ett möte mellan mig, industrihandledaren Thomas Källander samt IT-handledaren Björn Lisper kom vi fram till att examensarbetet skall innehålla minst den första punkten av följande:

- Analys, design, implementering samt driftsättning av WebFax med tillhörande undersökning av huruvida COM-specifikationen kan användas i projektet.
- Utveckling av en Web-klient som ett gränssnitt mot användare samt design och implementering av databas för lagring av information i systemet.
- Undersöka möjligheterna till implementering av ett nytt debiteringssystem med tillhörande databas.

2 Bakgrund

2.1 Företagsbakgrund

INEO Konsult utvecklar system och utför konsulttjänster inom PC-baserad datortelefoni (CTI), meddelandehantering och databaser. Företaget har utvecklat system för samtalsstyrning, hämtfax, massdistribution av fax, gateways för fax, simulering av telefonväxlar och IP-telefoni. Vid sidan av konsultverksamheten har företaget tagit fram ett antal egna produkter inom kompetensområdet, däribland en telefoniserver som används som kärna i ett befintligt telefonkonferenssystem.

2.2 Tidigare tjänster på INEO

WebFax-tjänsten har tidigare kunnat erbjudas genom INEO Konsult AB, men med vissa begränsningar gentemot användaren och med låg effektivitet när det gällde bearbetning av data. Den gamla WebFax behövde omkonstrueras och byggas helt från grunden. INEO hade inte heller någon Web-klient, utan samarbetade med ett annat företag för att kunna erbjuda denna möjlighet. Användare kunde endast logga in på Web-klienten från förbestämda arbetsplatser. Detta erbjöd användarna en större säkerhet eftersom Web-servern då genomförde en kontroll på IP-adressen.

2.3 Andra faxtjänster på marknaden

Det finns en hel del andra konkurrerande tjänster ute på marknaden, men ingen som riktigt erbjuder kombinationen "Web-klient och massutskick". Nedan går jag korfattat genom hur varje tjänst fungerar. Dessa beskrivningar är en sammanfattning av företagens egna genomgångar på respektive hemsidor.

2.3.1 Massfax

Massfax är en konkurrerande tjänst som erbjuds av det svenska företaget TeleTalk. Användaren faxar sitt dokument till TeleTalk och anger samtidigt sin sändlista. Då faxen når TeleTalk startar en vidaresändning via TeleTalks linjer och faxdokumentet når de namngivna mottagare på några minuter. Sändlistan kan användaren uppdatera direkt via Webben eller genom att skicka in en uppdaterad lista på en diskett till TeleTalk.

Det behövs en fax med inbyggd högtalare och som ger kopplingston innan sändning för att utnyttja tjänsten Massfax. Användaren ringer in sin beställning genom sin egen fax. Under beställningen får användaren muntliga instruktioner via högtalaren om hur han/hon ska gå till väga för att skicka in beställningen. Det som användaren behöver vid en beställning är en kod som identifierar användaren samt ett unikt nummer för varje sändlista som användaren har.

Fördelar med Massfax:

- Om ett utskick inte kan tas emot av en mottagande fax kan det automatiskt skrivas ut hos TeleTalk och skickas som brev via den vanliga posten, s.k. Posten eBrev.

- Ingen startavgift och inget månadsabonnemang behövs.

Nackdelar med Massfax:

- Användaren måste ha en fax med inbyggd högtalare som ger kopplingston innan sändning.
- Användaren måste hantera en lista med alla nummer (dessa nummer används för att identifiera sändlistorna) på sina sändlistor.
- Vid beställning får användaren instruktioner via faxens högtalare, vilket känns onaturligt och ökar risken för fel.

2.3.2 eFax

Svenska eFax har en tjänst liknande WebFax, men som använder användarens mail-server istället för en Web-klient. eFax gör det möjligt att både skicka och ta emot fax via epost. Det eFax behöver veta är vilka epost adresser användaren vill kunna ta emot fax på och vilka epost adresser användaren vill kunna skicka till. Vid ett utskick skriver användaren mottagarens epost adress i adressfältet och bifogar dokumentet som skall skickas. eFax stödjer en rad dokumentformat, t.ex. text, Word, Excel och PDF för att nämna några. Själva konverteringen sker hos eFax då dokumentet ska faxas iväg.

En trevlig finess med eFax är att om användaren vid något tillfälle inte har tillgång till sitt epost konto kan han/hon istället använda Webben. Konstigt nog heter denna kostnadsfria tilläggstjänst WebFax, precis som INEOs kommande tjänst. Tyvärr går det inte att se något demo av denna tilläggstjänst.

Fördelar med eFax:

- Användaren kan skicka fax med antingen epost eller via Webben.

Nackdelar med eFax:

- Det går inte att skicka flera fax på en gång, dvs. massfax.
- En start- och månadsavgift måste betalas.

2.3.3 FaxAway

FaxAway är en tjänst som istället för att utnyttja *webben* använder sig av epost för att möjliggöra faxutskick utan att gå via den traditionella faxmaskinen. Istället för att ange en vanlig adress på formen xxxx@yyyyy.zz används faxnummer@faxaway.com. Det går att skicka vanlig text som kan skrivas där text för ett vanligt epost brev skulle ha stått eller bifoga ett dokument. Det bifogade dokumentet kommer då att konverteras till TIFF-format för att kunna faxas iväg. Det går även att skicka fax till flera mottagare samtidigt. Detta blir möjligt genom att användaren utnyttjar distributionslistor som finns lagrade på servrar hos FaxAway. Hur dessa listor administreras framgår inte.

Som det framgår ovan krävs det att användaren har ett epost system för att kunna utnyttja FaxAway. Vidare känns proceduren att skicka fax lite omständig då det elektroniska brevet måste följa en bestämd mall. T.ex. får det inte finnas någon text i det elektroniska brevet om det finns ett bifogat dokument.

Fördelar med FaxAway:

- FaxAway klarar att konvertera från många olika filformat till TIFF.
- Debiterar inga fasta kostnader. Endast transmissionstid kostar.

Nackdelar med FaxAway:

- Användaren måste använda epost för att kunna göra utskick.
- Proceduren att skapa ett faxutskick är omständig.
- Det går inte att skicka flera dokument samtidigt.
- Det går inte att skicka text och dokument samtidigt.

2.3.4 PeopleFax

PeopleFax är ett USA-beläget företag som tillhandahåller en tjänst för faxutskick via *webben*. Denna tjänst har stora likheter med WebFax, men det finns en del skillnader också. Användaren kan utnyttja tjänsten hos PeopleFax genom att via *webben* öppna ett faxkonto med tillhörande identitet och lösenord. För att utnyttja tjänsten behöver användaren bara logga in, sedan är det bara att börja faxa.

Fördelar med PeopleFax:

- Gratis att använda.

Nackdelar med PeopleFax:

- Det går att endast skicka tre sidor vid ett faxutskick.
- Endast ett fax i taget kan skickas.
- Utskick kan endast ske inom USA.
- Tjänsten är reklamfinansierad. Reklamtext kommer att finnas på första sidan i faxutskicket.

3 Problemställning

En av målsättningarna med WebFax är att det ska vara ett flexibelt system. Nya funktionaliteter ska kunna läggas till och gamla funktionaliteter ska kunna modifieras på ett enkelt sätt. Med detta som utgångspunkt bestämde jag att WebFax ska implementeras som ett komponentbaserat system.

3.1 Web-klient

Efter de diskussioner som fördes mellan mig och min handledare på INEO kom vi fram till att en Web-klient är det bästa sättet för användaren att kunna administrera sina data i WebFax. Det enda användaren behöver är surfmöjligheter. Säkerheten behöver endast vara måttlig eftersom WebFax inte är ämnat för faxning av säkerhetsbelagda dokument.

