Nätverksarkitekturer, en jämförelse

(HS-IDA-EA-98-407)

Joakim Christoffersson (a94joach@ida.his.se)

Institutionen för datavetenskap
Högskolan i Skövde, Box 408
S-54128 Skövde, SWEDEN

Examensarbete på det dataekonomiska programmet under vårterminen 1998.

Handledare: Henrik Gustavsson

Nätverksarkitekturer, en jämförelse

Examensrapport inlämnad av Joakim Christoffersson till Högskolan i Skövde, för Kandidatexamen (BSc) vid Institutionen för Datavetenskap.

[98-06-08]

Härmed intygas att allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och att inget material är inkluderat som tidigare använts för erhållande av annan examen.

Signerat: _____

Nätverksarkitekturer, en jämförelse

Joakim Christoffersson(a94joach@ida.his.se)

Key words: COM, ActiveX, DCOM, Client/Server, Intranät

Abstract

This work describes some different network architectures and their pros and cons, compared to each other in order to give an understanding of which architecture best would fit a specific application or system. The four examined architectures, are Client/Server, Intranet in a static manner, Intranet with support of ActiveX and finally DCOM, differ quite a bit regarding age and suitable application areas. Therefore this report contains a list of some specific factors, such as for instance cost and effectivenes and how each of the examined architectures relate to these factors. This will hopefully ease the choice in a system development project so that it is possible, solely with help of the customers requirementsspecification and the listed factors in this report get an indication of what kind of architecture best would fit in that specific situation.

Innehållsförteckning

Sammanfattning

Detta arbete tar upp och beskriver fyra olika nätverksarkitekturer, med deras för och nackdelar gentemot varandra. Detta för att ge en insikt och förståelse till vilken arkitektur som bäst passar ihop med olika applikationer/system, detta för att förenkla och ge signaler på vilken arkitektur olika applikationer bör byggas upp kring i ett systemutvecklings projekt. De undersökta nätverksarkitekturerna varierar väldigt gentemot varandra, gällande ålder och användningsområde. Client/Server som är den första arkitekturen och ligger lite som grund för de övriga är dels komplext att administrera och dels kan även vara svårt för ovana datoranvändare att lära sig, då det finns så många olika funktioner att lära sig i respektive applikation, men erbjuder körning av komplexa applikationer över nätverket. För att få bukt med användarvänligheten och arbetsinsatserna vid administreringen kom det statiska intranätet som ett alternativ. Nackdelen är dock att ett statiskt Intranät saknar möjligheten att klara av komplexa program. Vidareutvecklingen, dynamiskt Intranät möjliggör datahantering på klienten till skillnad mot ett statiskt Intranät och erbjuder samtidigt större möjlighet att köra komplexa program. Nackdelen är dock att en av intranätets grundtanke med plattforms oberoende blir begränsad i och med viss prestanda och programvara krävs. Den sista arkitekturen som granskas är DCOM, vilken är en lösning på distribuerade applikationer och samtidigt en utbyggnad av Microsofts COM. DCOM möjliggör mycket av de fördelar ett dynamiskt Intranät erbjuder (stödjer nämligen även de protokoll som ett Intranät bygger på och möjliggör nyttjande av webläsare) men erbjuder samtidigt genom sin distribuerade karaktär möjlighet att låta applikationer köras där de gör bäst nytta.

Syftet med denna rapport är alltså att i ett systemutvecklings projekt underlätta och möjliggöra ett tidigt val av en lämplig arkitektur som kan fungera tillsammans med de applikationer eller det system som kunder behöver. Detta arbete förenklar detta val genom att lista olika faktorer, som t.ex. kostnad och effektivitet tillsammans med hur de olika arkitekturerna förhåller sig till dessa faktorer, detta så att man redan utifrån en kravspecifikation ska kunna se hur väl kundens önskemål och krav uppfylls av respektive arkitektur.

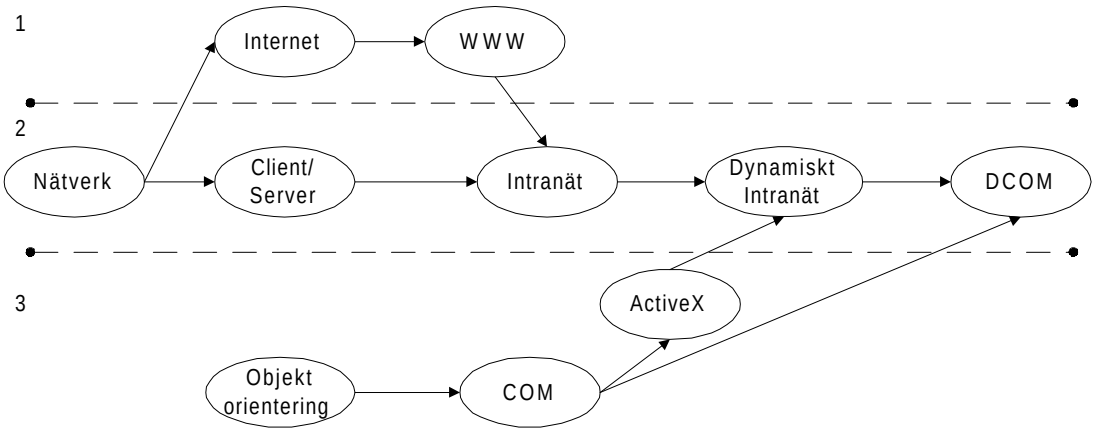
1 Introduktion

Sedan lång tid tillbaka har det funnits olika tankar och idéer om vilken arkitektur som är den bästa att nyttja sig av vid ett systemutvecklingsprojekt. Detta för att så bra som möjligt passa ihop med den applikation eller det system organisationen ska kretsa kring. Förr i tiden stod denna kamp oftast mellan stordator system eller nätverk. Valet av arkitektur gentemot program var då relativt lätt då programmen oftast var kopplade till respektive miljö. Idag dock har många liknande varianter av nätverk vuxit fram och gränsen mellan vilka program som bör köras över vilken form av nätverksarkitektur är idag därmed inte lika solklar.

Ett riktigt arkitekturval är av stor vikt för ett utvecklingsprojekt då förändringar av arkitekturen i ett sent skede blir mycket kostsam och tidsödande. Syftet med detta arbete är att ta fram ett antal kriterier som underlättar ett riktigt arkitekturval redan på tidigt stadium utifrån de krav som ställts på systemet.

Det finns visserligen tidigare jämförelser av olika arkitekturer. Exempelvis tidningen BYTE som jämför Java och ActiveX bl.a. i augusti 1996-, augusti 1997- och september 1997- numrerana. DCOM jämförs mot CORBA bl.a. i artikeln DCOM and CORBA Side by Side, Step by Step, and Layer by Layer, 1997 av P. Emerald, Yennun Huang, Shalini Yanik, Deron Liang, Joanne C. Shih, Chung-Yih Wang och Yi-Min Wang. Dock saknar dessa tidigare jämförelser denna bredd som detta arbete syftar till att resultera i. Dessutom tenderar tidigare jämförelser mer att gå in på tekniska skillnader två lösningar emellan och oftast då även inom samma nisch exempelvis Java kontra ActiveX, DCOM kontra CORBA (som är en annan lösning på distribuerade applikationer) o.s.v. och inte på de krav som gör en arkitektur fördelaktig.

