HANDELSHÖGSKOLAN
vid Göteborgs universitet
Institutionen för Informatik
Examensarbete I, 10p
ADB-programmet VT 1998

Två- och treskiktsmodellen i klient-serversystem under Windows NT

Författare: Patrik Turebo
Handledare: Birgitta Ahlbom

“If you are supporting more than 1.000 users on a single SQL-Server, you obviously don’t sleep much.”[1]

[1] Solomon, David m fl (1996)

SAMMANFATTNING

Under ett antal år har den traditionella tvåskiktsmodellen varit dominerande vid klient-serverapplikationer. Problemet med tvåskiktsmodellen är att säkerheten är svår att kontrollera. Varje klient kräver en koppling till databasen och all kod ligger oftast hos klienten. Detta kan lösas med användning av treskiktsmodellen. Treskiktsmodellen blir allt vanligare vid utveckling av klient-serverapplikationer. I ett skikt mellan klienten och servern återfinns transaktionshanterare som Microsoft Transaction Server (MTS). MTS använder sig av komponenter. Dessa komponenter förenklar kodskrivandet avsevärt. Komponenterna skrivs t ex i Visual Basic och kan sedan anropas från nästan vilket program som helst. Ändringar av kod görs endast på ett ställe. I detta arbete har jag inkluderat kodexempel där jag visar skillnaden mellan två- och treskiktsmodellen.

Webbläsaren har under de senaste åren växt fram som en ny typ av klientapplikation. Med webbläsaren kan en databas anropas från hela världen och med Active Server Pages kan kodningen av anropen till databasen göras mycket homogen.

FÖRORD

Efter två års studier på ADB-programmet har det blivit dags att skriva examensarbete. Examensarbetet utgör 10 högskolepoäng och är det moment som avslutar utbildningen.

Först och främst vill jag tacka Canopus AB för idén till examensarbetet, framförallt Daniel Broms som har hjälpt mig i många svåra situationer. Vidare vill jag tacka min handledare Birgitta Ahlbom för idéer och hjälp med utformningen av detta arbete. Jag vill även passa på att tacka alla de övriga personer som på något sätt har bidragit till detta arbete.

Vid eventuella frågor eller synpunkter är läsaren välkommen att kontakta mig.

Patrik Turebo
patrik.t@bigfoot.com

Göteborg 1998-05-28

INNEHÅLLSFÖRTECKNING

1 Inledning

1.1 Bakgrund

Efter att ha genomfört kursen "Handledd näringslivspraktik" på Canopus AB, var de liksom jag intresserade av ett fortsatt "samarbete". De gav mig ett förslag till examensarbete om transaktionshanteraren Microsoft Transaction Server (MTS), vilket jag tyckte lät intressant men svårt. Daniel Broms[2] hjälpte mig med att ta fram lämpligt grundmaterial för att läsa in mig på området. Dessutom gav han mig en mängd förslag på frågeställningar. Eftersom jag inte hade några förkunskaper på detta område har jag varit tvungen att lära mig det mesta från grunden. Detta har inneburit att arbetet behandlar MTS mindre än vad som var tänkt från början.

1.2 Syfte

Syftet med detta examensarbete är att försöka beskriva problematiken i klient-serversystem under Windows NT. Jag skall studera skillnaden mellan två- och treskiktsmodellen. Vidare skall jag undersöka hur komponenter/objekt[3] fungerar. Jag kommer att ge kodexempel på hur små enkla program kan byggas enligt de olika modellerna. Till sist skall jag undersöka vad transaktonshanteraren MTS är.

1.3 Frågeställningar

Från början var min förhoppning att behandla alla de frågeställningar som jag fick från Canopus. Detta har dock inte varit möjligt med tanke på den knappa tid som jag har haft till förfogande för detta arbete.

Jag skall i arbetet försöka behandla följande frågeställningar: Vad är skillnaden mellan två- och treskiktsmodellen? Vad är COM respektive DCOM? Hur görs databasanrop med Visual Basic (VB) via Open Database Connectivity (ODBC)? Hur görs databasanrop med Active Server Pages (ASP) via ActiveX Data Objects (ADO)? Hur används klasser i VB? Hur anropas COM och DCOM med VB och ASP? Vad är MTS och hur fungerar MTS? Hur utvecklas lämpligast komponenter med MTS? Hur ser framtiden ut för treskiktsmodellen och MTS?

1.4 Metod

De första veckorna lade jag ner mycket tid på att leta material om MTS och efter hand har jag erhållit mer och mer material. Jag har haft stor nytta av medarbetarna på Canopus, som har hjälpt mig i svåra situationer. Efter informationsinsamlingen har jag försökt testköra MTS och diverse kodexempel parallellt med författandet av uppsatsen.

---

[2] Daniel Broms är en av Canopus delägare.
[3] Ett objekt och en komponent är samma sak. I detta arbete använder jag mig av båda begreppen.

1.5 Avgränsningar

Eftersom transaktionsförfarandet handlar mycket om hur operativsystemet arbetar ihop med applikationen ger jag först en kort beskrivning av Windows NT. Programspråken liksom SQL-Server har jag inte beskrivit alltför detaljerat. Istället hänvisar jag till andra informationskällor. Beskrivningen av funktionerna i MTS är också begränsad.

1.6 Kommentarer

När man, som jag har gjort, studerar amerikansk litteratur kan det vara svårt för att inte säga omöjligt att översätta enstaka ord eller programtermer till svenska. Jag har i största möjliga mån försökt att få en bra översättning. Vissa program och kommandon i Windows har dock namn på både svenska och engelska. Första gången programnamn eller kommandon används har jag beskrivit dem genom att använda det svenska programnamnet med det engelska originalnamnet inom parentes, t ex Kontrollpanelen (Control Panel). Om ett programnamn eller ett kommandonamn förekommer upprepade gånger används det språk som jag anser lämpligast. Ett annat problem är långa programnamn och upprepningar av dessa. Detta har jag löst genom att använda programnamnet, t ex Microsoft Transaction Server, ihop med en förkortning, t ex MTS, första gången som jag skriver det i rapporten. Därefter använder jag mig av förkortningen i största möjliga grad. Vid kommentarer till kodexempel har jag till stor del skrivit dessa i kodexemplen för att spara utrymme och för att göra det mer lättläst.

2 Canopus AB

2.1 Allmänt

Canopus AB[4] är ett datakonsultbolag som ägs av tre personer och har två anställda. Canopus är *Microsoft Solution Provider*, vilket är ett samlingsnamn för Microsofts olika kunskapspartners. Detta innebär att Canopus nästan uteslutande arbetar med Microsofts produkter. De flesta program skrivs i Microsoft Access vilket innebär omfattande VBA-kodning. Andra verktyg som används i programutvecklingen är resterande Microsoft Office-produkter, Microsoft Visual Basic, Microsoft SQL-server mm. Vid utveckling av Intranet- och Internetprojekt används Microsoft Interdev ihop med ASP och VBScript.

2.2 Framtid med eller utan MTS?

Canopus funderar på att använda sig av MTS i framtida projekt. Det är även tänkt att vissa äldre projekt skall kunna överföras till att hanteras av MTS. Eftersom MTS är en relativt ny produkt är Canopus osäkra om man skall börja använda sig av produkten.

Figur 2-1 Canopus AB är Microsoft Solution Provider

[4] http://www.canopus.se

3 Windows NT

3.1 Allmänt

Windows NT (New Technology) är ett operativsystem som började utvecklas av Microsoft 1988. Tanken var att NT skulle vara tillräckligt säkert för affärskritiska tillämplingar. Övriga mål med NT var t ex skalbarhet, kompabilitet.[5] Den första lanseringen av Windows NT skedde 1992. I och med att Windows 3.1 lanserades ungefär samtidigt fick även Windows NT versionsnumret 3.1. Därefter har tre versioner lanserats 3.5, 3.51 samt 4.0. När Microsoft lanserade version 3.51 ökade användandet markant av NT Workstation. I och med version 4.0 fick NT samma utseende som Windows 95. Nu håller Microsoft på att utveckla Version 5.0 som finns i betaversion. Från att för några år sedan legat efter Unixleverantörerna i utvecklingen är det nu Microsoft som styr och leverantörerna av Unix följer efter.[6]

Skillnaden mellan NT Server och NT Workstation är den att NT Server körs på serverdatorer medan NT Workstation är tänkt att användas på klientdatorer till serverdatorerna.

3.2 Trådar

Windows NT möjliggör s k multithreading vilket innebär att ett program kan delas upp i ett antal separata trådar (delar) som körs oberoende av varandra. En tråd är ett kodsegment och tillhör en specifik process. Varje tråd tilldelas ett prioriteringsnummer från 0 till 31. Användning av multithreading effektiviserar hårdvaran.

3.3 Registret

Registret *(Registry)* har en enorm betydelse för att Windows NT skall fungera. Här lagras all konfigureringsinformation i en hierarkisk databas. För att köra registret öppnas "Kör" (Run) från Startmenyn, därefter skriver man "Regedit" och klickar på OK. Här finns all information om den lokala datorn lagrad. En ogiltig ändring i registret kan innebära instabilitet eller en krasch av datorn.