Användaren använder ett användar-ID och ett lösenord för att logga in på Web-klienten. Väl inloggad ska användaren kunna skapa ett faxutskick eller administrera sina sändlistor eller mottagare. För att skapa ett faxutskick ska användaren välja sändlista/sändlistor och/eller mottagare samt bifoga ett dokument. Utskicket tas emot av en server hos INEO för att sedan skickas ut via INEOs stora linjekapacitet. Via Web-klienten bestämmer användaren också vid vilken tidpunkt utskicket ska inträffa. T.ex. kan användaren skapa utskicket klockan 15:00, men bestämma att utskicket ska ske först dagen efter vid någon bestämd tidpunkt. På detta sätt kan användaren spara användaren mycket pengar genom att göra utskick vid tider med låg telefontaxa.

Via Web-klienten ska användaren också ha möjlighet att se hur det går med ett pågående eller planerat faxutskick. En detaljerad rapportsida på Web-klienten ska visa vilken status varje faxutskick har.

3.2 Dokumentkonvertering

Ett av de svårare problemen som WebFax måste lösa är att konvertera filer från olika filformat till TIFF-format. användarens dokument måste konverteras till TIFF-format för att kunna sändas som fax och tas emot som fax. För att öka chanserna för WebFax att lyckas på marknaden bör ett stort antal filformat kunna konverteras till TIFF. Som ett minimum bör konverteraren klara av att konvertera dokument från Microsoft Office.

Tidigare har man på INEO utnyttjat en printerdrivrutin för att klara av konverteringar. Dokumentet som ska konverteras skickas då till printerdrivrutinen som är inställd på att skriva till fil och inte på en skrivare. Denna nya fil kommer då att ha det filformat som printerdrivrutinen är inställd på. Detta sätt att konvertera filer är rätt omständigt och därför kommer jag att undersöka om det finns andra möjligheter att genomföra konverteringen på.

3.3 Meddelandehantering

Eftersom WebFax består av flera olika komponenter måste det finnas ett kommunikationssystem som klarar av att förmedla data mellan komponenterna. Det som

krävs av ett sådant kommunikationssystem är att förmedling av data är tillförlitlig och att den sker någorlunda snabbt.

För nuvarande finns det två olika lösningar som utnyttjas på INEO. Det ena systemet utnyttjar en databas i SQL-Server för mellanlagring av all data som skickas mellan olika mjukvarukomponenter. Varje komponent ställer upprepade frågor med jämna mellanrum för att kontrollera om nytt data har skickats till den. En stor nackdel med detta system är att varje komponent måste använda en stor del av sin tilldelade CPU[3] tid till att ställa frågor mot databasen. Är det dessutom stora mängder data lagrade i databasen kommer systemet att gå mycket långsamt. En liten fördel med detta system är att det är enkelt att hantera.

Det andra systemet[4] som finns implementerat på INEO utnyttjar sig av COM. De olika mjukvarukomponenterna är då skrivna som COM-komponenter och kommunicerar med varandra via sina objekts gränssnitt. Vanligtvis är det klientkomponenten som initierar förfrågningar för att erhålla de tjänster som serverkomponentens objekt har att erbjuda. Ett objekt kan returnera information till klienten som ett svar på ett funktionsanrop men objektet kan inte initiera ett anrop till klienten. Detta kommunikationssätt mellan klienten och objektet kallas envägskommunikation. Genom att programmera om komponenterna till att kunna hantera händelsestyrning enligt COM-specifikationen kan man skapa komponenter som klarar av tvåvägskommunikation, dvs. både klienten och servern (objektet) ska kunna initiera förfrågningar. Objektet kommer då att ha en referens till klienten som den kan anropa klienten genom. Med hjälp av händelser sparar systemet mycket tid som annars skulle användas till att ställa upprepade förfrågningar som i fallet med databaslösningen.

När det gäller meddelandehantering i WebFax har uppdragsgivaren INEO önskemål om att testa en helt ny produkt från Microsoft, nämligen Microsoft Message Queue (MSMQ) [AHDS98]. MSMQ har tidigare aldrig använts på INEO och därför önskas det att den utnyttjas i WebFax. MSMQ är baserad på asynkron meddelandehantering mellan definierade köer. Detta betyder att en klient kan skicka meddelande till en server även om servern inte är tillgänglig för tillfället. Servern i sin tur kan ta emot meddelande från klienten även då klienten har kopplat bort sig. Detta sätt att skicka meddelande mellan applikationer blir möjlig tack vare att MSMQ utnyttjar köer. Applikationerna skickar aldrig meddelande mellan varandra direkt, utan de skickar meddelandena till köer istället. Dessa köer kommer då att fungera som en tillfällig lagringsplats.

3.4 Rapportering

Det ligger i användarens intresse att få veta om varje faxtransaktion har nått fram till sin mottagare eller ej. Användaren skall ha möjlighet att via Web-klienten se rapporter över alla tidigare utskick som skapats. Ur rapporten ska det framgå mottagarens namn, det utskickade dokumentets namn samt vilken tid som faxen skickades iväg. Vidare ska

[3] Central Processing Unit.
[4] En annan examensarbetare är i slutfasen av utvärdering av systemet.

användaren kunna utläsa vid vilket försök som faxutskicket lyckades och hur många sidor som skickades.
I dessa utskicksrapporter ska det för varje enskilt faxutskick framgå:

- Mottagarens namn.
- Lyckat eller misslyckat utskick. Om utskicket lyckades ska det framgå vid vilket försök.
- Transmissionstid.
- Tidpunkt för faxutskicket.
- Antal sända sidor.

Övergripande information som ska finnas på rapportsidan är:

- Namnet på dokumentet som ska skickas. Antal sidor ska framgå.
- Aktuell status på varje enskild mottagare. Av denna statusinformation ska det framgå om utskicket är påbörjat, ej påbörjat eller om det ligger i utskickskön.

Rapportsidan kommer förmodligen att få ändrat utseende allteftersom nya önskemål från tilltänkta användare tillkommer.

3.5 Återställbarheten i systemet

En av de viktigaste aspekterna i hela WebFax är att inget data får gå förlorat. I WebFax skall all kommunikation mellan de olika komponenterna ske genom att utnyttja köer i MSMQ. För att inget data ska gå förlorat måste alla köer klara av att hantera transaktioner. Detta innebär att om en applikation ska skicka ett meddelande från kö A till kö B, kommer meddelandet i kö A inte att tas bort förrän MSMQ vet att kö B har fått meddelandet.

Om datorn där systemet körs går ner mitt under utförandet av en transaktion, skall den vid nästa start kunna starta om alla delar i WebFax och fullfölja transaktionen utan problem. Därför bör alla komponenter i WebFax byggas som tjänster (eng. service). Tjänster [MM99] är applikationer i Windows NT som kommer att starta då datorn startar. Ingen användare behöver vara inloggad för att de ska startas. Tjänsterna i sig har inget med att transaktionen fullföljs, utan det är själva WebFax komponenterna tillsammans med MSMQ köerna som ska ta hand om det.

3.6 Datalagring

Det kommer att finnas en hel del data om användare och deras sändlistor som måste lagras i någon form av databas. Hela databasen borde bli ganska enkel om den delas upp i två logiska delar, en del med statisk information om användaren och användarens sändlistor samt en annan del som innehåller information om varje utskick. Informationen

om utskicken kan sedan utnyttjas för att ge användaren utskicksrapporter samt ligga som grund för hur mycket användaren ska debiteras.

Eftersom hela WebFax kommer att köras i Windows-miljö borde Microsoft SQL Server vara ett klokt val av databasserver (val av databasserver ingick inte i examensarbetet). SQL Server 7.0 är den version som kommer att utnyttjas för datalagring.