Figuren nedan beskriver hur några olika teknologier, som utvecklats avskiljt ifrån varandra allt eftersom har börjat att vävas samman.



1. Globala nätverks lösningar
2. Lokala nätverks arkitekturer
3. Programvaru utveckling

Figur 1 Beskrivning över hur olika begrepp hör ihop

2 Bakgrund

Detta kapitel är till för att ge en grundläggande förståelse för hur termerna i figuren i introduktionen hänger ihop.

2.1 Nätverk

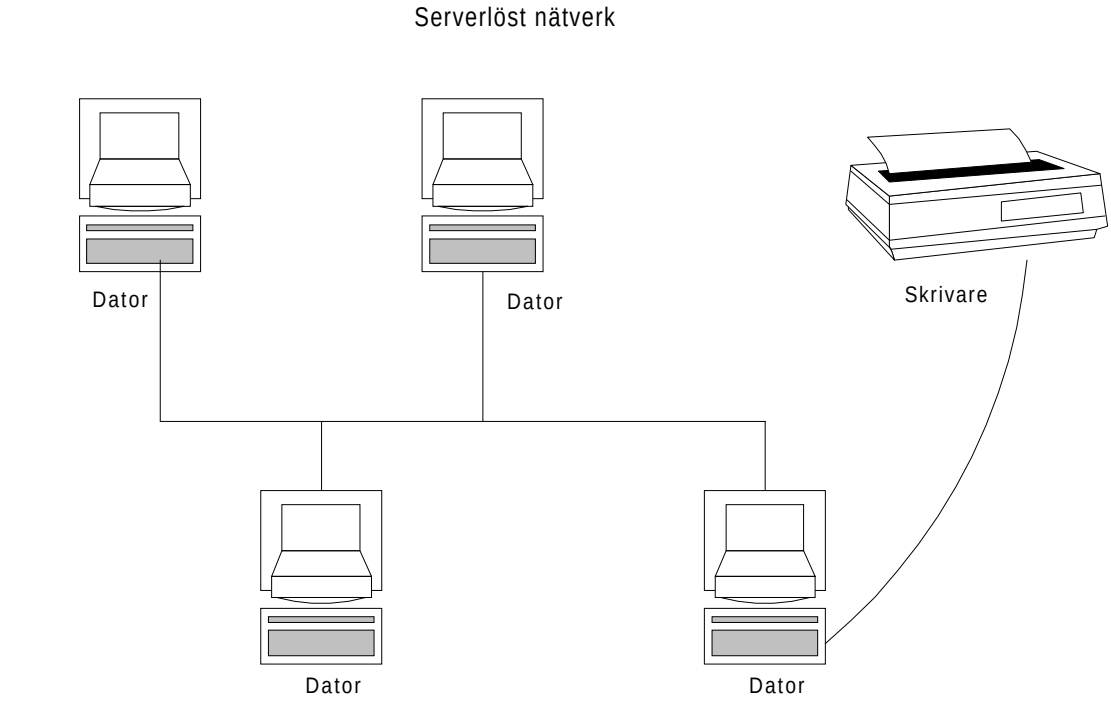
Ett nätverk innebär att två eller flera datorer har kopplats samman med kabel för att ha möjlighet att dela på information och hårdvara. I och med att användarna nu kan dela resurser och information över ett nätverk så kan stora investeringar i hårdvara som t.ex. skrivare och lagringsenheter undvikas. Dessutom kan den ineffektiva hanteringen med att utbyta externa lagringsmedier (så som t.ex. disketter) datorerna emellan undvikas.

Nätverk kan delas upp i tvåhuvud typer av nätverk, Peer-to-peer och Client /Server. Den enklaste formen, Peer-to-peer eller serverlöst nätverk innebär att alla datorerna i nätverket är jämlika, jämlika i avseende att alla datorerna i nätverket har samma potentiella åtkomsträttigheter. Dessutom finns det inte någon central lagringsenhet för data. (Tittel och Stewart, 1997)

2.1.1 Serverlöst Nätverk

Ett serverlöst nätverks fördelar ligger i att de är förhållandevis billiga och okomplicerade att installera och använda. Nackdelarna ligger dock i att det finns en begränsad kapacitet. Ett serverlöst nätverk klarar inte enligt Tittel och Stewart av att hantera mer än tio användare, dessutom så finns det ingen övergripande säkerhet och om användarna i nätverket nyttjar olika operativsystem eller olika plattformar så kan det innebära problem att koppla ihop dem. (Tittel och Stewart, 1997)

Figuren nedan visar hur flera datorer kan dela på en resurs, i fallet nedan skrivaren. På så vis behöver man inte ha en skrivare per dator, i detta fallet tre skrivare färre.



Figur 2 Serverlöst nätverk

2.1.2 Client/Server

Nedanstående material är hämtat ur (Elbert och Martyna, 1994; Khanna, 1994; Pressman, 1997) I ett server baserat nätverk, Client/Server finns det två sorters datorer, d.v.s. alla datorer i nätverket är inte jämlika, (jämlika gällande potentiell åtkomst, prestanda och användningsområde) som fallet är i ett serverlöst nätverk. Dessa olika sorters datorer är dels server, som innehåller delbara resurser och data, dels klienter vilken är den eller de datorer som nyttjar servrarnas resurser och data.

Server datorernas uppgift är att förse de andra datorerna med en mängd olika tjänster så som t.ex. kommunikation, mailhantering, filhantering, databaser osv. Servrarna blir vanligt vis anropade från en klient maskin gällande vilken tjänst servern ska utföra över ett LAN (Local Area Network) eller ett WAN (Wide Area Network).

Servern innehåller själv alla de regler och den logik som styr och övervakar serverns data. Detta innebär att en klient inte kan modifiera en servers data om nu detta strider mot serverns regler vilket resulterar i att en förbjuden transaktionen ratas. Detta medför också att en klient kan tillåtas att skriva var och när som helst utan att riskera integriteten hos en servers data.