[5] Garms, Jason m fl (1996), sid 4
[6] Garms, Jason m fl (1996)

3.4 Kernel

Kerneln i är det beslutande organet i NT. Kerneln är ytterst ansvarig för alla handlingar i operativsystemet och nästan alla systemets funktioner passerar kerneln. Kerneln består av diskret kod och bygger upp kärnan av operativsystemet. Administrationen och schemaläggningen av trådar är den största uppgiften som kerneln ansvarar för. Kerneln ger varje tråd tillstånd att exekveras under en viss tid innan en annan process tillåts köra. Detta kallas för "preemptive multitasking". Tråden exekveras isolerat och får inte tillgång till mer minne eller processresurser än den tilldelats. Kerneln kan gå in och stoppa en process vid en viss tidpunkt för att överlåta processortiden till en annan process. När kerneln är upptagen med annat överlåter den uppgifter till NT-Executive.

4 Beskrivningar av program och programspråk

4.1 Microsoft SQL-Server

Microsoft SQL-Server är ett databasprogram som går att använda ihop med MTS. Det är även mycket användbart ihop med Windows NT och Internet Information Server. I mina tester har jag använt mig av version 6.5.

4.2 VB - Visual Basic

VB är ett händelseorienterat språk och ett av de lättaste och mest kraftfulla kodverktyg som går att finna på marknaden. 1997 fanns det 4.600 program tillgängliga för Windows 95 som var skrivna i VB.[7] Variationen på vad de används till är mycket stor, allt ifrån spel till applikationer med databaskopplingar. VB har följt med i Microsoft snabba utveckling och den senaste versionen är VB 5.0. VB:s kod ligger till grund för Visual Basic for Applications (VBA), Visual Basic Scripting Edition (VBScript) och Active Server Pages (ASP), vilka beskrivs nedan.

4.3 VBA - Visual Basic for Applications

VBA är kod som kan skrivas i alla Microsoft Office 97 applikationer. VBA är mycket lik ren VB-kod och var till en början ett makrospråk i Word och Excel men allt eftersom har VBA inkluderats i samtliga Office-program.

4.4 VBScript - Visual Basic Scripting Edition

VBScript är VB-kod som bäddas in som ett script i ett HTML-dokument och körs när anrop sker. Problemet är att inte alla webbläsare stöder VBScript. Ett annat problem är att det inte går att dölja koden, eftersom scriptet visas när webbläsaren visar källan.

4.5 ASP - Active Server Pages

ASP är ett funktionsspråk och används tillsammans med något av Microsofts webbserverprogram, Personal Webserver eller Internet Information Server (version 3.0 eller senare). Finessen med ASP jämfört med scriptspråken är att ASP exekveras direkt på webbservern. Det går alltså inte att skriva koden, lagra den lokalt och titta på den i webbläsaren. Koden måste först lagras under webbserverprogrammet och sedan öppnas via webbläsaren.

Med ASP kan samma komponenter anropas som anropas från t ex VB, vilket gör att mycket dubbelarbete undviks. Vid anrop av en ASP-sida omvandlar webbserverprogrammet koden till vanlig HTML-kod. Denna kod skickas sedan till den anropades webbläsare. Detta innebär att källan till ASP-dokumentet enbart innehåller

[7] McKelvy Mike m fl (1997), sid 9

HTML-kod och inte avslöjar något om hur ASP-koden är skriven. ASP-koden inkluderas i den vanliga HTML-koden med **<%** för kodstart och **%>** för kodslut. Kommentarer i ASP görs på samma sätt som i VB. Texten remmas med '.

I kodexempel 4-1 ges ett enkelt exempel på ASP-kod. Sidan börjar med en h1-rubrik (ASP-test). Därefter används ASP-teknik för att få fram datum (<%=date%>) samt klockslag (<%=time%>). För att visa hur ASP fungerar med selektionsteknik har jag inkluderat en if-sats som kontrollerar om åldern (intAge) är större än 17. Efter detta skrivs intAge ut samt om personen är myndig eller omyndig. I det här exemplet har jag hårdkodat intAge till 16 vilket innebär att utskriften blir "16år=omyndig!".

ASP-kod

```
<html>
<body>
<h1>ASP-test</h1>
Datum: <%=Date%> <br>
Tid: <%=Time%> <br>
<% 'Detta är en kommentar som inte påverkar koden
intAge=16
if intAge > 17 then %>
<%=intAge%>år=myndig! <br>
<%else%>
<%=intAge%>år=omyndig! <br>
<%end if%>
</body>
</html>
```

HTML-källa

```
<html>
<body>
<h1>ASP-test</h1>
Datum: 5/24/98 <br>
Tid: 11:04:55 PM <br>
16år=omyndig! <br>
</body>
</html>
```

Kodexempel 4-1 **ASP-kod visar hur ASP-filen ser ut. HTML-källa visar hur denna kod ser ut i webbläsarens källa.**

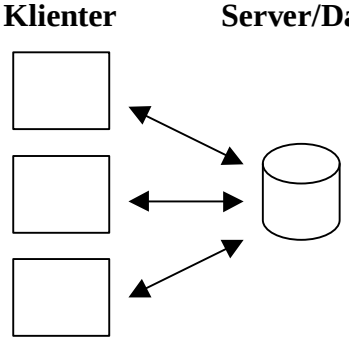
5 Klient-server

5.1 Allmänt

Klient-server är en speciell form av distribuerad databasbehandling. Funktionerna delas upp på ett transparent sätt mellan en klient och en server som är sammankopplade via ett nätverk. Klienten är en applikation som hanterar det grafiska gränssnittet mot användaren. Servern innehåller databasen som klienten anropar.

5.2 Tvåskiktsmodellen

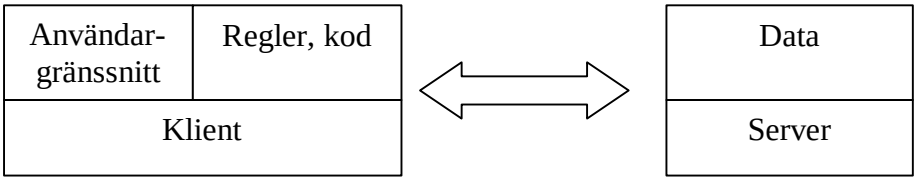
Tvåskiktsmodellen har sedan ett antal år tillbaka varit den mest populära modellen i att utveckla klient-serverapplikationer. I denna modell kommunicerar klienten (applikationen) direkt med servern (databasen). Detta beskrivs grafiskt i figur 5-1. Klientens huvudsakliga uppgift är att upprätthålla ett gränssnitt varifrån användaren kan skicka förfrågningar till servern.[8] På dessa förfrågningar returnerar servern data som klienten kan använda. Klientens roll är dessutom att validera data, exekvera kod samt att administrera lokala systemresurser.



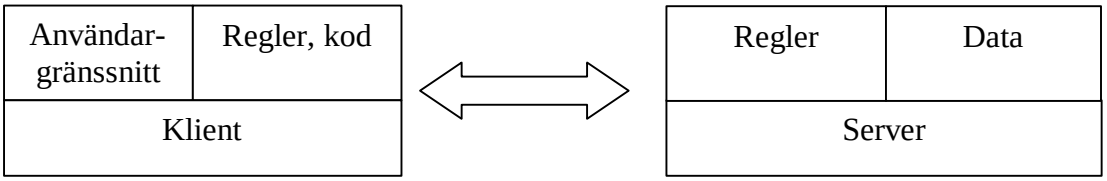
Figur 5-1 Tvåskiktsmodellen
Källa: Jennings R, m fl (1997), sid 13

Att utveckla system enligt tvåskiktsmodellen leder ofta till problem. Ett av problemen är att anropen sker direkt mot databasen och att varje anrop kräver en koppling till databasen. Ett annat problem är vad koden och reglerna skall placeras. Detta problem beskrivs i figur 5-2 och figur 5-3. Om ett system består av 200 klienter och använder sig av en modell lik figur 5-2 måste var och en av dessa klienter ha kod inkluderad i applikationen, vilket i tvåskiktsmodellen är den enklaste och smidigaste lösningen. Detta leder dock till stor minnesförbrukning och administrativa problem när applikationsprogrammet skall ändras. I vissa fall kan det vara lämpligt att databasen avlastar klienten och t ex sköter säkerhet, transaktionsprocesser, backup och återskapning av kritisk data, vilket visas i figur 5-3.

[8] Jennings, Roger m fl (1997) sid 23ff



Figur 5-2 Konfigurering av tvåskiktsmodellen.
Källa: Jennings R m fl (1997), sid 23

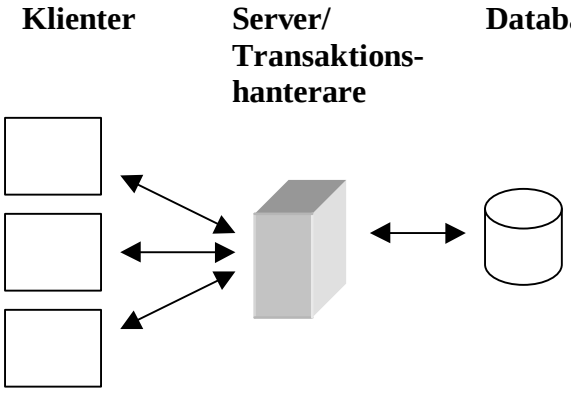


Figur 5-3 Konfigurering av tvåskiktsmodellen som använder servern till viss validering av regler.

Källa: Jennings R m fl (1997), sid 25

5.3 Treskiktsmodellen

Treskiktsmodellen är en modellarkitektur som blir allt vanligare. Programmen finns hos klienten, på en server samt i databasen. Programmen hos klienten styr användargränssnittet och anropar serverprogram (komponenter) som i sin tur skapar objekt som utför kommunikationen med databasen (se figur 5-4). Det är dessa objekt och komponenter som är det fördelaktiga i treskiktsmodellen, eftersom klienten aldrig jobbar direkt mot databasen. Det är i detta mellanskikt (middleware) som transaktionshanteraren MTS återfinns.