4 Lösning

För att kunna klara så många som möjligt av de krav som ställts på WebFax har jag fått utnyttja en hel del av Microsofts produktsortiment. I detta kapitel ger jag en kortfattad beskrivning av varje produkt som använts i WebFax. För ytterligare information hänvisas läsaren till de referenser som anges under respektive dekapitel.

4.1 COM

COM(Component Object Model) [GSTR98][Eddon98] är en arkitektur för komponenter som gör det möjligt för olika leverantörer att skapa mjukvara som kan interoperera på ett tillförlitligt och kontrollerat sätt. Mjukvaruleverantörer behöver inte prata med varandra för att utbyta specifikationer eller koordinera deras utvecklingsarbete. De behöver endast bygga komponenter som följer COM-standarden.

COM är inte ett objektorienterat språk, utan en standard. Språk, struktur och implementeringsdetaljer är upp till applikationsprogrammeraren. Vad COM-standarden specificerar är en objektmodell och vissa programmeringskrav för att möjliggöra för COM-objekt att samspela. Dessa objekt kan befinna sig i samma process, i olika processer på samma maskin eller på olika maskiner. De kan vara skrivna i olika språk och dess struktur kan ha helt olika former. Det enda språkkravet som COM har är att koden är skriven i ett programmeringsspråk som kan hantera pekare.

Genom att följa COM-specifikationen kan utvecklare distribuera sina komponenter som binära filer utan att exponera deras algoritmer i form av källkod. Genom att distribuera komponenter i binär form elimineras också många kompileringsproblem som annars uppstår vid användning av källkod. En till fördel som binära komponenter har över källkod är språkoberoendet. Är det flera utvecklargrupper inblandade i ett projekt kan varje grupp välja programmeringsspråk oberoende av de andra grupperna.

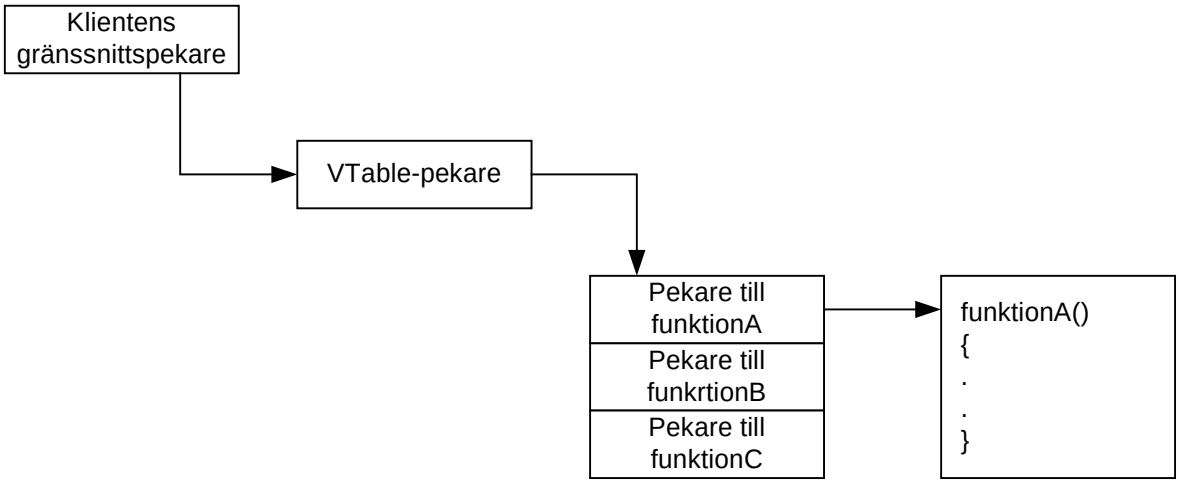
4.1.1 Begrepp

Nedan listas några korta förklaringar som(förhoppningsvis) kommer att göra det enklare att förstå efterföljande diskussioner.

- **Objekt**: ett objekt består av data och medlemsfunktioner som kan manipulera dessa data. Ett COM-objekt är ett objekt där åtkomst av data enbart kan ske genom en eller flera uppsättningar av relaterade funktioner. Dessa funktionsuppsättningar kallas *gränssnitt*(interface), och funktionerna i ett gränssnitt kallas *metoder*.
- **Komponent**: en COM-komponent består av ett eller flera COM-objekt. Objekten nås via var sitt gränssnitt som exponeras av komponenten. COM-komponenter är antingen DLL eller EXE filer. DLL-komponenter kallas *inprocess servers* eftersom de laddas in i samma process som den anropande klienten. exe-komponenter kallas *local servers* om komponenten exekveras i en annan process än klientens process men på samma dator och *remote servers* om komponenten körs på en annan dator än den klienten ligger på.

- **Gränssnitt**(interface): ett gränssnitt är det sätt på vilket ett objekt exponerar dess funktionalitet gentemot andra komponenter. I COM är ett gränssnitt en tabell av pekare(virtuell tabell) till objektets metoder. COM-standarden kräver att det enda sättet att få åtkomst till metoderna i ett gränssnitt är genom en pekare till gränssnittet. Enligt COM får inte gränssnitt ändras, metoder får varken läggas till eller tas bort.
- **Virtuell tabell**(v-table): när en applikation har en pekare till en COM komponent, har den egentligen en pekare till en pekare till en virtuell funktionstabell, eller v-table. Denna v-table innehåller pekare till funktioner inuti komponenten (Figur 2).
- **Klient och Server:** en COM-komponent kan vara både en klient och en server. En komponent betraktas som en server om den innehåller objekt som erbjuder sina tjänster till andra objekt. På samma sätt betraktas en komponent som en klient om den innehåller objekt som utnyttjar andra objekts tjänster. Naturligtvis kan en COM-komponent vara både server eller klient beroende på vilket scenarion det är frågan om.
- **Component Object Library (COL)**[5]**:** COL är en Windows systemkomponent som inkapslar allt arbete som har med att starta komponenter och etablera kontakt mellan dem. En klient, som vill utnyttja de tjänster som tillhandahålls av ett viss objekt, börjar med att anropa COL för att få en gränssnittspekare till den aktuella objektet. COL kommer sedan att utföra det nödvändiga "legoarbete" som är nödvändigt för att koppla ihop klienten och den komponenten som innehåller objektet. COL hittar och kör den DLL- eller EXE-fil som är associerad med komponenten för att sedan be denna att skapa en instans av sig själv (komponenten). Om allt går väl erhåller COL en gränssnittpekare till objektet som den returnerar till den anropande klienten.

[5] Component Object Library är en del av Microsofts operativsystem Windows NT. Den finns där i form av en binär fil.



Figur 2: En klient har egentligen en pekare till en pekare till en tabell som innehåller funktionspekare som pekar på de egentliga funktionerna i COM-objektet.

4.1.2 Komponentbaserat system

Förr i tiden byggdes stora applikationer genom att flera hundra källkodsfiler kompilerades på en och samma gång för att skapa en enda exekverbar fil. Denna utvecklingsmetodik skapade många problem, t.ex. väldigt stora filer och långa kompileringstider. Om en enda rad källkod ändras, måste alla filer omkompileras vilket gjorde koordinering mellan grupper i stora projekt väldigt svår. Dessutom var alla utvecklingsgrupper tvungna att använda samma programmeringsspråk.

Komponentbaserade system löser många av de ovannämnda problemen genom att utnyttja binära komponenter. Varje utvecklingsgrupp skapar en färdigkompilerad komponent istället för en massa källkodsfiler. Upptäcker man fel i en komponent behöver endast den kompileras om och inte hela applikationen. Eftersom komponenterna är binära filer kan utvecklingsgrupperna jobba med olika programmeringsspråk utan att det påverkar slutprodukten. Detta minskar utvecklingstiderna radikalt. T.ex. kan en utvecklingsgrupp som utvecklar lågnivåkomponenter använda Microsoft Visual C++, medan en annan utvecklingsgrupp som utvecklar användargränssnitt kan använda det mer lättanvända Microsoft Visual Basic.