Client/Server var först utformad för att underlätta användandet av stora databassystem där många olika användare ska kunna komma åt samma data. Tanken är att dela upp arbetsbelastningen datorerna emellan så att varje operation kan utföras på den dator som bäst är lämpad för just den uppgiften. Servern är oftast en kraftfullare maskin där datan lagras, medan klienten däremot oftast är en dator med något lägre kapacitet som

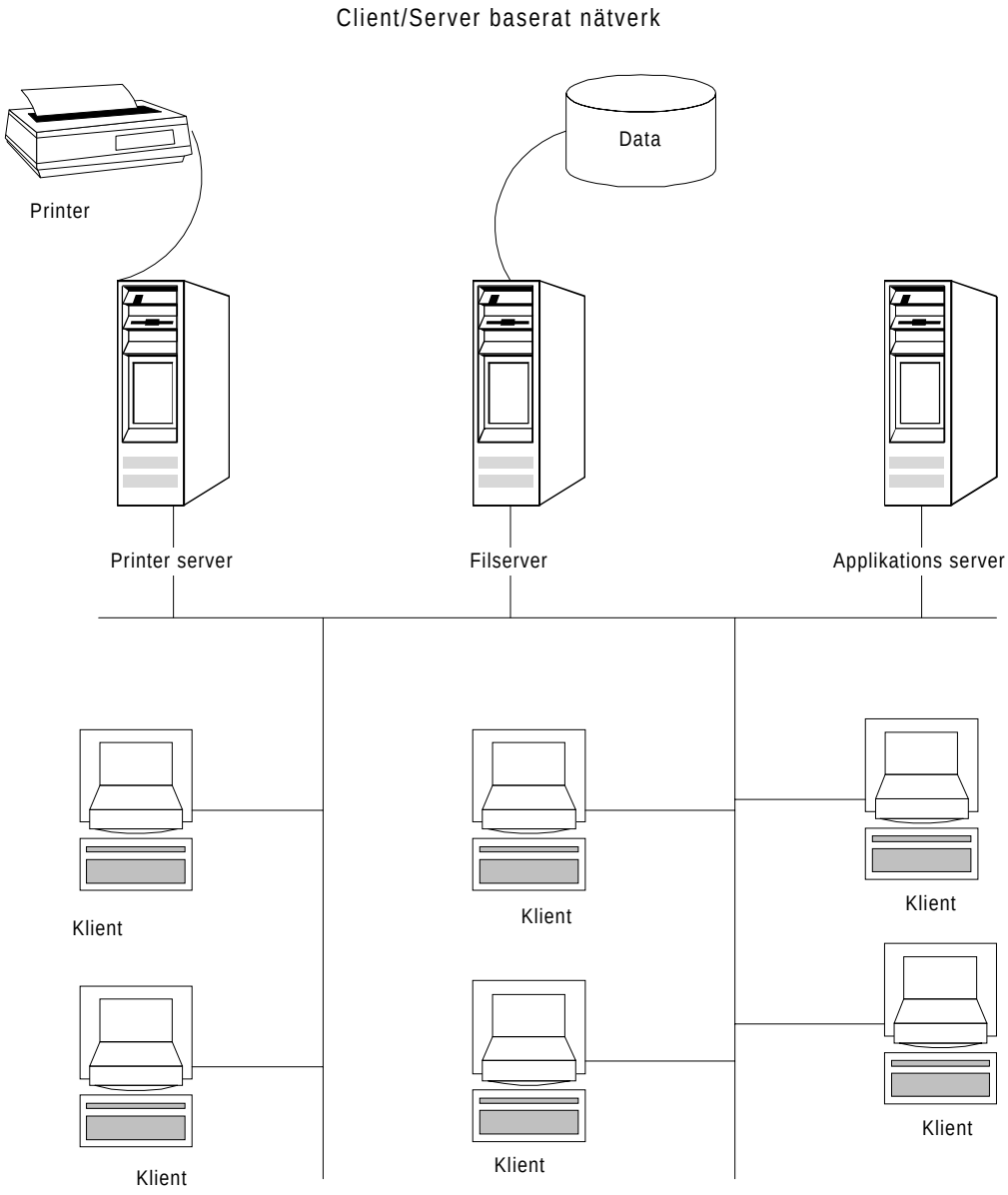
applikationerna körs på. Vissa av de applikationer som finns hos klienten är till för att komma åt den information som finns lagrad på servern. Servern skickar då tillbaka den önskade datan till klienten som lokalt omvandlar datan till rätt format beroende på vilken typ av dokument det rör sig om t.ex. excelformat, worddokument, bildformat, osv.

Man ser alltså i en Client/Server miljö inte mjukvaran som en enda stor applikation som ska implementeras på en enskild maskin. Mjukvaran avsedd för Client/Server består istället av flera klart avskilda applikationer som kan placeras ut på en server eller en klient eller distribueras mellan olika maskiner. Vanligtvis så är en applikation i en Client/Server arkitektur uppdelad i tre beståndsdelar:

- **Gränssnitts komponenten** som inkluderar alla de funktioner som brukar förknippas till GUI:et (Graphical User Interface eller Användargränssnitt på svenska).
- **Affärslogiken** som implementerar de krav som definieras av applikationen inom den domän där applikationen körs.
- **Datahantering** som utför den datamanipulation och styrning som en applikation kräver

Förutom dessa komponenter så existerar i varje Client/Server miljö ännu ett byggblock som brukar kallas för "middleware". Middleware består dels av de mjukvaru element som existerar både hos klienten och servern, dessutom innehåller den element från nätverkets operativsystem likväl som viss speciell mjukvara som stöder Client/Server kopplingen.

Figuren nedan beskriver uppdelningen i två olika sorters datorer, klienter och servrar. Figuren visar även att det är serverns ansvar att sköta de gemensamma resurserna som t.ex. printer, gemensamt datalager, gemensamma applikationer osv.



Figur 3 Client/Server nät

2.2 Komponent baserade applikationer

Kapitel 2.2 tar upp och beskriver problem och brister med hur programvara hittills har utvecklats och vilka möjligheter som dagens programvaruutveckling kan bidra med. Framför allt leder det här kapitlet till att man får en förståelse för konceptet COM (Component Object Model) som ligger till grund för en av de undersökta arkitekturerna, nämligen DCOM (Distributed Component Object Model).