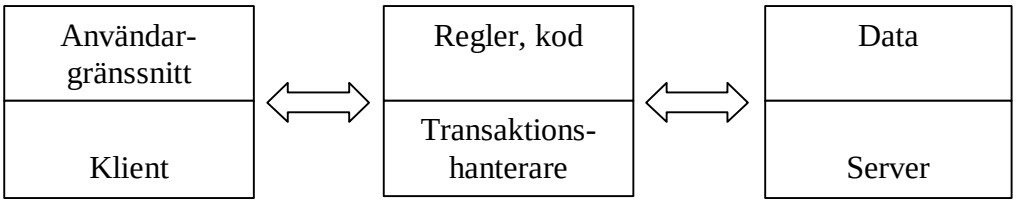
Figur 5-4 Treskiktsmodellen
Källa: Jennings R, m fl (1997), sid 13

Istället för att ha hundratalet klienter som var och en hanterar all kod lagras ytterst lite kod i klientapplikationerna. Koden läggs istället i komponenter som alla klienter kan använda sig av. Detta innebär att ändringar i koden oftast görs på ett ställe. Dessutom minskas minnesförbrukandet.

Nedan beskrivs händelseförloppet i treskiktsmodellen vid ett lokalt anrop:
1. Klienten anropar en komponent i mellanskiktet.
2. Komponenten aktiverar ett objekt. Ett COM-objekt föds.
   Objektet tilldelas minnesutrymme som används tills det dör.
3. Objektet vidarebefordrar anropet till databasen.
4. Databasen ger ett resultat till objektet.
5. Objektet förmedlar resultatet till klienten.
6. Objektet dör tills nästa anrop görs av en klient.

Om en annan process anropar samma komponent under det första objektets livstid så föds ett till likadant objekt. När ett anrop släpper ett objekt stannar det kvar för att sedan återanvändas nästa gång ett anrop kräver just detta objekt, s k objektpooling. Detta är ett sätt att minska antalet instanser av ett objekt vilket är viktigt för att uppnå skalbarhet. Helst skall inte resurserna och belastningen öka linjärt med antalet inloggade klienter.

Fördelarna med detta till synes krångliga tillvägagångssätt är att säkerheten ökar, klientprogrammen blir minimala och databasanrop över nätet undviks. Om serverprogram och databas ligger på samma maskin slipper man dessutom nätverkstrafik mellan dessa. I figur 5-5 beskrivs vad de olika skikten innehåller. Klienten presenterar endast gränssnittet samt ytterst lite kod. Transaktionshanteraren hanterar de tunga bitarna av logiska regler och kod. Servern innehåller endast databasen.



Figur 5-5 Konfigurering av treskiktsmodellen
Källa: Jennings R, m fl (1997), sid 27

5.4 Internet

Den snabba utvecklingen av Internet har skapat en ny typ av klient till den traditionella klient-servertekniken; webbapplikationen. Ett anrop fån en webbapplikation sker via en HTTP-server till en databas. Fysiskt sett är detta alltså en treskiktsmodell, dock inte logiskt. En logisk modell kräver ett skikt till; en transaktionshanterare.

Dessa applikationer håller på att ersätta de mer traditionella klient-serverapplikationerna som har varit dominerande i många år. Eftersom inget klientprogram (förutom

webbläsaren) behöver köras på klientdatorn är detta ett mycket smidigt sätt att anropa databaser. Inga justeringar i klientprogrammen behövs göras utan allt sköts centralt.

Med användning av Intranet kan ett företag skapa access till sin databas från hela världen genom en vanlig webbläsare. Är koden skriven i ASP blir det ännu enklare, då spelar det ingen roll vilken webbläsare som används. Vid koppling till en databas via Intranet är det mycket lämpligt att använda sig av en transaktionshanterare som sköter säkerheten i in- och utflödet av information.

5.5 ODBC - Open Database Connectivity

ODBC är en sorts middleware som ligger mellan klientapplikationen och databassystemet. I Windows ligger ODBC-administratören under Kontrollpanelen (Control Panel). ODBC används för att olika applikationsprogram skall kunna få tillgång till data från olika databaser eller andra register t ex textfiler. När en ODBC-datakälla skall skapas namnges denna med ett DSN (Data Source Name), sedan anges vilken sorts datakälla det är, var den finns samt vilken databas eller fil som det gäller. Det är även viktigt att skriva in eventuellt användarnamn (UID - User ID) och lösenord (PWD - Password) till databasen när datakällan konfigureras. I mina kommande exempel har jag använt mig av "sa" som användarnamn (UID) och lösenordet (PWD) har jag lämnat tomt. Detta är de s k defaultinställningarna för UID och PWD i många databaser. När sedan ett applikationsprogram vill använda sig av ODBC:n anges DSN, UID och PWD. Vid behov modifierar ODBC:n applikationens SQL-frågor (SQL-queries) till en form som kan förstås av datakällan. För att anropa ODBC-datakällor från Internet är det mycket lämpligt att använda sig av ActiveX Data Objects (ADO) i ASP-koden. ADO erbjuder ett enkelt och effektivt sätt att förfoga över ODBC datakällor. ADO är designad för att vara snabb och enkel att använda. I kapitel 5.7 har jag inkluderat kodexempel med ADO-kopplingar.

5.6 Kodexempel - Visual Basic

När jag skulle göra kopplingar till databaser valde jag mellan att använda DAO (Data Access Object) och RDO (Remote Data Object). Valet var inte särkilt svårt då det visade sig att DAO inte var lämpligt att använda tillsammans med MTS.[9] Vid kopplingar via DAO där felaktig data eller inte tillräkligt med data förmedlas, visas en dialogruta där användaren måste fylla i rätt information. Detta kan innebära att applikationen hänger sig vid användning med MTS.

I följande kodexempel kommer jag att visa hur kopplingar från ett program skrivet i VB kommunicerar med databasen via ODBC enligt tvåskiktsmodellen. Då jag har använt mig av kopplingar med RDO är "Microsoft Remote Data Object 2.0" satt som referens.

[9] Microsoft Corporation, MSDN, April 1998

Följande kod (kodexempel 5-1) anropar en SQL-databas "BeckwithHR". Alla poster från tabellen "Department" selekteras ut. Därefter förs alla posters "Description" in i en listbox (lstDepartment).

```
Private Sub cmdListDepRDO_Click()
Dim rdoEv As rdoEnvironment
Dim rdoCn As rdoConnection
Dim rdoRs As rdoResultset
Dim SQLSel As String
SQLSel = "Select * From Department"
Set rdoEV = rdoEngine.rdoEnvironments(0)
'Öppnar databasen
Set rdoCn = rdoEv.OpenConnection("BeckwithHR", , , "UID=sa;PWD=")
'Lägger in hela resultatet från frågan i rdoRs
Set rdoRs = rdoCn.OpenResultset(SQLSel, rdOpenKeyset)
'Den följande loopen skriver in hela kolumnen "Description" från rdoRs
'i en listbox vars namn är lstDepartment
Do Until rdoRs.EOF
  lstDepartment.AddItem rdoRs!Description
  rdoRs.MoveNext
Loop
rdoRs.Close
rdoCn.Close
End Sub
```

Kodexempel 5-1 Anrop av en SQL-databas för att lista en kolumn från en tabell.

I kodexempel 5-2 adderas den text som finns i textboxarna txtAddDes och txtAddDepID till tabellen "Department" som "Description" samt "DepartmentID". Koden är liknande föregående kodexempel. Skillnaden är att inget rdoResultset (liknande recordset) skapas utan frågan körs direkt.

```
Private Sub cmdAddDepRDO_Click()
Dim rdoEv As rdoEnvironment
Dim rdoCn As rdoConnection
Dim SQLIns As String, strDes As String, strDepID As String
'Hämtar strängvariabler från textboxarna txtAddDes och txtAddDepID
strDes = txtAddDes
strDepID = txtAddDepID
'Frågan konstrueras med hjälp av strängarna ovan
SQLIns = "Insert into Department (DepartmentID,Description) values _
( '" & strDepID & "' , '" & strDes & "')"
Set rdoEv = rdoEngine.rdoEnvironments(0)
Set rdoCn = rdoEv.OpenConnection("BeckwithHR", , , "UID=sa;PWD=")
'Insättning av posten sker genom att frågan körs
rdoCn.Execute (SQLIns)
rdoCn.Close
End Sub
```

Kodexempel 5-2 Addering av en post i SQL-databas

5.7 Kodexempel - ASP

Vilket jag tidigare nämnde, används ADO vid kopplingar till databaser i ASP-kod. I kommande två exempel visas kopplingar med datakällan "DSN=BeckwithHR". Observera att koden är statiskt skriven för att förenkla förståelsen hos läsaren.