För att kunna utnyttja alla de fördelar som komponentbaserade system för med sig måste man följa någon bestämd komponentteknologi. Som det redan har nämnts finns det två stora teknologier (och flera mindre) ute på marknaden, COM och CORBA. En av dessa teknologier (i detta examensarbete valdes COM) måste utnyttjas för att kunna bygga flexibla system och undvika de problem som associeras med komponentbaserade system.

Problem som ska lösas i ett komponentbaserat system:

- **Interoperabilitet**: hur kan komponenter från olika leverantörer interoperera på ett säkert sätt?

- **Versionshantering**: Hur kan nya versioner av komponenter skapas och distribueras samtidigt som de ska bibehålla bakåtkompatibilitet mot gamla klienter?
- **Språkoberoende**: hur kan funktionsanrop ske mellan komponenter skrivna i olika språk?
- **Transparent remoting**: hur kan en komponent kommunicera med andra komponenter utan att ta hänsyn till om dessa komponenter körs i samma process, i en annan process på samma maskin eller på en helt annan maskin?
- **Plattform**: i ett större system kan komponenter ligga på olika plattformar. Hur kan problem som uppstår på grund av olika plattformar lösas?

COM har lösningar på alla dessa problem och de kan korfattat beskrivas så här:

- **Interoperabilitet**: En klient erhåller tjänsterna av en COM-komponent genom att anropa gränssnitten för de objekt som finns i komponenten. När en komponent anropar en annan komponent kan följande fråga ställas: hur kan detta anrop ske när den andra komponenten kan vara skriven vid en annan tidpunkt, av utvecklare på ett annat företag och i ett annat programmeringsspråk? Svaret är enkelt om man följer COM. COM möjliggör anrop mellan komponenter genom att specificera en binär standard för sådana anrop. Standarden specificerar hur objekt i komponenter konstruerar den virtuella tabellen för sina gränssnitt och hur klienten anropar gränssnittens metoder indirekt via den virtuella tabellen.
- **Versionshantering och bakåtkompatibilitet**: COM tillhandahåller versionshanteringsmekanismer som möjliggör en smärtfri uppgradering av komponenterna. Existerande klienter behöver inte omkompileras då en komponent uppdateras. Den nya versionen av komponenten bibehåller bakåtkompatibilitet med gamla klienter samtidigt som den tillhandahåller nya funktionaliteter som kan användas av nya klienter. Eftersom det inte är möjligt att addera nya funktionaliteter till ett gammalt gränssnitt, är ett nytt gränssnitt det enda sätt som en komponent kan tillhandahålla nya funktionaliteter. Detta nya gränssnitt är inte en ny version av ett gammalt gränssnitt, utan är ett helt nytt gränssnitt. Bakåtkompatibilitet bibehålls eftersom de gamla gränssnitten finns kvar. OBS! så länge som syntaxen och betydelsen av ett gränssnitts metoder bibehålls kan metoderna förbättras för att ge bättre prestanda. Existerande klienter anropar dessa förbättrade metoder som de alltid gjort och får då bättre prestanda "på köpet".
- **Språkoberoendet**: COM är en språkoberoende standard. Genom att specificera en binär standard (till skillnad från källkodstandard i C++), möjliggör COM anrop mellan komponenter skrivna i olika språk.
- **Transparent remoting**: COM komponenter kan implementeras på tre olika sätt: *inproccess, local* och *remote*. Klienter vet inte och behöver inte heller veta vilken typ av komponent de anropar. Klienten använder helt enkelt samma programmeringsmodell när den anropar alla sorters komponenter och det tack vare COL. COL gör att klienten inte behöver veta vilken typ av

komponent den pratar med, utan skillnader mellan *inproccess*, *local* och *remote* komponenter är transparenta för både klienten och komponenten.

- **Plattform:** COM-komponenter måste köras på ett operativsystem från Microsoft. Tyvärr går det inte att blanda olika plattformar för ett system som bygger på COM-komponenter. Detta håller dock på att förändras då COM blir applicerbar på flera olika plattformar inom en snar framtid.

DCOM (Distributed COM) är en vidareutveckling av COM som lanserades av Microsoft 1996 tillsammans med Windows NT 4. DCOM utökar möjligheterna i COM till att gälla över nätverk. Tidigare kunde COM endast användas då komponenter på samma dator skulle kommunicera, men med DCOMs intåg kunde nu komponenterna kommunicera fritt med andra datorer på ett nätverk. Nuförtiden skiljer man inte på COM och DCOM, utan använder endast termen COM.

4.1.3 COM vs CORBA

Förutom COM finns det en annan accepterad komponentarkitektur som heter CORBA (Common Object Request Broker Architecture). CORBA, som är utvecklat av OMG (Object Management Group), introducerades i början på 1990-talet och har, precis som COM, fått en stor spridning världen över. En stor fördel som CORBA har över COM är att CORBA är plattformsoberoende. Med det menas att CORBA fungerar på alla existerande operativsystem, medan COM endast fungerar på operativsystem från Microsoft. Dock håller detta på att förändras då Microsofts COM börjar bli användbar i andra operativsystem som t.ex. Solaris UNIX, Digital UNIX och Macintosh.

Eftersom INEO är en Microsoft Solution Provider (INEO jobbar endast med Windows NT lösningar) krävde uppdragsgivaren att jag skulle använda mig av COM och inte CORBA.

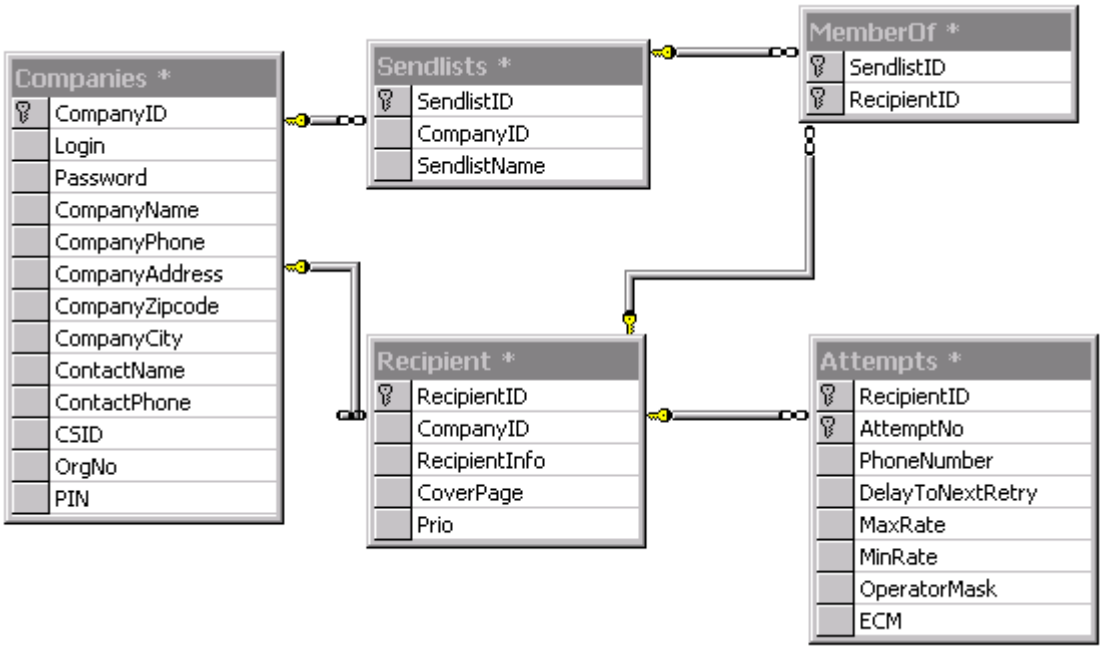
4.2 Dokumentkonverteraren

Det var länge osäkert hur filkonverteringsbiten i WebFax skulle lösas. Till att börja med prövade jag att söka på Webben i förhoppning om att jag skulle hitta ett program för detta ändamål. Trots en grundlig genomsökning på Webben kunde jag inte hitta något program som kunde utföra konverteringar mellan de önskade dokumentformaten. Nästa utmaning blev att söka bland Microsofts dokumentation, men jag insåg direkt hur dålig dokumentationen var (Microsoft antydde att det skulle bli bättre till Windows 2000). Då det var tidspress på projektet beslutade jag mig för att använda en printerdrivrutin från BlackIce. Detta begränsade konverteringsmöjligheterna till endast ett fåtal filformat. Inom ramen för examensarbetet kommer konverteraren att endast kunna konvertera två filformat, doc och txt.