2.2.1 Objektorientering

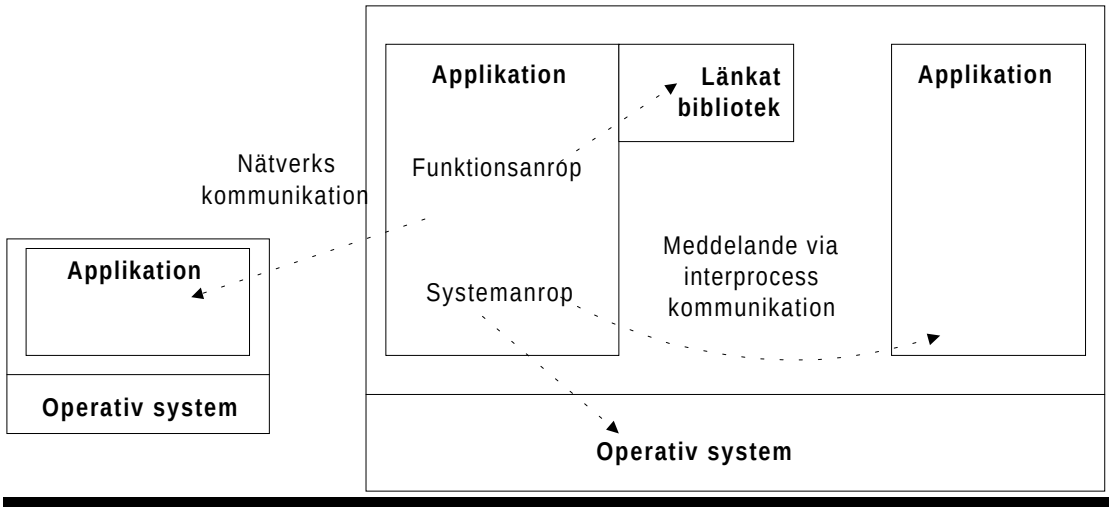
Målet med objektorientering är att försöka efterlikna den verklighet, vilken vi lever i. Efterlikna i den aspekten att dela upp verkligheten i objekt och egenskaper, t.ex. att objektet bankomat har egenskapen att ta emot kort och ge pengar (om ägaren har några vill säga) lika väl som att programmet ”addera” har egenskapen att ta emot och retunera tal. Objektorientering bygger på två viktiga principer nämligen abstraktion och klassificering. Med tanke på att vår tankeverksamhet är begränsad och för att minska antalet saker som behövs hållas reda på så används abstraktion för att vi ska kunna bortse från alla små detaljer och kunna arbeta på en mera förståelig nivå, dvs. vi behöver t.ex. inte känna till exakt hur datorn går tillväga för att spara eller hämta en fil från hårddisken för att kunna göra detta. Klassificeringens uppgift är också att förenkla förståelsen. Liksom i verkligheten klassificerar vi, t.ex. om någon säger till personen Pelle att objektet Ferarri är allt bra dyr, så säger inte detta Pelle något i och med att Pelle nu inte råkar känna till detta begrepp. Att nyttja klassificering innebär då att man säger att objektet Ferarri tillhör klassen ”bil” och har speciella egenskaper som dyr, på så sätt kan Pelle lättare göra sig en förståelse för vad personen i fråga pratade om samtidigt som Pelle kan förstå att dyr i samband med det speciella objektet är i förhållande till andra objekt i klassen ”bil”. Pelle behöver på det här sättet inte komma i håg en massa detaljer som bilen har gemensamt med andra bilar, som t.ex. har hjul, har ratt osv. utan det räcker för Pelle att veta att det är som en bil fast… Och på samma sätt är det då tänkt att objektorientering ska fungera gällande programmering, design osv. (Davis, 1994)

2.2.2 COM

COM definierar ett standardsätt på hur en datamängd skall anropa en annan datamängd. COM är alltså inte en teknologi för att implementera komponenter. Vid nyttjande av en icke COM baserad applikation beror det på vad datamängden är för något för hur ett anrop skall göras. (Chappell, 1996; Sessions, 1998)

- En applikation kan t.ex. länka till ett bibliotek och sedan få tillgång till bibliotekets tjänster genom att anropa bibliotekets funktioner.
- Fall två är när en applikation nyttjar tjänsterna som förses av en annan applikation, vilken körs i en helt separat process. Är detta fallet så kommunicerar de båda processerna med hjälp av en processintern kommunikation, vilket i normala fall kräver att man definierar ett protokoll mellan de båda applikationerna.
- Ett tredje fall är när en applikation nyttjar tjänster som tillhör en annan processor. Här får applikationen göra system anrop, ett för varje som hanteras av processorn.
- Ett fjärde fall är när en applikation behöver tjänster från en mjukvara som körs på en helt separat maskin, sammankopplade via nätverk, som då får lösas genom meddelande hantering med den externa applikationen eller genom att använda sig av RPC (Remote Procedure Calls).

Figuren nedan visar de olika anropssätten för en applikation som inte stödjer COM.



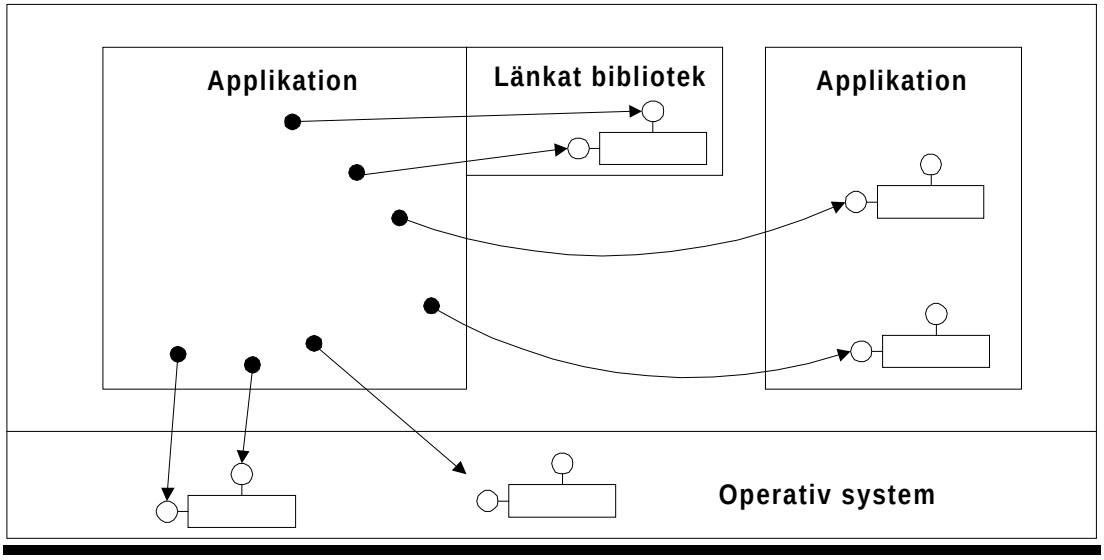
Figur 4 Anropsförfarande för vanliga applikationer. Bearbetat från (Chappell, 1996)

Vad det handlar om är att en datamängd måste nyttja en tjänst som innehas av en annan. Det är alltså detta som COM löser genom att standardisera angreppssättet där en datamängd visar sina möjliga tjänster för en annan. Detta innebär att samma metod kan användas för att anropa alla sorters mjukvara.

Ett gränssnitt i COM anropar en mängd funktioner, funktioner som en komponent implementerar och en klient kan använda sig av på samma sätt som t.ex. en dll (dynaic linked library). Men COM gör en snävare definition av vad ett gränssnitt är. I COM så är ett gränssnitt till en komponent en specifik minnesstruktur som innehåller en uppsättning av funktionspekare. Varje element i pekar uppsättningen innehåller adressen till en funktion som är implementerad i en komponent. I COM är alltså gränssnittet endast denna minnesstruktur, allt annat är en implementationsdel som COM inte bryr sig om.