Kodexempel 5.3 är mycket flexibelt eftersom koden kan lista vilken tabell som helst från vilken databas som helst. Det enda som behövs bytas är det som är skrivet i kursiv stil.

```
<html> <head><title>ASP_Select</title></head><body >
<%'En connection skapas med hjälp av ADO
Set DB = Server.CreateObject("ADODB.Connection")
'Databasen öppnas med ADO
DB.Open "dsn=BeckwithHR;uid=sa;pwd="
'Ett recordset skapas
Set RS = DB.Execute("Select * from Department")%>
<P><table border =1>
<tr>
<%'Antalet kolumener räknas och tabellen får lika många kolumner som recordsetet innehåller.
'Namnet på kolumnerna skrivs i fetstil i tabellens första rad.
For i = 0 to RS.Fields.Count - 1 %>
<td><b><%=RS(i).Name %></b> </td>
<% Next %>
</tr>
<% ' Varje post i RS skrivs in i en tabellrad
    Do While not RS.EOF %>
<tr>
  <% For i = 0 to RS.fields.count -1 %>
       <td valign=top><%=RS(i)%> </td>
  <% Next %>
</tr>
<%RS.MoveNext
Loop
RS.Close%>
</table></body></html>
```

Kodexempel 5-3 Använda ADO med ASP för att lista valfri tabell från en databas i webbläsaren

Kodexempel 5-4 visar hur en post adderas med hjälp av ADO. Vanligtvis hämtas strängarna från textboxar och ASP-sidan anropas när ett kommando ges. I detta fallet adderas posten till databasen så fort som sidan anropas.

```
<html> <head><title>ASP_Add</title></head><body >
<%strDepID = "Admin"
strDes = "Administration"
strSQL= "Insert into Department (DepartmentID,Description) values _
( '" & strDepID & "' , '" & strDes & "')"
'En connection skapas och databasen öppnas
Set DB = Server.CreateObject("ADODB.Connection")
DB.Open "dsn=BeckwithHR;uid=sa;pwd="
'Frågan körs vilket innebär att en post läggs till samt en text skrivs ut i webbläsaren att posten
'har lagts till.
DB.Execute(strSQL)
response.write("Post added!")
end if%>
</body></html>
```

Kodexempel 5-4 Använda ADO och ASP för att addera en post till en databas.

6 Objekt

6.1 Allmänt

Objektorienterad programmering (OOP) är idag mycket populärt. Den största fördelen med denna typ av programmering är återanvändning av objekt. För att använda OOP måste objekten stödja inkapsling, arv och polymorfism.

6.2 Klasser i Visual Basic

I VB används klasser. Dessa har filtillägget CLS. Om en klass har skapats i VB kan denna klass användas i alla VB-program som utvecklas. Klasser är väldigt enkla att använda samtidigt som de är mycket kraftfulla.

Vid användandet av en klass undviks att samma kod skrivs på flera ställen i flera olika program. I ett och samma program kan detta kan i och för sig undvikas genom att använda globala procedurer och funktioner. Men skall dessa globala procedurer och funktioner användas i flera program uppstår problem. Om en modifiering skall göras behövs denna göras på alla ställen där proceduren är skriven. Vid användning av en klass behövs ändringen enbart göras på ett ställe.

Klasserna i VB använder sig av inkapsling och polymorfism. Dock kan de inte skapa en ny klass baserad på definitionen av en existerande klass. Detta innebär att VB inte räknas som ett OOP-språk.

Klasserna kan läggas i lämpliga kataloger för att programmen skall få tillgång till dem därifrån. För att sedan använda en befintlig klass importeras denna till programmet. Problemet som i detta fall kan uppstå är att klassen av misstag kan modifieras i programmet och sedan inte går att använda i de andra programmen. Detta löser VB genom att kompilera en CLS-fil till en DLL-fil. På detta sätt skapas ett objekt som inte är modifierbart. Fördelen är att detta objekt även kan anropas av program skrivna i andra språk. Det är mycket viktigt att källkoden sparas. Den behövs om objektet skall skrivas om.

Klasser i VB kan skapas på två sätt.
1. Definiera en Public-variabel.
2. Använda en Property-procedur.

Det första sättet anropas som en vanlig funktion eller procedur. Det är ett mycket enkelt sätt att kommunicera med klassen. Det andra alternativet är krångligare men smidigare. Genom att använda property-procedurerer ges användaren (programutvecklaren) tillgång till gränssnittet av objektets egenskaper. De låter användaren även skriva kod för att kontrollera så att endast giltig data förs över till klassen vilket skyddar klassens funktioner att krascha pga ogiltig data. Man kan även skapa "read-only" egenskaper något som inte är möjligt vid användning av public-variabler.

Det finns tre typer av property-procedurer:
1. "Property let" används för att lagra data i objektet.
2. "Property get" används för att hämta data från objektet.
3. "Property set" är en special form av "let"-proceduren. Den används för att sätta värdet av en objektvariabel.

I bilaga 1 finns två kommenterade kodexempel; en klass som är skapad i VB samt ett formulär med kommandon som anropar denna klass. I dessa exempel använder jag mig av "Property let" och "Property get".

En klass kan inte användas för sig själv. Först måste ett objekt skapas och därefter är det möjligt att använda objektet genom att ändra dess egenskaper och använda dess metoder. Ett objekt kan skapas på två sätt.

1. Att direkt skapa en instans av objektet.
   Dim objPerson as new clsPerson

2. Först deklarera variabeln som ett objekt, därefter sätta den till det nya objektet. Detta kan göras på två sätt, antingen med "early binding" eller "late binding", vilka jag visar nedan. "Early binding" är att rekommendera eftersom man direkt erhåller mycket mer information om vad det är för objekt. "Early binding" fungerar dock inte i VBScript eller ASP, där man måste använda sig av "late binding".
   Dim objPerson as clsPerson **'Early binding**
   Dim objPerson as Object **'Late binding**
   När Set-metoden sedan exekveras skapas objektvariablerna vilket innebär att objektets egenskaper och dess metoder blir tillgängliga för programmet.
   Set objPerson = New clsPerson

Att använda metoderna i objekten är exakt detsamma som att använda VB:s metoder. För att exekvera en metod från den vanliga koden skrivs objektets namn och metodens namn t ex: objPerson.FName = "Anders".

När objektet har används klart skrivs följande kod för att "döda" objektet:
Set objPerson = Nothing

6.3 Kompilera en klass till en komponent

Föra att kompilera en ny klass till en komponent med VB väljer man att skapa ett nytt projekt, ActiveX-DLL. Det går även att ändra ett EXE-projekt till ActiveX-DLL i projektets egenskaper. Innan kompileringen av koden sker är det mycket viktigt att klassens egenskap "Instancing" är satt antingen till "5 - MultiUse" eller "6 - Global MultiUse". Alla referenser i klassen måste också stämma annars kommer inte komponenten att fungera.

För att på ett enkelt sätt illustrera hur klasser kan användas till återkommande anrop har jag i kodexempel 6-1 skrivit en enkel klass (SQLSel). Tanken är att från applikationen överföra databasens namn (strDSN), inloggningsdata (strLogin) och en SQL-selektionsfråga (strSqlSel). Klassen returnerar sedan ett rdoResultset (Opendata).

För att kompilera klassen väljs "File" och "Make SqlSel.dll". Innan VB kompilerar klassen testar kompilatorn att allt är korrekt skrivet. Därefter väljer användaren var DLL-filen skall sparas.

```
'Deklarerar rdo-variabler
Public rdoCn As rdoConnection
Public rdoEv As rdoEnvironment
Public rdoRs As rdoResultset
'Opendata används för att hämta rdoRs från sbObenDb
Public Property Get Opendata() As rdoResultset
Set Opendata = rdoRs
End Property
'sbOpenDb öppnar en databas med hjälp av de strängar skickas från klienten och returnerar ett
'rdoResultset (rdoRS).
Public Sub sbOpenDb(strDSN As String, strLogin As String, strSqlSel As String)
Set rdoEv = rdoEngine.rdoEnvironments(0)
Set rdoCn = rdoEv.OpenConnection(strDBName, , , strLogin)
Set rdoRs = rdoCn.OpenResultset(strSqlSel, rdOpenKeyset)
End Sub
```

Kodexempel 6-1 SQLSel - En klass som selekterar ut svar till ett rdoResultset.

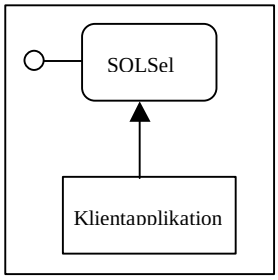
6.4 COM - Component Object Model

När en ny instans av ett objekt skapas, föds en s k COM-komponent. COM-komponenter är begränsade till att kommunicera med binära komponenter på samma dator. Att skriva COM-komponenter har stora fördelar:[10]

- Språkoberoende - COM-komponenter kan skrivas i olika språk och trots det kommunicera mellan varandra.
- Klientflexibilitet - Det är möjligt att använda COM-komponenter till alla applikationer som stöder COM-modellen. Detta gör att klientapplikationer skrivna i C++, Visual Basic eller webbaserade klienter skrivna i ASP kan använda exakt samma komponent.

[10] Jennigs Roger m fl (1997), sid 44ff

I figur 6-1 illustreras ett anrop till en COM-komponent.



Figur 6-1 Beskrivning av en komponent som körs i samma process som klientapplikationen.

Källa: Jennings R m fl (1997), sid 287

Vid användning av komponenten som kompilerades i kodexempel 6-1, måste denna sättas som referens i VB:s "Project-meny". I "References" listar VB alla tillgängliga komponenter. Observera att det inte är på DLL-filen som listas utan vad komponenten är döpt till i dess egenskaper. Anropet till komponenten i kodexempel 6-1 beskrivs i kodexempel 6-2.