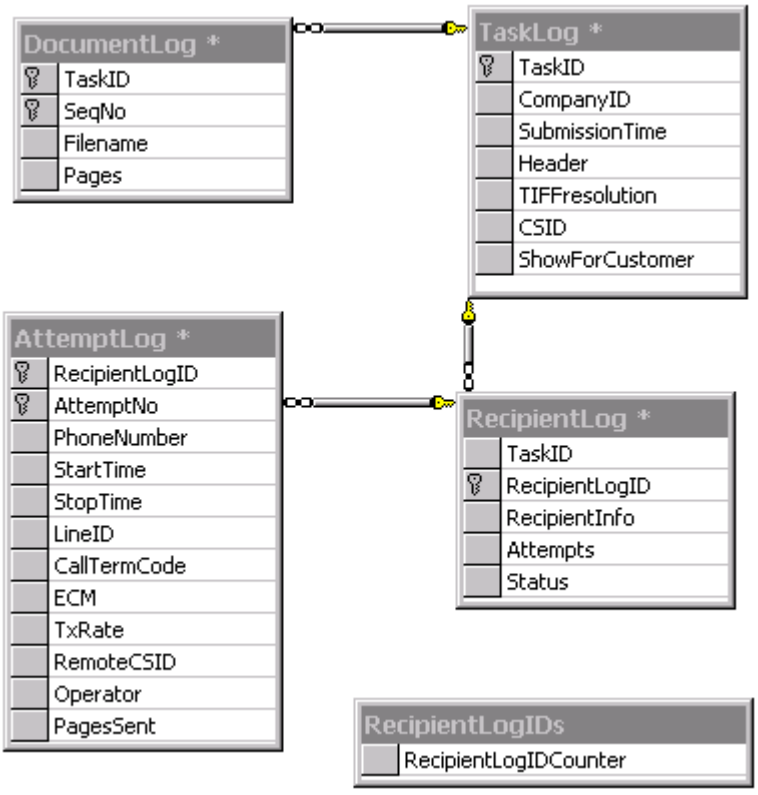
4.3 Databas

SQL Server 7.0 [RSKD99] visade sig vara ett klokt val av databasserver och jag hade inte några som helst problem med den implementerade databasen. Själva databasdesignen var mycket enkel och resulterade i några mycket enkla tabeller. Databasen kan delas in i två delar som vi kan kalla WebFaxDB och WebFaxLogDB. WebFaxDB (Figur 3) består av fem tabeller som innehåller information om varje användare samt information om alla sändlistor och deras relaterade mottagare. WebFaxLogDB (Figur 4) består av fem tabeller som innehåller information om varje faxutskick som skapats i systemet. Det är denna del av databasen som ligger som grund för de utskicksrapporter som användaren kan se via webben.

Förmodligen kommer nya tabeller att läggas till och de gamla tabellerna att modifieras under det fortsatta utvecklingsarbetet då nya finesser ska läggas till eller tas bort i systemet.



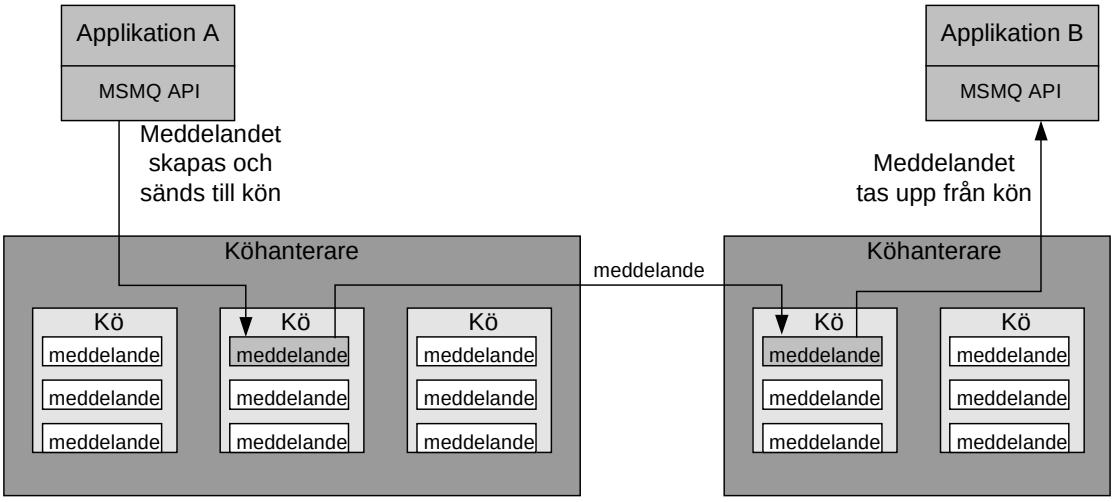
Figur 3: De fem tabeller i databasen som innehåller information om användaren och dess sändlistor med tillhörande mottagare och omförsök. Förbindelserna mellan tabellerna är "en till många" relationer (markeras i figur med en nyckel förbunden med ett oändlighetstecken). T.ex. finns det en sådan relation mellan CompanyID i Companies-tabellen och CompanyID i Sendlists-tabellen.



Figur 4: De fem tabeller i databasen som innehåller information om hur varje enskilt faxutskick gick. RecipientLogIDCounter-tabellen används för att generera unika till RecipientLog-tabellen.

4.4 MSMQ och köer

Ett av de största problem i ett komponentbaserat system är att kunna möjliggöra för komponenterna att på ett enkelt och säkert sätt föra information mellan varandra. Att använda köer för denna informationsförmedling mellan komponenterna visade sig vara ganska enkelt. Applikation A skapar ett meddelande och skickar det till en kö. Applikation B kan sedan läsa detta meddelande från den nämnda kön och om nödvändigt skicka tillbaka ett svar i form av ett nytt meddelande. Detta är den enklaste formen av meddelandehantering. Se Figur 5 för ett lite mer avancerat exempel. Naturligtvis kräver komponenterna i WebFax mer finesser än att bara kunna skicka och ta emot meddelande via köer. Efter att ha undersökt MSMQ från Microsoft kunde jag konstatera att nästan alla finesser som behövdes i WebFax fanns med i den.



Figur 5: Informationsförmedling i ett komponentbaserat system. Ett av de många sätt att skicka meddelande mellan två applikationer.

MSMQ (Microsoft Message Queue) är en utmärkt köhanteringsserver från Microsoft som ingår som en del i Windows NT. MSMQ gör det möjligt att skicka meddelanden asynkront mellan applikationer på en dator eller mellan applikationer på olika datorer sammanknutna via ett nätverk. Den tillhandahåller också ett flertal COM-objekt och ett API (Application Programming Interface) som gör det möjligt för utvecklare att använda alla de funktionaliteter som erbjuds av den.

Idén med MSMQ är att en applikation skapar ett meddelande och skickar ner det i en kö. En annan del av applikationen, eller en helt annan applikation, kan sedan läsa upp detta meddelande. I MSMQ ingår det två sorters klienter och en server.