I COM så är alltså gränssnitt allt. En klient ser en komponent som en uppsättning gränssnitt och det är endast genom gränssnittet som klienten kan kommunicera med komponenten. På grund av att all kommunikation sker via gränssnitt så kan man plocka bort en komponent och ersätta den med en annan. Så länge den nya komponenten stödjer samma gränssnitt som den gamla gjorde så kommer applikationen fortfarande att fungera. Det är alltså inte de enskilda komponenterna som definierar en applikation utan istället definieras en applikationen av gränssnitten som är kopplade till komponenterna. Så länge gränssnitten finns kvar så kan man uppgradera och byta ut komponenter allt eftersom. (Rogerson, 1997)

Figuren nedan ska visa att anropssättet för en applikation till en funktion utförs på samma sätt oberoende av lokalisation. Detta sker genom att applikationen anropar objektets gränssnitt som demonstreras av de vita ringarna på COM objektet nedan. Att notera dock är att COM inte klarar av anrop över ett nätverk.



Figur 5 Anropsförfarande för COM applikationer. Bearbetat från (Chappell, 1996)

2.2.3 COMs relation till objektorientering

Nedanstående är hämtat från (Chappell, 1996). Ett objekt består av två element: en definierad datatyp (även kallad tillstånd eller attribut) och en grupp metoder, (dessa kallas även objekt och egenskaper). Dessa metoder, vanligtvis implementerade som en procedur eller en funktion, tillåter ett objekts klient att utföra diverse uppgifter. Detta gäller även för objekt i COM.

Normalt i objektorienterad teknik, så stödjer varje objekt ett enskilt gränssnitt med en enda typ av metoder. COM däremot kan stödja mer än ett gränssnitt. T.ex. i C++ så innehåller ett enskilt gränssnitt all metoder som objektet har.

Ett annat bekant begrepp är klass, vilket även existerar för COMs objekt. I COM så identifierar en klass en specifik implementation av en mängd gränssnitt. Flera olika implementationer av samma sorts gränssnitt kan finnas, där varje är en enskild klass. Denna möjlighet att arbeta likadant med olika sorters objekt, där varje stödjer samma gränssnitt, men implementerar dem olika, kallas för polymorfism.

Tre viktiga begrepp i objektorientering är även inkapsling, arv och polymorfism

- Inkapsling innebär att ett objekts data inte är direkt tillgängligt för objektets olika klienter. Datan är nämligen undangömd för direktåtkomst. Objektets data kan endast kommas åt genom att använda objektets metoder. Inkapsling skyddar alltså objektets data från olämplig åtkomst och låter objektet själv kontrollera hur datan ska kommas åt. Genom att förhindra olämpliga, felaktiga förändringar att utföras direkt på ett objekts data så kan inkapsling vara till stor hjälp gällande skapande av bättre mjukvara. COM stödjer inkapsling.

- Polymorfism innebär att en klient kan behandla olika objekt som om de var samma och ändå så kommer varje objekt att uppträda som avsett. COM objekt stödjer detta koncept fullt ut.
- Arv innebär att man utifrån ett redan existerande objekt kan skapa ett nytt objekt som automatiskt inkluderar några eller alla egenskaper från det redan existerande objektet. Inom objektorientering så finns det två olika sorters arv, implementations arv som är en mekanism för att återanvända kod och gränssnitt arv som är till för att återanvända en specifikation av de metoder som ett objekt stödjer. Gränssnittsarv underlättar tillämpandet av polymorfism. COM stödjer endast gränssnittsarv, men löser återanvändning av kod genom att nyttja två mekanismer som heter containment och aggregering. Containment medför att ett objekt kan anropa ett annat för få hjälp att lösa uppgiften. (Med hjälp av aggregation så kan ett objekt visa upp ett eller flera av ett annat objekts gränssnitt som om det vore dess egna. Vad en klient ser som ett enda objekt med flera olika gränssnitt är i själva verket två eller flera objekt som aggregerar tillsammans.)

Alltså COM har mycket gemensamt med andra objektorienterade teknologier. COMs grundläggande föreställning om att ett objekt är en samling data och metoder liknar t.ex. programspråk som C++'s uppfattning om vad ett objekt är. Även COM stödjer inkapsling, polymorfism och gränssnittsarv, men återanvänder kod med hjälp av containment och aggregation istället för implementations arv. Objekt är fundamentala även för COM, men hur dessa uppträder och hur de är definierade skiljer sig något från tidigare använda objektorienterade teknologier.

2.2.4 Mjukvaruåteranvändning

Nedanstående material är hämtat ur (Chappell, 1996). Under de senaste årtiondena har hårdvaruutvecklare utvecklats från att bygga rumsstora datorer till att bygga lätta, små laptops byggda av små, kraftfulla mikroprocessorer. Under samma tidsrymd så har mjukvaruutvecklare utvecklats från att bygga stora komplexa system i assembler och COBOL till att skapa ännu större och komplexare system i C och C++, trots att detta är en framgång, så har mjukvaruvärlden inte utvecklats i samma takt som hårdvaruvärlden. Frågan är alltså vad det är som hårdvaruutvecklarna har som mjukvaruutvecklarna saknar? Svaret är förmodligen utan diskussion återanvändningsbara komponenter. Hårdvaruutvecklare bygger normalt system av färdigbyggda komponenter, där varje komponent utför en speciell uppgift och kan redovisa vad den klarar av. På detta vis så kan återanvändning av tidigare tillverkade komponenter möjliggöras och förenklas för hårdvaruutvecklaren.

Idén med att definiera återanvändningsbara delar, där varje komponent talar om dess funktionalitet genom väl definierade gränssnitt är just vad COM handlar om. Detta är knappast någon ny ide. Utvecklare upptäckte återanvändning av mjukvarans potential och kraft redan innan kompilerarens tid var här. Tekniken begränsade dock det potentiella nyttjandet av återanvändning. Existerande återanvändnings mekanismer är visserligen viktiga men de är inte tillräckligt kraftfulla.

I dag används framförallt två återanvändningstekniker: objekt och bibliotek.

- Bibliotek har mycket att erbjuda gällande återanvändning, det gäller särskilt för dynamiskt länkade bibliotek, (dll, Dynamic Linked Library). En viktig nackdel är

dock svårigheten med att lägga till funktionalitet, t.ex. hur ska man installera en ny version av ett bibliotek utan att konflikt uppstår för applikationer som nyttjar den äldre versionen av biblioteket? Bibliotek är helt enkelt inte tillräckligt kapabla för att lösa problem sammankopplade med återanvändning.

- Genom inkappsling av data och metoder, så kan objekt agera som en bra lösning gällande att plocka ihop återanvändningsbara funktionsbitar. Objekt har mer än så att erbjuda. Genom arv, så kan objekt återanvända varandra, med avseende på objektets definition av gränssnitt eller dess kod eller båda delar. Återanvändning kan förenklas ytterligare genom att använda polymorfism som döljer orellevanta skillnader från objektets klienter.