Kodexempel 6-1 och 6-2 utgör tillsammans samma funktion som kodexempel 5-2. I kodexempel 6-2 skapas ett nytt objekt (objSQL) av SQLSel (klassen som kompilerades i kodexempel 6-1).

```
Private Sub cmdDepRDO_cls_Click()
'Objektet deklareras
Dim objSQL As SQLSel
'Ett rdoResultset deklareras för att kunna ta emot information från objektet.
Dim rdoRs As rdoResultset
Dim strTable As String, strDSN As String, strLogin As String
strTable = "Department"
strDSN = "DSN=BeckwithHR"
strLogin = "UID=sa;PWD="
strSQLsel = "Select * from " & strTable
'Objektets egenskaper och dess metoder blir tillgängliga för programmet.
Set objSQL = New SQLSel
'SbOpenDb anropas i objektet och tre strängvariabler skickas.
objSQL.OpenDb strDSN, strLogin, strSQLsel
'Anrop av Opendata vilket resulterar i ett rdoResultset från sbOpenDb.
Set rdoRs = objSQL.Opendata
'Objektet "dödas".
Set objSQL = Nothing
rdoRs.MoveFirst
Do Until rdoRs.EOF
   lstDepartment.AddItem rdoRs!Description
   rdoRs.MoveNext
Loop
rdoRs.Close
End Sub
```

Kodexempel 6-2 Subrutin som använder sig av objektet SQLSel.

6.5 DCOM - Distributed Component Object Model

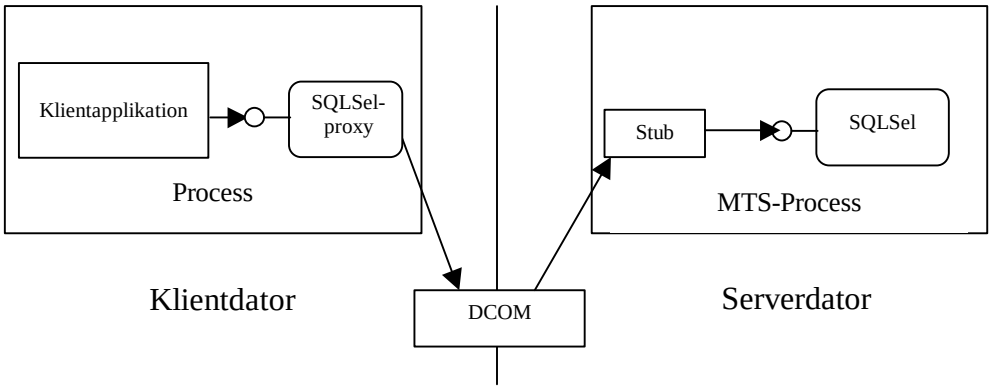
Den stora skillnaden mellan COM och DCOM är att DCOM kan kommunicera med komponenter på andra datorer. DCOM är helt enkelt COM med ett nätverksprotokoll. DCOM har ansvaret att administrera kommunikationen mellan t ex en komponent som körs på servern och dess klient som körs på en annan dator.

För att kunna hantera DCOM med VB måste Visual Basic Enterprise Edition vara installerad. Om Windows NT används som operativsystem behöver det inte konfigureras. Däremot måste DCOM installeras om Windows 95 används som operativsystem.

Nedan har jag beskrivit hur anropet i kodexempel 6-2 fungerar med DCOM.[11] Detta illustreras grafiskt i figur 6-2:

1. Klienten försöker skapa en instans av objektet SQLSel: **Set ObjSQL as NEW SQLSel**
2. COM undersöker Registry för att se om SQLSel finns där. Om objektet körs på en annan dator delegerar COM ansvaret till DCOM.
3. DCOM kommunicerar med server datorn för att undersöka dess Registry och användarnas säkerhetsrättigheter.
4. DCOM skapar både SQLSel-objektet på servern och en s k stub för att ta emot förfrågningar.
5. DCOM skapar ett proxyobjekt i VB:s klientprocess för SQLSel-objektet. Proxyn får också en referens i retur från DCOM.
6. VB-klienten anropar SQLSel-metoden. VB tror att anropet sker direkt till objektet men i själva verket sker anropet via proxyobjektet.
7. Proxyn paketerar de parametrar som är skickade till SQLSel för att förbereda en nätverkstransaktion.
8. DCOM överför parameterpaketet från klienten till servern.
9. DCOM ger parameterpaket till stubben.
10. Stubben packar upp parametrarna och kallar på OpenDB()-metoden i SQLSel-objektet.
11. SQLSel returnerar resultatet av anropet till stubben.
12. Resultaten paketeras av stubben och ger dessa till DCOM.
13. DCOM sänder över resultaten.
14. Resultaten från DCOM tas emot av proxyn.
15. Resultaten packas upp av proxyn och returneras till VB-klienten.
16. Stubben packar upp parametrarna och kallar på Opendata()-metoden i SQLSel-objektet. Punkterna 11-15 körs igen med Opendata()-metoden.
17. Objektet dör genom: Set obJSQL = Nothing

[11] Jennings Roger m fl (1997), sid 289ff



Figur 6-2 DCOM som använder sig av proxy och stub
Källa: Jennings R m fl (1997), sid 289

7 MMC - Microsoft Management Console

7.1 Allmänt

Microsoft Management Console (MMC) är ett verktyg som erbjuder ett gemensamt gränssnitt för administrativa program över nätverket.[12] MMC är mycket lik Utforskaren (Explorer) och ingår i Microsoft Backoffice. MMC administrerar inte själv programmen utan använder sig av s k snap-ins. Detta innebär att MMC körs igång när ett program som använder sig av MMC startas. Inom kort släpps Microsoft SQL-Server 7.0, som kommer att ligga under MMC.

7.2 IIS - Internet Information Server

Microsoft Internet Information Server administreras från MMC. I IIS administreras de kataloger och de filer som är tillgängliga via Internet. Här anges egenskaper för dessa kataloger och filer, vilken katalog som kan innehålla ASP-sidor, vem som har tillgång till filerna i katalogen mm. Dessutom kan lösenord sättas för att begränsa åtkomsten till vissa kataloger.

7.3 *MTS - Microsoft Transaction Server*

Liksom IIS administreras MTS från MMC. MTS är en mycket kraftfull transaktionshanterare som gör det enklare att utveckla och underhålla högprestanda, skalbara klient-server- samt Internet- och Intranetapplikationer. MTS ligger i treskiktsmodellens mellanskikt, d v s mellan klienten och databasen. MTS kontrollerar att komponenter kommer rätt och att databastransaktioner fullföljs. MTS ger även användaren möjlighet att studera vad som händer när dennes program körs; Vilka objekt skapas? Hur ser deras livscykel ut?

I kommande kapitel ger jag en mer noggrann beskrivning av MTS. Jag kommer dock inte att beskriva allt vad MTS tillhandahåller, eftersom produkten är alldeles för omfattande för att göra detta.

[12] MMC-hjälpprogram

8 Vad är MTS?

8.1 Vad som behövs...

För att kunna använda MTS krävs att datorn är utrustad med följande program:

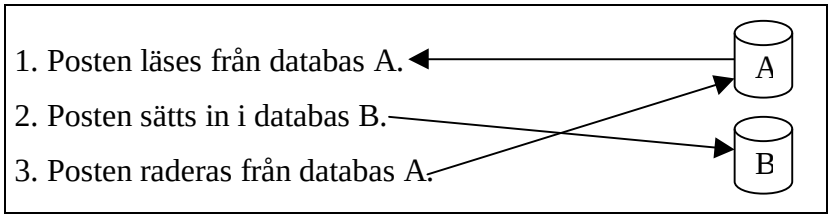
- Windows NT Workstation 4.0+. Windows NT Server+ eller Windows 95+.
- Microsoft SQL Server 6.5+, Oracle eller någon annan av de databaser som stöds av MTS, dock ej Microsoft Access.
- Någon av följande programutvecklingsprodukter; Visual Basic 5.0, Visual C++ 5.0 eller Visual J++ 1.1.
- Microsoft Internet Information Server 4.x (krävs om man vill använda sig av ASP).

Om man nöjer sig med att utveckla kod i ASP kan en enkel textbehandlare, t ex Anteckningar (Notepad) användas istället för någon av programutvecklingsprodukterna.

8.2 Transaktioner och dess problem

Som jag tidigare har beskrivit innebär det stora problem vid programmering i en s k tvåskiktsmodell. En hel del kod måste skrivas för att kontrollera enbart säkerheten. Dessutom måste kod t ex skrivas för att hantera problem som uppkommer när fler än en användare vill göra uppdatering av samma post. Detta kan lösas genom att använda treskiktsmodellen. I det här fallet har MTS som uppgift att ligga som transaktionshanterare i skiktet mellan applikationen/applikationerna och databasen/databaserna.

Ett praktiskt exempel - Att flytta pengar från ett konto i databas A till ett annat konto i databas B. Detta sker i de tre faser vilket åskådliggörs i figur 8-1. Denna relativt enkla transaktion kan skapa oerhörda problem. Två av scenarierna som kan inträffa beskrivs nedan.



Figur 8-1 Exempel på transaktion mellan databaser
Källa: Jennings R, m fl (1997)

Scenario 1 är att om fas 1 och fas 2 utförs utan problem men det därefter uppstår problem vilket leder till att fas 3 inte genomförs. Detta innebär att pengarna aldrig tas bort från kontot i databas A, men sätts in på kontot i databas B.

Scenario 2 är att det efter genomförandet av fas 1 uppstår problem med fas 2 men att fas 3 exekveras utan besvär. Detta innebär att pengarna inte sätts in på kontot i databas B men tas bort från kontot i databas A och pengarna försvinner.