- **Servern**: Det måste finnas minst en server för varje installation av MSMQ. Det är den viktigaste delen i ett MSMQ system. Servern innehåller bl.a. köer, köhanterare, MSMQ API och mjukvara för att dirigera meddelanden mellan köer.
- **Oberoende klient**: denna klient stödjer också MSMQ API och har en meddelandedirigerare med egna köer. På denna oberoende klient kan applikationer sända meddelanden när som helst eftersom de kan sparas i klientens köer. Denna typ av klient är idealisk för datorer som inte har kontinuerlig kontakt med andra klienter eller servrar.
- **Beroende klient**: denna typ av klient är ännu mer begränsad än oberoende klienter. Den har ingen meddelandedirigerare och inga egna köer. Den kan endast stödja applikationer som skickar eller tar emot meddelande. Eftersom denna typ av klient inte har några egna klienter måste den hela tiden ha kontakt med en MSMQ server. Denna typ av klienter används mer sällan och endast med kontinuerlig kontakt med en server.

En av de stora fördelarna med MSMQ är att den också kan tillhandahålla transaktionsköer. Transaktioner utnyttjas då man har två eller flera uppgifter och vill att alla utföras eller ingen alls. Ett välkänt exempel är bankärenden. Vid överföring mellan två konton finns det två uppgifter som måste utföras, dra av pengar från första kontot och sätta in pengar i det andra kontot. Om någon av uppgifterna inte kan utföras får den andra uppgiften inte utföras heller.

I MSMQ kan en användare bestämma om en kö ska kunna hantera transaktioner eller inte. Köer som kan hantera transaktioner garanterar att ett meddelande aldrig försvinner, även om hela systemet kraschar. Detta är en mycket trevlig egenskap som med fördel kan utnyttjas i WebFax. Alla komponenter i WebFax blir då oberoende av varandra och kan forsätta skicka meddelande mellan varandra även om en komponent skulle krascha. När denna komponent senare startas om kan den läsa de meddelanden som var ämnade för den under tiden som den var nere.

För att kunna skicka meddelande till en kö måste kön först öppnas. Nästa fas är att skapa ett MSMQ meddelande som ska bära det data man vill skicka. Ett MSMQ meddelande består av en kropp, en etikett samt en hel del parametrar som kan ändras efter önskat behov. Man kan t.ex. specificera att ett meddelande, som inte nått sin destination inom en viss tid, ska slängas. Etiketten kan endast innehålla en textsträng medan meddelandets kropp kan innehålla strängar, vektorer och till och med COM-objekt. Det enda kravet som ställs på COM-objektet för att kunna skickas i kroppen på ett meddelande, är att det kan lagra sig själv på fil. Hade det kravet inte funnits hade MSMQ inte heller kunnat garantera meddelandeleverans. Om objektet inte kan sparas på fil kommer det att endast vara lagrat i datorns primärminne och skulle då förloras vid ett eventuell datorhaveri.

En mycket irriterande begränsning som finns i MSMQ är att kroppen på ett MSMQ-meddelande inte få vara större är 4 Mb. En däremot mycket trevlig egenskap är att köer har möjlighet att generera en händelse som talar om för applikationer, som lyssnar på kön, att ett nytt meddelande har anlänt. På detta sätt slipper man ställa upprepade frågor mot kön. En annan trevlig egenskap som finns i MSMQ är att varje meddelande kan ha en prioritet (talen 0 till 7 där 0 är högsta prioritet).

Relaterade artiklar som jag fann nyttiga vid instuderingen av MSMQ: [DC98].

4.5 ADO och dataåtkomst

Microsoft har tidigare haft flera olika dataåtkomstteknologier som kunde användas beroende på vilket sätt data var lagrat. Detta bringade mycket förvirring och iteration bland utvecklarna, och Microsoft insåg att en enda dataåtkomstteknologi behövdes. Mycket arbete lades på att göra denna nya teknologi till en universell (COM-baserad) dataåtkomstteknologi. Vidare skulle den vara lätt att använda och flexibel att vidareutveckla. Resultatet att Microsofts arbete blev Active Data Objects[6] [DSAH98].

[6] Ibland kallad ActiveX Data Objects

Active Data Objects (ADO) är en uppsättning COM-komponenter som möjliggör åtkomst åt alla typer av datalagringsformat. De tre viktigaste objekten i ADO är *Connection*, *Recordset* och *Command*, varav jag endast använde de två första. Connection-objektet används för att etablera kontakt som möjliggör åtkomst åt data i en databas. Recordset-objektet används för att komma åt data som returneras efter att ha exekverat en SQL-fråga.

4.6 Webben

Web-klienten som ingår i WebFax behöver klara av mer än att bara visa statiska HTML-sidor. Data måste kunna hämtas från databasen och visas på Web-klienten, vilket statiska HTML-sidor inte klarar av. Därför var jag tvungen att utnyttja någon mekanism som gör Web-sidorna mer användarvänliga. Eftersom hela systemet utvecklas i Microsoft Windows miljö låg Internet Information Server (IIS) [MM99] bäst till hands. IIS 4.0 är senaste version av Microsofts ledande kommersiella Web-server och är en viktig del i Microsofts Internetstrategin.

Fundamentalt för Web-utveckling med IIS är Active Server Pages (ASP) [FF98] version 2.0. Enkelt uttryckt är en Active Server Page en fil som innehåller en kombination av HTML kod och script. Istället för att ha filändelsen .htm har ASP-sidor ändelsen .asp. Då IIS tar emot en HTTP begäran för en ASP-fil kommer den att dynamiskt generera en HTML-fil från den statiska HTML-koden och den HTML-kod som genereras av scripten.

4.6.1 Script

Det finns två[7] dominerande scriptspråk för utveckling av Web-sidor i Windows-miljö, JavaScript och VBScript. JavaScript är en delmängd av det nu världsberömda programmeringsspråket Java, medan VBScript är en delmängd av Microsofts Visual Basic. Båda är lika enkla att använda, men det finns en mycket viktig skillnad mellan dem. Skillnaden är att VBScript endast fungerar på Microsofts Web-bläddrare Internet Explorer (IE) medan JavaScript fungerar på både IE och Netscape Navigator.

Script kan köras antingen på klientdatorns Web-bläddrare eller på Web-servern, därav benämningarna klientscript och serverscript.

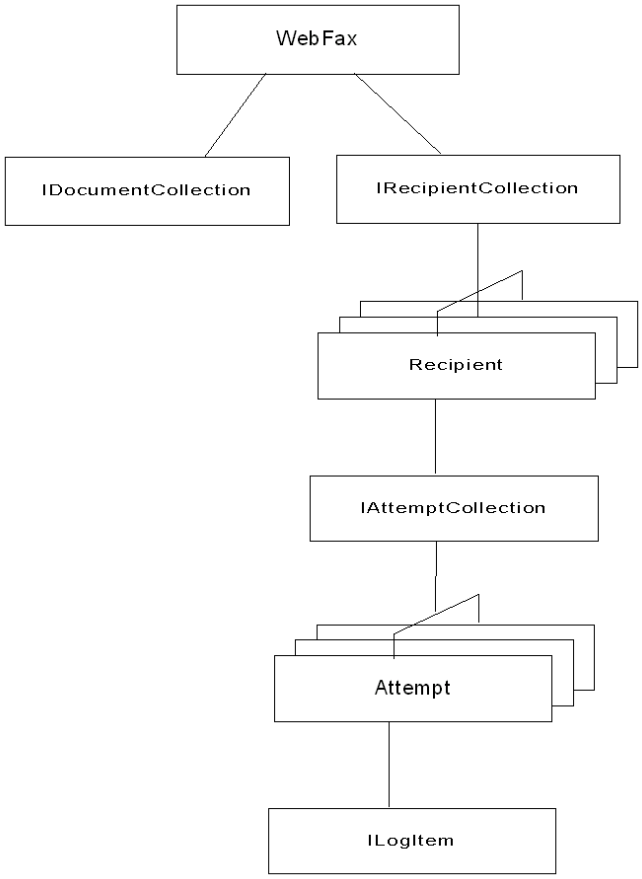
- **Klientscript**: körs i Web-bläddraren på klientdatorn. Bläddraren känner av att det finns ett script. Den främsta anledningen till att ha klientscript är att kunna skapa händelseprocedurer för kontroller. Om en användare t.ex. trycker på en knapp kan man genom klientscript generera en händelse som svar på knapptryckningen. Klientscript är varken kompilerad eller krypterad kod, utan vem som helt kan komma åt innehållet, inklusive Web-klienten.
- **Serverscrcipt**: dessa script körs i en ASP på Web-servern innan servern skickar iväg sidan till klientdatorn. När en klientdator begär en