Trots dessa möjligheter så har objektteknologin inte nått ända fram till dess fulla potential gällande möjligheten att återanvända mjukvara. Hur kommer detta sig? Problemet är att det inte finns någon möjlighet att i en katalog slå upp och söka efter de komponenter man är ute efter eller att gå in i en stormarknad för mjukvara och plocka åt sig efter intresse, som fallet är för hårdvaruutvecklare. Anledningen till att detta är fallet ligger rotat i den objektteknologi vi använder i dag.

Objektorienterade språk som t.ex. C++ var utformade för att erbjuda återanvändning, men endast inom arbetsgruppen eller i alla fall inte utanför organisationen. I dag ligger tre stora problem i vägen för att möjliggöra skapandet av ett globalt ”mjukvaru stånd”.

- Det första och förmodligen det viktigaste problemet är att det inte finns någon riktig standard för att länka ihop binära objekt. Andra objektorienterade teknologier som t.ex. C++ saknar standards för korskompilering av binära objekt, vilket medför problem att skapa och distribuera binära objektbibliotek. En annan sak är att återanvändning av kod genom implementations arv tenderar att knyta ihop barn- och föräldra-objekten ganska hårt. Och är det rimligt att skaparen av en mjukvarukomponent ska skicka med källkoden och därmed avslöja sina hemligheter.

- Det andra problemet är att det för närvarande är svårt att återanvända ett objekt som är skrivet i ett språk i en applikation skriven i ett annat.

- Det tredje problemet är att om någon skapar en applikation av objekt skrivna i ett visst språk, t.ex. C++ och sedan vill göra ändringar i ett av objekten, då måste denna person som bäst, länka och kanske till och med kompilera om applikationen. Om fallet nu är att flera applikationer i ett system nyttjar det ändrade objektet så måste alla applikationer länkas eller kompileras om.

COM löser alla dessa ovanstående problem. COM objekt kan paketeras in i bibliotek eller exekverande filer och sedan distribueras i ett binärt format (utan källkoden). På grund av att COM definierar ett standard sätt att anropa dessa binära objekt, så kan COM objekt skrivas i ett språk och sedan användas i ett annat. På grund av att COM objekt blir instansierat när de behövs, så när en ny version installeras i systemet så kommer alla klienter automatiskt få den nya versionen nästa gång de nyttjar objektet.

COM tillhandahåller de fördelar som hårdvaruutvecklarna har haft av en utbredd återanvändning för mjukvaruutvecklarna. Det finns faktiskt idag flera websidor som är fullproppade med COM baserade komponenter, där man kan bläddra runt och även ladda ner komponenter. Det finns även tidningar som är fyllda med reklam för mjukvarukomponenter.

COMs generella arkitektur är användningsbar i flera syften, men att stödja skapandet av komponenter var förmodligen det största enskilda målet hos dess skapare.

2.2.5 ActiveX

ActiveX är ett väldigt luddigt och vitt begrepp, och det verkar inte finnas någon entydigt definition, men enligt Kurt D. Fenstermacher så är ActiveX en mängd verktyg som Microsoft har skapat för att göra det enklare att tillverka Windows program och nätverksapplikationer. ActiveX innebär också ett nytt sätt att formge websidor och Intranätapplikationer som avviker från mängden. Idag så består ActiveX av följande: (Fenstermacher, 1997)

- ActiveX komponenter som är en rad småprogram som t.ex. möjliggör visning av filmer, spela musik, men även funktioner i t.ex. ett formulär i form av knappar, ”radiobuttons” osv.
- ActiveX dokument som gör det möjligt för webläsaren (Internet Explorer 3.0 eller senare) att visa andra format än HTML direkt i fönstret utan att starta ett bakomliggande program.
- ActiveX skript som ser till att komponenterna fungerar tillsammans och utbyter information med varandra.
- ActiveX server framework som innehåller en mängd verktyg så som säkerhet, databas hantering osv. för att underlätta skapandet av en websida.
- Java Virtual Machine som ser till att alla ActiveX komponenter även fungerar tillsammans med Java- applikationer.

David Chappell lägger dock en annan och mera allmän betydelse i ordet ActiveX då han menar att ActiveX handlar om att definiera och ibland implementera gränssnitt för att utföra väldefinierade funktioner och att det enda som skiljer en ActiveX komponent från ett vanligt COM objekt är att ActiveX komponenten måste stödja självregestrering. Han menar därmed att på grund av likheten så kommer snart skillnaden att suddas ut helt. (Chappell, 1996)

Ytterligare ett sätt att förstå vad ActiveX är, är att titta på ActiveXs bakomliggande historia och varför det har utvecklats.

ActiveX, tidigare känt som OLE

ActiveX och OLE (Object Linking and Embedding) är ett steg i rätt riktning gällande ”bättre” mjukvara bättre med avseende på tillförlitlighet och effektivitet. Nedanstående material är hämtat från (Chappell, 1996; Brockschmidt, maj 1996). Det grafiska användargränssnittets intåg på datorerna medförde en önskan att sammanföra olika sorters format, från olika sorters program t.ex. bilder, text osv. in i ett enda dokument så att en presentation kunde göras mera lättöverskådlig. Man var nämligen tvungen att skriva ut dokumentet i fråga separat från respektive program och sedan manuellt klippa, sammanfoga och klistra fast de olika delarna in i ett dokument. Detta slapp man snart i och med införandet av ”klippbords” metaforen där man kunde nyttja kopiera, klippa och klistra operationer på datorn, för att sammanföra olika sorters programs data till ett sammansatt dokument. Ett problem fanns dock med denna teknik, den var omständlig att använda, framförallt när man skulle ändra i befintlig data och rätt

program skulle öppnas, framförallt grafik krävde en rad komplicerade, manuella steg om det ens var möjligt. För att underlätta dessa steg så skapade Microsofts Application division ett komplext protokoll. Ur detta protokoll växte OLE fram.

Den första tillämpningen av OLE, (OLE 1) var en mekanism för att skapa och arbeta med sammansatta dokument. T.ex. att bädda in (alternativet länka) ett Excel ark i ett Word dokument. Man kan alltså samla informationen på ett ställe även fast man nyttjat olika program.

Inom kort upptäcktes dock ett problem med denna teknik vilket i och för sig var ett mera generellt problem, nämligen hur skall olika mjukvarukomponenter hjälpa varandra. För att lösa detta problem skapade OLEs arkitekter en uppsättning teknologier som var tillämpningsbara till mycket mera än bara sammansatta dokument. Den främsta av dessa teknologier var COM som låg till grund för nästa OLE, (OLE 2).

COM etablerar en gemensam paradigm för samverkan mellan olika sorters mjukvara, bibliotek, applikationer och mycket annat. Följaktligen kan i stort sett vilken sorts mjukvaruteknologi som helst bli implementerad.