Dessa typer av problem löses av MTS som kontrollerar att alla kommandon utförs utan problem. Om problem uppstår, som i ovanstående scenarier, kommer MTS att återställa databaserna till deras utseende innan transaktionerna startade. I distribuerade databaser måste man annars använda en algoritm kallad "Two-Phase Commit". Denna försäkrar att en transaktion mellan eller i olika databassystem genomförs. Om transaktionen misslyckas skall algoritmen återställa databassystemen till hur de såg ut innan transaktionen. Detta kräver en hel del kodning. Vid användning av MTS undviks en stor del av denna kodning.

8.3 Vad MTS tillhandahåller:[13]

- Komponenttransaktioner - MTS kordinerar och visar transaktionsstatusen av komponenter som kör under dess kontroll, då dessa kommunicerar med t ex en databas.

- MTS administrerar skapandet, underhållet och förstörandet av COM-objekt samt trådarna som exekveras inuti dessa.

- *Resursfördelning* - Genom att skapa en pool av resurser och fördela dessa mellan ett flertal klienter optimerar MTS serverresurserna. MTS hanterar trådningen utan att den som skriver koden behöver oroa sig för de problem som kan uppkomma med trådningen. När komponenter utvecklas med MTS skrivs dessa som om de endast består av en tråd vardera. MTS koordinerar dessutom kommunikationen mellan komponenterna och databasen med hjälp företablerade kopplingar, vilket innebär att hundratalet komponenter kan få tillgång till databasen med endast ett dussin kopplingar.

- *Säkerhet* - MTS har en flexibel säkerhetsstruktur vilket är en av MTS främsta egenskaper. Enbart säkerhetsaspekten kan ibland vara skäl nog för att sätta sina databaser under MTS.[14] Istället för att ha en tabell i databasen med användaridentiteter, registreras användarna under paketet i MTS. Där är det möjligt att bestämma vilka användare som har rätt att använda olika komponenter. För att undvika att gå in och lägga till nya användare och deras rättigheter i MTS kan kod skrivas som sedan exekveras i klientapplikationen. Detta kan vara lämpligt att använda sig av när administratören enkelt vill kunna lägga till användare i applikationsprogrammet. Säkerheten i MTS är uppdelad i två delar, deklarativ säkerhet och programmatisk säkerhet. Vid användning av s k deklarativ säkerhet kontrollerar MTS användaren enbart vid första anropet till MTS. Detta förenklar

[13] Jennings m fl (1997), sid 6ff
[14] Jennings m fl (1997), sid 14

systemadministrationen avsevärt. Den deklarativa säkerheten är uppdelad i nivåer, komponentnivå och gränssnittsnivå. På komponentnivån är användaren antingen given eller nekad access till hela komponenten. På gränssnittsnivån kan säkerheten mot användaren kontrolleras på metodnivå. Vid programmatisk säkerhet är det möjligt för komponenten att fråga MTS vem dess klient är. Komponenten kan ändra sitt beteende beroende på vem användaren är. MTS använder sig av roller för dess användare. MTS ger inte svar vem den enskilda klienten är utan vilken roll den tillhör.

- *Administration* - MTS låter användaren (administratören) enkelt ändra konfigureringsinställningar av klient-serversystemet under och efter utvecklingen utan att ändra någon kod. Under MTS kan komponenternas egenskaper enkelt konfigureras. Med de olika transaktionsverktygen är det mycket enkelt att kontrollera om transaktionerna har lyckats eller ej.

- MTS hanterar komponentutveckling och installation av komponenter på ett mycket smidigt sätt. Den vanligtvis arbetsamma processen att registrera DCOM-komponenter sköts av MTS.

8.4 Praktiskt exempel - SAS

SAS använder sedan i juli 1997 MTS för att ge sina kunder möjlighet att boka sin flygresa via Internet.[15] Den enda restriktionen är att kunden måste vara registrerad användare. Kunden får ett "Travel Pass" (ett plastkort i kreditkortsformat) och en personlig kod för inloggning. De kunder som använder sig av denna bokningsmetod behöver inga biljetter. Kortet och koden gäller som resebevis. Detta har reducerat SAS:s administrativa kostnader rejält för de kunder som använder sig av "Travel Pass". Systemet är uppbyggt kring MTS med komponenter byggda i Visual Basic och C++. Enligt ansvariga på SAS fungerar MTS mycket bra som ett lager ovanpå COM.

8.5 MTS och DCOM

För att kunna köra de komponenter som ligger under MTS från andra datorer måste information om dessa komponenter registreras i klientdatorns Registry.[16] COM på klientens maskin använder Registry för att avgöra om komponenten skall köras på en annan maskin (DCOM). Följande information registreras i Registry:

- Komponenternas CLSID:s.
- Den andra datorns namn för varje komponent.
- Referenser till typbibliotek för varje komponent och dess metoder.

[15] Lidfeldt, Torun, *Datateknik*, nr 17, 1997
[16] Jennings Roger m fl (1997), sid 291ff

Detta kan göras på två sätt:

1. Att skapa en EXE-fil på klientens dator med hjälp av MTS. Detta är det enklaste sättet att konfigurera klientens dator. Detta beskrivs i kapitel 8.6 samt i bilaga 2.
2. Att manuellt kopiera filer från serverdatorns Registry och registrera dessa på klienternas datorer. Detta är dock ett relativt jobbigt förfarande och rekommenderas inte.

8.6 Utveckling med MTS[17]

MTS arbetar med s k paket. De installerade paketen visas i Transaction Server Explorer (TSE). Dessa paket kan enkelt skapas i MTS vilket beskrivs i bilaga 2.

För att skapa komponenter till MTS skapas DLL-filer. Dessa filer kan skapas i VB 5.0. Efter skapandet av en DLL-fil skall denna installeras i önskat paket under MTS. När ett paket är färdigskapat skall det exporteras. Detta måste göras för att andra datorer skall kunna anropa paketets komponenter och för att dra fördel av MTS egenskaper.

Vid exporten av paketet skapar MTS tre filer på serverdatorn; en paketfil (PAK - innehåller information om paketet), en komponentfil (DLL - denna registreras i serverdatorns Registry), och en klientfil (EXE). För att kunna anropa paketet från en klientdator måste EXE-filen från serverdatorn köras lokalt på klientdatorn. Denna fil innehåller all information om komponenterna på servern för att man skall kunna köra DCOM. När denna fil körs registrerar klientdatorn DCOM-komponenten i Registry och referensen till denna komponent blir synlig i VB. För att ta bort denna DCOM-komponent från klientdatorns Registry används Lägg till/Ta bort program (Add/Remove programs) under Kontrollpanelen.

Om komponenterna ändras måste hela proceduren upprepas. Detta är ett mycket arbetsamt förfarande. Jag rekommenderar andra utvecklare att använda VB ihop med MTS på den dator där MTS är installerad. Detta sparar mycket tid genom att utvecklaren kan sitta vid en och samma dator hela tiden. Med en tilläggskomponent (add-in) till VB uppdateras komponenten när den ändras i VB. Därefter uppdaterar man komponenten i MTS.

[17] Dokumenterad beskrivning om att arbeta med MTS återfinns i bilaga 2.

8.7 SetComplete - SetAbort

MTS utför respektive avbryter en komponents arbete med kommandona SetComplete och SetAbort. Detta är en mycket viktig funktion i MTS. Om vi refererar till transaktionsexemplet av pengar i figur 8-1, kan programmeraren använda sig av en felhanterare som körs när något går snett. I kodexempel 8-1 görs en mycket enkel beskrivning hur SetAbort och SetComplete används.

```
Public Sub test()
Dim objContext as ObjectContext
Set objContext sa CreateObjectContext
On Error Goto ErrHandler
'kod
objContext.SetComplete
exit sub
ErrHandler:
objContext.SetAbort
End Sub
```

Kodexempel 8-1 SetAbort och SetComplete

9 Implementation

9.1 Allmänt

Under arbetets gång har jag studerat en bok av Roger Jennings, Microsoft Transaction Server 2.0. Det är den enda boken som för tillfället finns om MTS. Till denna bok bifogades en CD-skiva med fem databaser och diverse kodexempel. Dessa databaser installerade jag under SQL-server. Testerna mot databaserna, som jag ämnade köra med de olika kodexemplen, havererade totalt. Något var fel i koden och jag fick börja om från början. Dessa databaser har jag dock använt i mina tester.

I MTS kan mycket kodkrävande komponenter användas. En komponent kan t ex plocka fram stora mängder data från olika databaser. Komponenten selekterar sedan datan så att den till slut innehåller mycket liten mängd data men med stort informationsvärde för klienten. Denna mängd data returneras till klienten. Om klienten och serverdatorn (där MTS och databasen är installerad) befinner sig på olika kontinenter har dataflödet lång väg att gå. För att undvika att stora mängder data transporteras över långa avstånd kan komponenter som den ovan beskrivna användas. Detta är en av MTS allra viktigaste användningsområden.