[7] Perl (Practical Extraction and Reporting Language) är ett tredje mycket spritt scriptspråk men som används allt mindre (nästan inte alls) vid utveckling av Web-sidor i Windows-miljö.

ASP-sida kommer all serverscript att köras och genererad HTML-kod skickas till klientdatorn. Serverscript har fördelen att den kan utnyttja serverdatorns resurser och att Web-bläddraren inte behöver veta om det är VBScript eller JavaScript. Det är ju trots allt bara HTML-kod som skickas tillbaka till bläddraren.

5 Resultat

Som det ser ut vid skrivandets stund består WebFax av fem Windows NT tjänster (services), en Web-klient samt två COM-komponenter. Själva kärnan i hela WebFax är en COM-komponent som kallas Webfax01. Webfax01 (Figur 6) innehåller alla objekt som hanterar all data som ska flyttas mellan de fem olika Windows NT tjänsterna. Webfax01 har skapats i Microsoft Visual C++ (VC++) och alla objekt har skapats med tanke på att de ska vara *persistable* (kunna lagra sig på fil), med tanke på kravet från MSMQ.



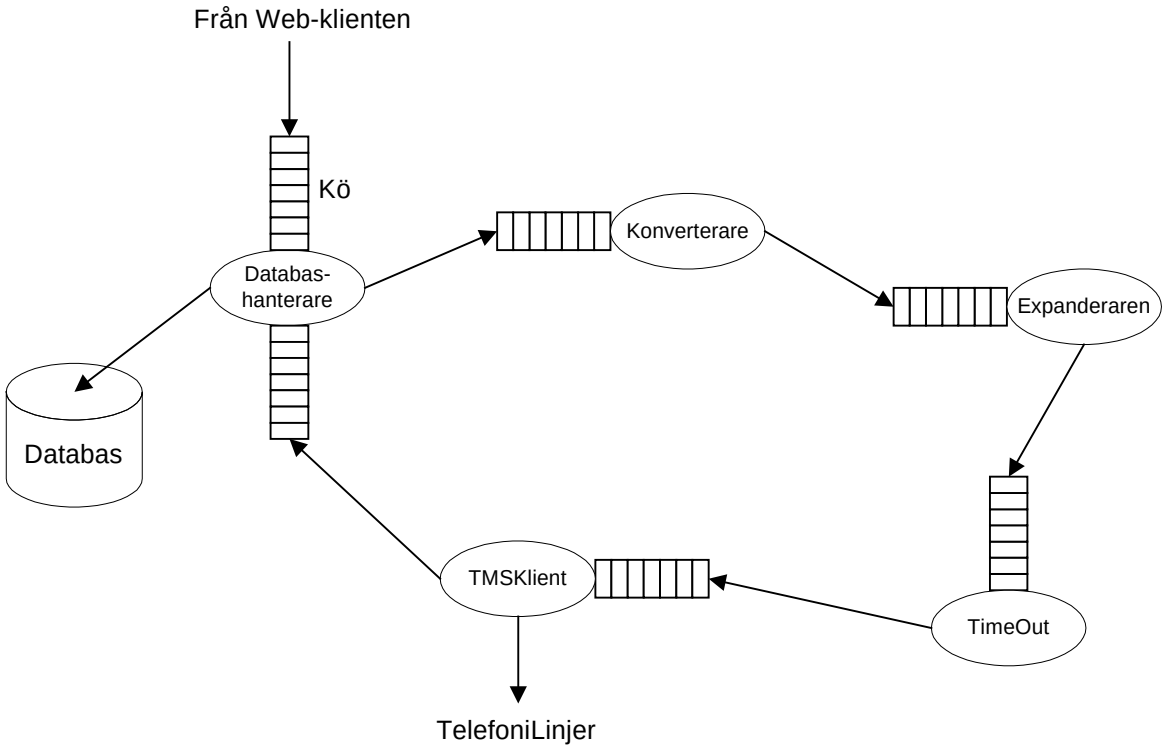
Figur 6: Alla de objekt som finns i Webfax01 komponenten. OBS! det finns inget "C++ arv" i figuren. Figuren visar att WebFaxobjektet består av ett IDocumentCollection-objekt ("I" står för *Interface*) och ett IRecipientCollection-objekt. Ett IRecipientCollection-objekt består i sin tur av en mängd Recipientobjekt osv.

Den andra COM-komponenten, WebFaxNewTask, är också skapad i VC++. Den används av ASP-sidorna då användaren skapar ett nytt faxutskick. Denna komponent innehåller endast ett COM-objekt kallat NewTask. Då användaren skapar ett nytt faxutskick fylls NewTask-objektet med data (mottagarnas ID) och läggs sedan ner i kroppen på ett MSMQ meddelande för vidarebefordring mot övriga delar i WebFax. Då NewTask-

objektet endast innehåller heltal är det väldigt liten risk att 4 MB gränsen för ett meddelande i en MSMQ-kö överskrids.

Web-klientsidorna blev till stor del som det var tänkt från början. Utvecklingen skedde i Visual InterDev 6.0, men ej helt problemfritt. Tyvärr lyckades jag aldrig få avlusningsmekanismen i InterDev att fungera, vilket gjorde att utvecklingen av Web-klienten tog längre tid än beräknat. En hel del problem uppstod också med åtkomsträttigheter på Web-servern. Web-klienten är ännu ej färdig för driftsättning, men fungerar väl som ett demonstrationsprogram.

De övriga delarna av WebFax är de fem (Figur 7) Windows NT tjänsterna med tillhörande köer. Alla tjänster är skrivna i Microsoft Visual Basic (VB) eftersom det är ett enkelt utvecklingsprogram. För att kunna skapa dessa systemtjänster i VB (VB klarar inte av att skapa tjänster) var jag tvungen att använda en COM-komponent (NTSVC.OCX) skriven i VC++ och som fanns att ladda ner från Microsofts hemsida.



Figur 7: De fem tjänsterna samt de köer som hanterar informationsförmedlingen mellan tjänsterna.

Var och en av tjänsterna beskrivs nedan:

- **Databashanteraren**: denna tjänst tar hand om all kommunikation som går mellan de övriga tjänsterna och databasen. På detta sätt blir all kod för databasåtkomst samlad på ett och samma ställe, vilket förenklar framtida

modifieringar. Databashanteraren tar emot det NewTask-objekt som skapas då en användare skapar ett nytt faxutskick via Webben. I nästa steg registrerar den utskicket i databasen och skapar ett WebFax-objekt (Figur 6) som den sedan skickar till Konverterarens kö.