På grund av dessa fördelar så blev snart COM en uppsättning teknologier som inte alls längre hade någonting att gör med sammansatta dokument. Microsoft ville dock ha ett gemensamt namn att referera till gällande all COM baserad teknologi. Microsoft bestämde sig för att förkorta ner Object Linking and Embedding till helt enkel OLE, som inte längre sågs som en akronym, (initial förkortning). Versions numret togs även bort för att ytterliga förtydliga vad det handlade om.

I början av 1996 lanserade Microsoft ytterligare ett begrepp, ActiveX. I sitt första framträdande var detta begrepp förknippat med Internet och applikationer som vuxit fram ur det. På grund av att denna teknologi var baserad på COM så förknippades ActiveX direkt med OLE. Snart så började begreppet ActiveX mer och mer användas för tidigare OLE områden. Nu igen så kom termen OLE att endast stå för den teknologi som används för att skapa sammansatta dokument via antingen inbäddning eller länkning. Alla de teknologier som tidigare hade hamnat under kategorin OLE kom således att hamna under ActiveX istället.

2.3 Internet applikationer

En särskild form av Client/Server system är Internet eller ett Intranät. Internet kan beskrivas som ett WAN bestående av tusentals LAN. Internet är relativt gammalt och grundstommen har funnits i mer än två decennier. Internet har sitt ursprung i ARPANET (Advanced Research Projects Agency Networks). ARPANET var ett militärt forskningsprojekt som initierades av USAs försvarsdepartement i slutet av 1960- talet. Forskningsprojektets syfte var att undvika att skapa unika och på sätt sårbara informationscentra och istället fördela informationen och intelligensen jämnt över nätverket och på så sätt minska sårbarheten. Även fast ARPANET från början var en försvarsangelägenhet så nyttjade även olika utbildnings- och forskningsinstutioner det nya nätets finesser. Under 1980- talet så blev det uppenbart att det var de civila tillämpningarna av ARPANET som expanderade mest. Denna utveckling resulterade i världens största datornätverk, Internet.

Internet är som sagt en Client/Server miljö. Informationen är lagrad på servrar, klienter gör förfrågningar på information. Servern skickar sedan tillbaka begärd information till

klienten. Servern har ingen kännedom om klientens möjlighet att ta vara på datan och binder inte heller upp sina egna resurser genom att utföra uppgifter som lika gärna kan utföras av klienten, Detta så att servern snabbare kan bli ledig för att ta emot nya förfrågningar. (Petroutsos, 1997; Bark 1997)

2.3.1 WWW

Internet är idag ett väldigt omtalat och välkänt begrepp hos allmänheten, men det tog lång tid innan det kom att se ut på det viset. Detta berodde till stor del till att tidigare så hade gränssnittet varit kommandobaserat och man var tvungen att känna till de olika IP adresserna (Internet Protocol) för att komma rätt. Dessutom var man tvungen att ladda ner filen till den lokala datorn innan filens innehåll kunde undersökas. Internet nyttjades då främst av folk som var väldigt insatta.

Men i och med Tim Berners-Lee's (som anses som WWW's fader) "skapelse" så gjordes det möjligt att hoppa runt bland informations utbudet endast med hjälp av så kallade hyperlänkar i form av markerad text eller knappar som användaren kunde aktivera genom att klicka med musen. Uppkopplandet mot rätt IP-adress sköttes då automatiskt och användandet blev på det viset mycket smidigare i och med att adressen till rätt sida redan var inlagt av sidans konstruktör, dvs. man behövde inte själv känna till den 32-bitars långa adressen. Hyperlänkarna tillsammans med den andra informationen visas upp i en webläsare. Om nu någon vill ladda ner en websida kan det gå till så här. Klienten gör förfrågningar på information som ligger lagrad på en server, webläsaren som är en programvara hos klienten skickar denna förfrågan till servern. Servern skickar sedan tillbaka en HTML- fil (Hyper Text Markup Language) som beskriver hur sidan ska se ut och det är sedan webläsarens uppgift att tolka HTML-filen och visa upp datan.

Den dator där webserverns program körs kan också koppla upp sig mot andra datorer och agerar då själv som en klient. Det är alltså datorns funktion, och inte hårdvaran som avgör om den är klient eller server. Detta betyder då att en dator som vid ett tillfälle är en server kan vid ett annat agera klient.

WWW kan för enkelheten ses som ett nätverk av dokument där kopplingen mellan de olika dokumenten består av hyperlänkar där en användare kan "hoppa" runt med hjälp av en webläsare och hyperlänkarna utan att behöva bekymra sig om vilken server som tillhandahåller dokumentet (så vida den servern inte är nerkopplad). (Tittel och Stewart, 1997)

2.3.2 Intranät

Intranät innebär att man nyttjar en verksamhetsintern användning av Internetteknik. Det vill säga att man nyttjar den teknik som Internet grundar sig på i sitt interna LAN-nätverk. Skillnaden mellan Internet och Intranät ligger på så vis mer i storlek, användningsområde och kontrollmöjligheter än i den teknik som nyttjas. Nedan följer en uppräkning av några väsentligheter dessa två varianter emellan. Detta enligt: (Tittel och Stewart, 1997)

- Intranät är ett förhållandevis nytt begrepp och så sent som 1994 fanns det inga kända Intranät och namnet Intranät myntades först under senare delen av 1995 medan som sagt Internet har funnits ett bra tag.
- En annan aspekt är storleken. Internet är världsomspännande medan Intranät oftast förknippas med LAN, det finns visserligen Intranät som nyttjar sig av WAN men det är inte så vanligt och de blir sällan eller aldrig lika stora som Internet.
- En tredje aspekt är ägandeskapen. Internet som bekant varken ägs eller kontrolleras av någon person, företag eller land medan ett Intranät ägs och kontrolleras av den organisation som skapat det.
- En fjärde aspekt är innehållet. På Internet är informationsinnehållet inte reglerat och man kan inte alltid lita på den information som finns. I ett Intranät så finns det däremot möjlighet att kontrollera vad som ska finnas i och med att det har en ägare.

2.3.3 Dynamiskt Intranät

Att kunna koppla objekt i ett program till ett annat har fungerat länge, i alla fall om de funnits i samma dator och nu har det i och med ActiveX börjat fungera över nätverk också. Meningen är att en användare ska ha möjlighet att t.ex. kunna arbeta med objekt skapade med olika program utan att behöva hoppa mellan programmen. Den största finessen är att ActiveX kan köras i en webläsare, dvs. det är möjligt att göra programkörning i en webläsare (för tillfället endast i Internet Explorer 3 eller senare). I och med att ActiveX kan köras i en webläsare så kan ActiveX komponenten laddas ner till användarens lokala maskin och exekverar där programmet lokalt. Detta innebär då att data kan bearbetas innan den skickas iväg över nätverket, exempelvis kan en kod som ska fyllas i på en websida krypteras lokalt innan den skickas iväg eller ett fält i ett formulär kontrolleras så att det är riktigt ifyllt innan det skickas iväg över nätet.