9.2 Komponent/Objekt som anropas från VB och ASP

I kodexempel 9-1 finns kod för klassen clsMTS_Beck1 som jag har döpt till mtstest och kompilerat till mtstest.dll. Denna klass är i det närmaste identisk med koden i SQLSel som finns i kodexempel 6-1. De enda skillnaderna är att en procedur sbSQL är tillagd samt att anropet till databasen är något annorlunda. Detta objekt (mtstest.dll) kan sedan anropas genom MTS från både VB och ASP. Objektet måste då ingå i ett MTS-paket. Opendata används för att generera ett rdoResultset från sbOpenDb medan sbSQL kör en SQL-fråga (strSQL), t ex en insättningsfråga eller en borttagningsfråga. Denna kod kan nu användas av en mängd applikationer för att kommunicera med databaser.

```
'Deklarerar de rdo-variabler som behövs
Public rdoCN As rdoConnection
Public rdoRS As rdoResultset
'Opendata används för att hämta rdoRs från sbObenDb
Public Property Get Opendata() As rdoResultset
    Set Opendata = rdoRS
End Property
'sbOpenDb öppnar en databas med hjälp av de strängar som skickas från klienten och returnerar
'ett rdoResultset (rdoRS). Jag har här använt mig av ett något annorlunda förarande än
'kodexempel 6-1 eftersom jag inte har deklarerat rdoEv as rdoEnvironment.
'På detta sätt slipper jag dela upp strLogin-variabeln.
Public Sub sbOpenDb(strLogin As String, strSQLsel As String)
   Set rdoCN = rdoEngine.rdoEnvironments(0).OpenConnection("", rdDriverNoPrompt, False, strLogin)
   Set rdoRS = rdoCN.OpenResultset(strSQLsel, Options:=rdExecDirect)
End Sub
```

```
'Insättning eller borttagning av poster i en databas med hjälp av en fråga - strSQL
Public Sub sbSQL(strSQL As String, strDSN As String, strLogin As String)
'Databasen öppnas
Set rdoEV = rdoEngine.rdoEnvironments(0)
Set rdoCN = rdoEV.OpenConnection(strDSN, rdDriverNoPrompt, False, strLogin)
'Frågan (strSQL) körs
rdoCN.Execute strSQL
rdoCN.Close
End Sub
```

Kodexempel 9-1 Klassen clsMTS_Beck1 innan kompilering till MTStest.dll

9.3 Anrop av objekt från VB

I kodexempel 9-2 anropas clsMTS_Beck1 som returnerar ett rdoResultset. Därefter sätts kolumnen "Description" in i listboxen "lstDepartment". Resultatet av detta exempel är identiskt med kodexempel 5-1 och kodexempel 6-2. Den enda skillnaden är att kodexempel 9-2 och kodexempel 6-2 anropar ett objekt som utför viss del av arbetet. Kodexempel 9-2, 9-3, 9-4 anropar ett objekt som är kompilerat till en DLL-fil.

```
Private Sub cmdDepRDO_Click()
'Objektet deklareras
Dim obj As clsMTS_Beck1
'Ett rdoResultset måste deklareras för att kunna ta emot information från objektet
Dim rdoRs As rdoResultset
Dim strTable As String, strLogin As String, strSQLsel As String
strTable = "Department"
strLogin = "DSN=BeckwithHR;UID=sa;PWD="
strSQLsel = "Select * from " & strTable
'Objektets egenskaper och dess metoder blir tillgängliga för programmet.
Set obj = New clsMTS_Beck1
'SbOpenDb anropas i objektet och två strängvariabler skickas
obj.sbOpenDb strLogin, strSQLsel
'Anrop av Opendata vilket resulterar i ett rdoResultset från sbOpenDb
Set rdoRs = obj.Opendata
'Objektet "dödas"
Set obj = Nothing
'Fältet "Description" från varje post sätts in i en listbox "lstDepartment"
Do Until rdoRs.EOF
   lstDepartment.AddItem rdoRs!Description
   rdoRs.MoveNext
Loop
rdoRs.Close
End Sub
```

Kodexempel 9-2 Anrop från en VB-applikation till clsMTS_Beck1

Resultatet av kodexempel 9-3 är identiskt med det i kodexempel 5-2. Skillnaden är att kodexempel 9-3 använder sig av ett objekt för att lägga till posten i databasen. I kodexempel 9-4 bortages en post. Det enda som skiljer från exempel 9-3 är utformningen av frågan (strSQL samt strSQLDel).

```
Private Sub cmdAdd_Click()
'Objektet deklareras
Dim obj As clsMTS_Beck1
Dim SQLIns As String, strDes As String, strDepID As String, strTable As String
Dim strDSN as String, strLogin As String
strDSN = "BeckwithHR"
strLogin = "UID=sa;PWD="
strDepID = "Admin"
strDes = "Administration"
strTable = "Department"
SQLIns = "Insert into " & strTable & " (DepartmentID,Description) values _
( '" & strDepID & "' , '" & strDes & "')"
'Objektets egenskaper och dess metoder blir tillgängliga för programmet.
Set obj = New clsMTS_Beck1
'SQLIns skickas till sbSQL tillsammans med DSN-namn och logininformation
obj.sbSQL, SQLIns, strDSN, strLogin
'Posten är adderad och objektet "dödas"
Set obj = Nothing
End Sub
```

Kodexempel 9-3 Tillägg av post genom anrop av sbSQL

```
Private Sub cmdDelete_Click()
Dim strSQLDel As String, strTable As String
Dim strLogin As String, strDSN As String
strTable = "Department"
strDSN = "BeckwithHR"
strLogin = "UID=sa;PWD="
strSQLDel = "DELETE FROM " & strTable & " WHERE Description = '" & lstDepartment.Text & "'"
Set obj = New clsMTS_Beck1
obj.sbSQL strSQLDel, strDSN, strLogin
Set obj = Nothing
End Sub
```

Kodexempel 9-4 Borttagning av post genom anrop av sbSQL

9.4 Anrop av objektet från ASP

I de två nedanstående exemplen anropas objektet från ASP. Kodexempel 9-5 har samma funktion som kodexempel 5-3, d v s en hel tabell från en databas listas i webbläsaren. Koden för att anropa objektet är mycket lik den kod som används i kodexempel 9-2. Kommentarer som är gjorda i kodexempel 5-3 är inte gjorda nedan. I kodexempel 9-6 adderas en post till en databas. Denna kod är mycket lik kodexempel 9-3. För att ta bort en post kan strSQL ersättas med strSQL från kodexempel 9-4.

```
<html><head><title>SelMTS</title></head><body>
<% strSQL = "Department"
strSQL= "Select * from " & strSQL
strDSN = "BeckwithHR"
'Inloggningssträngen sätts samman
strLogin = "dsn=" & strDSN & ";uid=sa;pwd="
Set obj = Server.CreateObject("MTStest.clsMTS_Beck1")%>
<table border =1>
<%'sbOpenDb anropas. Variablerna konverteras till strängar genom kommandot Cstr
obj.sbOpenDb Cstr(strLogin), Cstr(strSQL)
'Resultatet returneras i ett rdoResultset
Set rdoRs = obj.Opendata%>
<tr>
<%For intCount = 0 to rdoRs.rdoColumns.Count - 1 %>
<td><b><%=rdoRs(intCount).Name %></b> </td>
<% Next %>
</tr><tr>
<%Do While not rdoRs.EOF
For intCount = 0 to rdoRs.rdoColumns.count -1 %>
<td>
<%Response.Write(rdoRs(intCount))%>
</td>
<%next%>
</tr>
<%rdoRs.MoveNext%>
<%Loop
rdoRs.Close
end if%>
</table></body></html>
```

Kodexempel 9-5 **Använda ASP och en MTS-komponent för att köra en SQL-fråga mot en databas.**

```
<html><head><title>addMTS</title></head><body>
<%strDepID = "Admin"
strDes = "Administration
set obj = server.CreateObject("MTStest.clsMTS_Beck1")
strSQL="Insert into Department (DepartmentID,Description) values _
 ( '" & strDepID & "' , '" & strDes & "')"
strDSN="BeckwithHR"
strLogin="UID=sa;PWD="
'Objektet anropas och posten adderas
obj.sbSQL cstr(strSQL), cstr(strDSN), cstr(strLogin)
response.write("Post added")
end if%>
</body></html>
```

Kodexempel 9-6 **Addering av en post till en databas med hjälp av MTS**

10 Slutsatser

Detta arbete behandlar till stor del komponenter/objekt. Användningen av komponenter effektiviserar kodskrivandet i allra högsta grad. Kod som är likadan och används i många olika applikationer kan skrivas i VB 5.0 och sedan kompileras till en komponent. När koden skall ändras räcker det att ändringen sker på ett ställe. Denna komponent kan sedan anropas från alla dessa applikationer. COM/DCOM är komponenter som skapas när en ny instans av en komponent skapas. Skillnaden mellan COM/DCOM är att DCOM kan kommunicera med andra komponenter på andra datorer medan COM endast kan kommunicera med komponenter på den lokala datorn. Med VB är det mycket enkelt att anropa komponenter och göra kopplingar till databaser via ODBC vilket jag har visat i ett antal kodexempel. Det är svårt att ge en svar på specifika kodningsmetoder här. Istället hänvisar jag till mer detaljerade beskrivningar i de olika kodexemplen som finns i rapporten.

Fler och fler klient-serversystem kommer att utvecklas enligt treskiktsmodellen, d v s med ett skikt mellan klienten och servern. Den största utvecklingen tror jag kommer ske med webbläsare som klienter. ASP, som ingår i Visual Basic-familjen, har en stor potential att bli ett av de nya stora kodspråken. Det är mycket lätt att arbeta med och enkelt att använda tillsammans med ADO för att göra kopplingar mot databaser. Microsoft Transaction Server (MTS) har möjlighet att kunna utvecklas till ett oerhört betydelsefullt program, bl a eftersom det krävs en hel del säkerhet vid transaktioner över Internet. Dessutom är det mycket lämpligt att använda komponenterna i MTS till att paketera värdefull data från databasen och skicka denna data till klienten. Än så länge är MTS endast i början av sin livscykel vilket innebär att det finns mycket övrigt att önska. Det är t ex omständigt och tidskrävande att uppdatera komponenter om dessa är skapade på en dator som inte har MTS installerad. För en programutvecklare som skall utveckla klient-serverapplikationer med MTS är det oerhört viktigt utvecklaren sitter vid datorn där MTS är installerad.