- **Konverteraren**: det enda, men viktiga, denna tjänst gör är att utföra dokumentkonverteringen till TIFF-format. Den plockar upp WebFax-objektet ur meddelandet och tar fram listan med alla dokument som ska konverteras. Efter en lyckad konvertering vidarebefordrar den WebFax-objektet till kön för nästa tjänst.
- **Expanderaren**: denna tjänst kommer att plocka upp WebFax-objektet och ta fram alla mottagare som finns lagrade i IRecipientCollection (Figur 6). I detta stadium fylls varje mottagarobjekt på med ytterligare information som tidigare låg på en högre nivå i WebFax-objekt för att undvika upprepning av samma data. Detta är till för att undvika den gräns (4 MB) för hur stora meddelandena får vara i MSMQ. När ett mottagarobjekt är klart skickas det till kön för "TimeOut"-tjänsten. I etiketten står det när detta meddelande ska faxas iväg.
- **TimeOut**: kön till denna tjänst lagrar alla meddelande som innehåller faxutskick som ska ske vid en senare tidpunkt. Ska meddelande skickas tas det omedelbart upp av TimeOut-tjänsten och skickas iväg till nästa tjänst. Ska det skickas vid en senare tidpunkt ligger det kvar orört tills tidpunkten för utskicket inträffar. Trots att kön för denna tjänst kan innehålla flera tusen (vid stora faxutskick) meddelanden, är det inte tidskrävande att titta på alla meddelande för att plocka upp endast de som ska skickas iväg. Detta tack vare att tidpunkten för utskicket står i etiketten, därmed behöver TimeOut-tjänsten inte plocka upp hela meddelandet utan endast titta på etiketten.
- **TMSklient**: denna tjänst sköter hand om själva distribueringen av utskicken på INEOs telefonilinjer. Om något av utskicken inte lyckades sätts det tillbaka i kön för TimeOut-tjänsten med en ny tid i etiketten. Detta kan inträffa högst fyra gånger eftersom användaren i dagsläget har möjlighet att endast definiera fem försök att göra utskicket på. I fall utskicket lyckas eller misslyckas registreras allt som sker i ILogItem-objektet. Detta objekt utnyttjas senare av DatabasHanteraren för att registrera hela utskicket i databasen.

Det var till en början tänkt att prioritet ska utnyttjas i MSMQ-köerna för att erbjuda vissa användare förtur i systemet, men eftersom alla köer är transaktionsköer föll den tanke. Det går ej ett utnyttja finessen med prioritet i transaktionsköer.

Ett stort hinder visade sig vara att kunna ladda ner filer från en klients dator. Efter en hel del teoriläsning och sökning på Internet bestämde jag mig för att utnyttja en COM-komponent som säljs av ett företag i USA. Komponenten, som heter ASPUpload, är mycket lättanvänd och är dessutom mycket billig. ASPUpload behöver endast installeras på Web-server på INEO. Klientbläddraren måste vara RFC[8] 1867-kompatibel för att uppladdning av filer skall vara möjlig. Netscape 3.0+ och Microsoft IE 4.0+ är RFC 1867-kopatibla.

[8] Request For Comments

Eftersom WebFax är byggt för att klara stora faxutskick har den inte riktigt kunnat testas. Det finns ingen möjlighet att på riktigt börja skicka ut flera tusen fax. Det kan först ske då INEO får en riktig kund. Tester har dock genomförts med hjälp av en faxsimulator som finns på INEO samt mindre utskick till en riktig fax på INEO och till en på Ericsson.

Köerna i systemet har belastats med flera tiotusentals meddelanden utan några problem. Det uppstod inte heller några fördröjningsproblem eftersom den långsammaste länken i hela systemet är den tjänst som ska skicka ut faxen på telefonilinjerna.

6 Slutsatser

Efter nästan sex månaders arbete står nu en fullt fungerande demoversion av WebFax klar att använda. Det har varit både upp och ner under arbetets gång, men framför allt har det varit mycket lärorikt. Att lära sig COM var verkligen en stor fördel för att bättre förstå alla andra produkter från Microsoft. Å andra sidan är COM ingen lätt arkitektur som man lär sig på några månader, utan fortfarande ser jag lite av dimman kvar framför mig.

Det jag känner så här dags är att exjobbet hade en ganska stor spridning. Det har inkluderat allt från lågnivåprogrammering till att skapa användargränssnitt. Jag har fått läsa en hel del böcker och artiklar (se referenslistan) för att lära mig COM, ASP, ADO, MSMQ, SQL Server, Windows NT Server m.m. Jag kan ännu inte påstå att jag behärskar alla dessa teknologier, men jag har kommit en bra bit på vägen.

Den största motgången under exjobbstiden har nog varit att skapa en dokumentkonverterare som är stabil och som klarar flera filformat. En vidareutveckling av denna del av WebFax kommer att ske för att kunna ge användare större valmöjligheter än bara Word- och textfiler. Microsoft har lovat bättre stöd i Windows 2000 för direkt lågnivåprogrammering gentemot lagrade dokument, t.ex. i Office, vilket jag ser fram emot. Kanske skall man beakta möjligheten att låta konverteringen ske hos användaren.

Ett generellt debiteringssystem är under utveckling, men är ännu ej färdig för användning. WebFax behöver utnyttja detta debiteringssystem i syfte att kunna få så billiga telefonikostnader som möjligt för användarna. Det finns också funderingar på om användaren ska, via rapporterna i Web-klienten, kunna se kostnader för varje enskilt utskick och på så sätt ha en kontinuerlig överblick över sina kostnader. Debiteringssystemet kommer att bestå av en COM-komponent med en tillhörande databas (en ganska komplex sådan).

När det gäller WebFax som helhet känns det som om det blivit lite för många delar. En möjlighet vore kanske att lyfta in implementeringen av Expanderaren in i någon av de andra tjänsterna. En annan förenkling som kommer att implementeras är att ersätta NewTask-objektet vid kommunikationen mellan Web-klienten och DatabasHanteraren med ADO. Detta gör att utvecklarna har två komponenter färre att hålla reda på.

7 Referenser

[DC98] David Chapell. Microsoft Message Queue Is a Fast, Efficient Choice for Your Distributed Application. *Microsoft System Journal*, juli 1998.

[Eddon98] Eddon, Eddon. *Inside Distributed COM*. ISBN 1-57231-849-X Microsoft PRESS.

[FF98] Francis, Fedorov, Harrison, Homer, Murphy, Sussman, Smith och Wood. *Professional Active Server Pages 2.0.* ISBN 1-861001-26-6. WROX.

[GSTR98] Grimes, Stockton, Templeman, Reilly. *Beginning ATL COM Programming.* ISBN 1-861000-11-1. WROX.

[AHDS98] Alex Homer, David Sussman. *Professional MTS MSMQ with VB and ASP*. ISBN 1-861001-46-0. WROX.

[MM99] Mark Minasi. *Mastering Windows NT Server 4*. ISBN 0-7821-2445-3. SYBEX.

[SS99] Steven Shaw, *IP Fax Enters the Standards Arena.* http://www.tmcnet.com/articles/itmag/0399/0399dialogic.htm

[RSKD99] Ron Soukup, Kalen Delaney. *Inside Microsoft SQL Server 7.0*. ISBN 0-7356-0517-3. Microsoft Press.

[DSAH98] David Sussman, Alex Homer. *ADO 2.0.* ISBN 0-861001-83-5. WROX.

[COMSPEC] http://www.microsoft.com/com/comdocs.asp

8 Ordlista

ADO	ActiveX Data Object
API	Application Programming Interface
ASP	Active Server Page
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
HTTP	Hyper Text Transport Protocol
HTML	HyperText Markup Language
IE	Internet Explorer
IIS	Internet Information Server
MSMQ	Microsoft Message Queue
OMG	Object Management Group
Perl	Practical Extraction and Reporting Language
RFC	Request For Comments
SQL	Structured Query Language
TIFF	Tagged Image File format
TMS	Tele Media Service
VBScript	Visual Basic Script