I stort sett är jag nöjd med vad jag har kommit fram till i detta arbete. Det skulle dock ha varit roligt om jag hade kunnat behandla alla de frågeställningar som jag från början hade. Tyvärr så räckte inte tiden till. Under arbetets gång har jag lärt mig mycket. Mina kunskaper i ASP har förbättrats liksom kunskaper om objekt. Jag har blivit familjär med användandet av MTS. Detta kommer jag ha stor glädje av om, eller rättare sagt när, jag kommer att börja använda MTS i utvecklingen av olika program.

Vissa saker som jag har gjort borde jag undvikit. T ex gick alldeles för mycket tid åt för att köra testexempel från Jennings bok som inte fungerade. Jag borde ha börjat med att skriva egna små programsnuttar från början.

Tyvärr finns det för tillfället bara en bok som behandlar MTS. Under detta året kommer dock fler böcker att ges ut och andra perspektiv ges. Jennings bok är dock mycket bra och jag rekommenderar den varmt.

Käll- och litteraturförteckning

Böcker

Casad Joe & Dalton Wayne (1997), *MCSE Training Guide: Windows NT Server 4*, ISBN 1-56205-768-5, New Riders Publishing, Indianapolis.
Garms, Jason m fl (1996), *Windows NT Server 4 Unleashed*, ISBN 0-672-30933-5, Sams Publishing, Indianapolis.
Jennings, Roger m fl (1997), *Microsoft Transaction Server 2.0*, ISBN 0-672-31130-5, Sams Publishing, Indianapolis.
Nielsen, Morten Strunge (1995), *Client/Server*, ISBN 91-634-1255-1, Liber Utbildning AB, Växjö.
McKelvy, Mike m fl (1997), *Special Edition Using Visual Basic 5*, ISBN 0-7897-0922-8, Que Corporation, Indianapolis.
McKinney, Bruce (1997), *Hardcore Visual Basic*, ISBN 1-57231-422-2, Microsoft Press, Washington.
Solomon, David m fl (1996), *Microsoft SQL Server 6 Unleashed*, ISBN 0-672-30903-3, Sams Publishing, Indianapolis.

Muntliga källor

Broms, Daniel, Canopus AB.
Lindholm, Torgny, Canopus AB.
Segell, Per, Canopus AB.
Östling, Andreas, Canopus AB.
Östlund, Markus, Canopus AB.

Tidningsartiklar

Karlander, Lars, "Treskiktsmodellen är en modell som håller", *Datateknik*, nr 7, 1998.
Lidfeldt, Torun, "Microsoft Transaction Server som utvecklingsverktyg", *Datateknik*, nr 17, 1997.
Sandred, Jan, "NT - Unix: Skillnaderna minskar", *Datateknik*, nr 18, 1997.

Övriga källor

Microsoft Corporation, *Microsoft Transaction Server FAQ: Databases and Transactions*, MSDN - Misrosoft Developer Network , April 1998.
MMC:s och MTS:s egna hjälpprogram.

BILAGA 1

Kodexempel klass

CmdCreatePerson i kodexempel 2 skapar en person i ett objekt genom att anropa clsPerson (kodexempel 1) som använder sig av "property let" för att lagra informationen. För att få fram information om personerna anropas clsPerson med cmdTalk1 eller cmdTalk2. CmdTalk1 anropar talkfunktionen i clsPerson. CmdTalk2 använder sig av "property get" för att få fram information om personen.

```
Option Explicit
'Variabler skapas för att lagra värden om personen
Private mvarFirstName As String
Private mvarLastName As String
Private mvarAge As Integer
'En funktion som returnerar en sträng med information om personen
Public Function Talk() As String
   Talk = mvarFirstName & " " & mvarLastName & " is " & mvarAge & " years old."
End Function
'Åldern läggs till i mvarAge
Public Property Let Age(ByVal vData As Integer)
  mvarAge = vData
End Property
'Ålder hämtas från mvarAge
 Public Property Get Age() As Integer
  Age = mvarAge
 End Property
'Efternamnet läggs till i mvarLasName
Public Property Let LastName(ByVal vData As String)
   mvarLastName = vData
End Property
'Efternamnet hämtas från mvarLasName
 Public Property Get LastName() As String
    LastName = mvarLastName
 End Property
'Förnamnet läggs till i mvarFirstName
Public Property Let FirstName(ByVal vData As String)
   mvarFirstName = vData
End Property
'Förnamnet hämtas från mvarFirstName
 Public Property Get FirstName() As String
    FirstName = mvarFirstName
 End Property
```

Kodexempel 1 Koden i clsPerson som anropas i frmPerson

```
Option Explicit
'Ett objekt deklareras
Dim objPerson As clsPerson
'En collection skapas där data kan lagras
Dim colPeople As New Collection

'Addering av person till ett objekt. Dessutom adderas personen till "colPeople".
Private Sub cmdCreatePerson_Click()
'Objektet "objPerson" skapas
Set objPerson = New clsPerson
'En person skapas genom att hämta värden från textboxar.
objPerson.FirstName = txtFName
objPerson.LastName = txtLName
objPerson.Age = txtAge
'Personen lagras i en collection "colPeople".
colPeople.Add objPerson, txtFName
'Personens förnamn läggs till i en listbox "lstPeople".
lstPeople.AddItem txtFName
txtFName.SetFocus
End Sub

'Utskrift med information om personen i etiketten "lblReturn"
Private Sub cmdTalk1_Click()
'ObjPerson deklareras till den person vars förnamn är markerad i listboxen
Set objPerson = colPeople.Item(lstPeople)
'Funktionen talk returneras från "objPerson" till etiketten lblReturn
lblReturn = objPerson.Talk
End Sub

'Ett alternativ till proceduren ovan.
Private Sub cmdTalk2_Click()
Set objPerson = colPeople.Item(lstPeople)
'Istället för att anropa talk-funktionen hämtas informationen direkt
lblReturn = objPerson.FirstName & " " & objPerson.LastName & " is " & objPerson.Age & " years old."
End Sub

'objPerson och colPeople "dödas"
Private Sub cmdDelete_Click()
Set objPerson = Nothing
Set colPeople = Nothing
End Sub
```

Kodexempel 2 **Koden i frmPerson, ett formulär som anropar clsPerson i kodexempel 1**

BILAGA 2

Skapa ett nytt paket

1. Starta Transaction Server Explorer (TSE).
2. Högerklicka på *Packages installed*. De paket som är installerade visas.
3. Välj *New -> Package*.
4. Välj att skapa ett nytt paket - *Create an empty package*.
5. Ange det nya paketets namn.
6. MTS frågar efter användarnamn. Acceptera *Interactive user* (Nuvarande användare).
7. Ett paket är nu skapat och detta visas i TSE.

Skapa komponenter

1. I VB väljer man att skapa ett nytt ActiveX-dll projekt.
2. Därefter skapas de klasser som behövs. Varje klass som skapas blir en komponent i MTS. Det är mycket viktigt att använda rätt referenser till MTS ifall man skall använda sig av den kod som MTS tillhandahåller.
3. Ange projektets namn i dess egenskaper. Det är detta namn som kommer att identifiera komponenten.
4. När projektet är färdigskrivet, skall koden kompileras till en DLL-fil. Detta görs genom kommandot *Make namn.dll.*

Installera komponenter i MTS

1. Öppna MTS
2. Klicka på projektets namn där komponenterna skall installeras.
3. Högerklicka på *Components*
4. Välj *New -> Component*
5. Välj att installera ny komponent - *Install new component(s)*. Om komponenten/ komponenterna redan är registrerade väljer man *Import component(s) that are already registred.*
6. Välj att lägga till filen som innehåller komponenterna- *Add file*
7. Välj den DLL-fil som skall användas och klicka på OK. Filen samt dess komponenter visas.
8. Slutför installationen - *Finish*.
9. Varje klass som skapades i VB har nu blivit en komponent och de visas nu under *Components*.

Exportera paket från MTS

1. Högerklicka på paketet som skall exporteras.
2. Välj namn på och plats där paketet skall sparas. Det är lämpligt att lägga dessa filer på samma dator där MTS är installerad.
3. Klicka på OK och exporten slutförs.

Installation av MTS-komponenter (DCOM) på klientdatorn

1. Kör EXE-filen, som skapades vid exporten av paketet, på klientdatorn.
2. Kontrollera att komponenten är installerad genom att öppna *Lägg till/Ta bort program* i *Kontrollpanelen*. I detta fönster bör det stå *Remote Application - Name (Remove only)* där *Name* är det namn som paketet har i MTS.
3. Nu kan komponenten användas i VB. Det enda som krävs är att en referens sätts till komponenten.

Förändringar i komponenter

1. Ta bort den installerade DCOM-komponenten i *Lägg till/Ta bort program*.
2. Gå in i källkoden till DLL-filen i VB.
3. Utför ändringarna.
4. Kompilera koden igen.
5. Ta bort komponenten från paketet i MTS.
6. Installera den nya komponenten - *Installera komponenter i MTS.*
7. Exportera paketet till en fil på serverdatorn - *Exportera paket från MTS.*
8. Kör klientfilen som skapas vid exporten från klientdatorn - *Installation av MTS-komponenter (DCOM) på klientdatorn*.

Övriga kommentarer

Den omständiga processen att förändra komponenten kan undvikas om DLL-filen direkt kompileras på serverdatorn där MTS är installerad. I så fall är det möjligt att uppdatera komponenten/komponenterna i MTS genom en add-in i VB.