Tidigare så var CGI- skript (Common Gateway Interface) den enda möjligheten att hantera interaktion. Nackdelen med detta skript är att det endast kan köras på servern, vilket innebär större belastning på nätverket än vid nyttjandet av ActiveX.

ActiveX ger alltså möjlighet till utökad funktion för en websida, men risken är att oönskade program ”slinker” med. I och med att ActiveX komponenter som laddas ner inte är något annat än vanliga program som laddas ner, så finns det inget som säger att programmet verkligen gör vad skaparen av programmet utlovar att det ska göra, vilket innebär att ActiveX komponenten lika väl kan formatera hårddisken som att t.ex. räkna ihop en summa på ett formulär.

Säkerheten i ActiveX hänger på något som kallas för ”authenticode” vilket innebär att alla har möjlighet att se vem som tillverkat ActiveX komponenten och välja att ladda ner programmet om tillverkaren verkar seriös. Användandet av authenticode är dock frivilligt och kostar dessutom pengar, vilket har bidragit till att det inte är speciellt utspritt och finns mest på komponenter som laddas ner från Microsofts egna hemsidor.

En ytterligare fördel vid nyttjandet av ActiveX ligger i själva tillverkandet av en websida. Genom att bygga upp en websida med redan existerande komponenter så går processen snabbare. Dessutom kan avlusning av programmen ignoreras då komponenterna redan är välfungerande program.

2.4 Distribuerade applikationer

Detta kapitel tar upp och beskriver Microsofts lösning på distribuerade applikationer, nämligen DCOM. Dessutom beskrivs även ett alternativ för att visa på att de finns fler varianter.

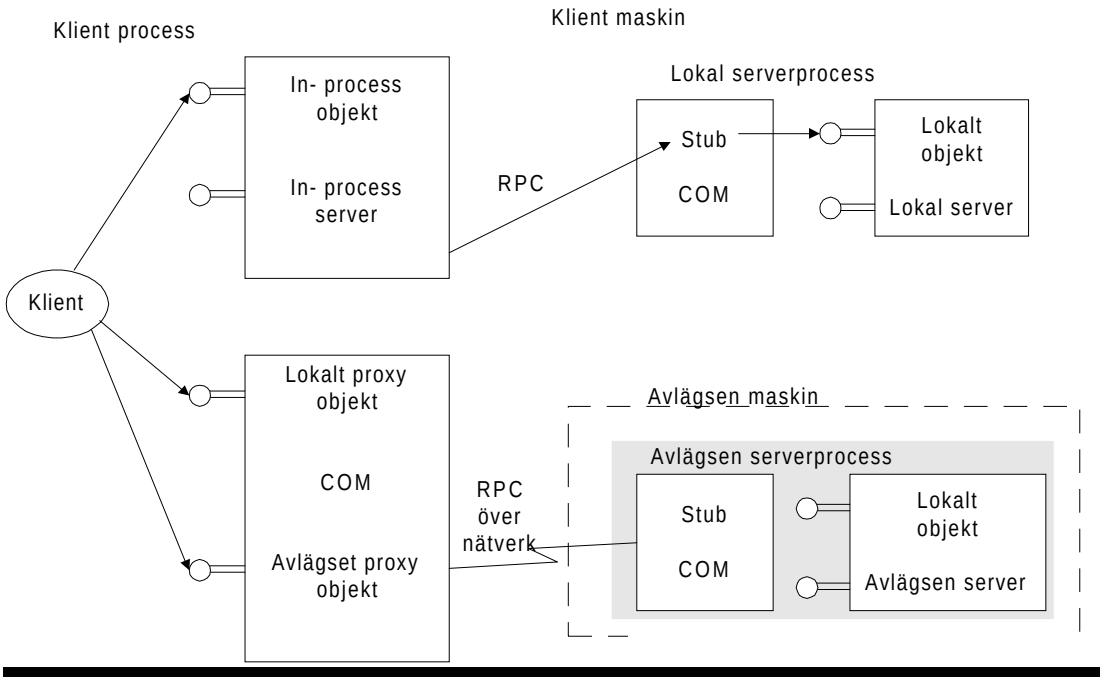
2.4.1 DCOM

DCOM (Distributed Component Object Model) är en förlängning av COM så att kommunikation numera även fungerar över nätverk. COMs uppgift är att definiera hur komponenter och deras klienter ska kommunicera. En klient som behöver kommunicera med en komponent i en annan process kan inte anropa denna direkt utan måste använda någon form av ett protokoll som erhålls av operativsystemet. COM möjliggör denna kommunikation på, för användaren, ett helt genomskinligt sätt. COM översätter då anrop från klienten och skickar dem vidare till en komponenten. Om nu en klient och den komponent som klienten anropar skulle råka befinna sig på två separata datorer så ersätter DCOM det lokala protokollet med ett nätverksprotokoll, (RPC) detta utan att vare sig klienten eller komponenten känner till det. DCOM döljer härmed en komponents lokalisation helt för klienten, så att vare sig om komponenten fysiskt finns i samma process eller på en dator på andra sidan jorden så anropar klienten komponenten på samma sätt.

I praktiken så finns det inga större skillnader mellan COM och DCOM. DCOM kräver dock tre funktioner utöver vad som automatiskt kommer med COM för att fungera. Dessa extra funktioner är teknik för att skapa objekt på en extern maskin, ett protokoll för att anropa dessa och en mekanism för att möjliggöra behörighetskontrollerad åtkomst till dessa objekt.

I och med att DCOM endast är en förlängning av COM så innebär det att DCOM direkt kan nyttja de investeringar, komponenter och verktyg som gjorts med tanke på COM.(MSDN, 1998; Sessions 1998)

Figuren nedan påvisar att klienten kan anropa ett objekt på samma sätt oberoende av det objektets lokalisation. Den övre halvan nyttjar COM och den nedre COM med hjälp av RPC över ett nätverk. Serverprocessens lokalisation är helt osynlig för en klient som nyttjar COM förfarandet och gör därmed anrop på samma vis oberoende av serverprocessens lokalisation då klienten tror att serverprocessen alltid har samma lokalisation.



Figur 6 Kommunikation COM och DCOM bearbetat från (Bernstein, 1998)

2.4.2 Alternativa lösningar till DCOM

Microsoft är nu inte ensamma att lösa problemet med distribuerade applikationer. OMG (Objekt Management Group) har löst detta genom att ta fram ORB (Objekt Request Broker) som är en ”middleware” komponent. Denna ”middleware” (ORB) låter ett objekt som finns hos klienten sända ett meddelande till en metod som är inkapslad av ett objekt som finns hos servern. ORBs uppgift är att översätta meddelanden, hantera kommunikation och koordinera aktiviteter som behövs för att leta reda på det objekt som meddelandet är adresserat till, starta dess metoder, förse objektet med lämplig data och skicka tillbaka resultatet till det objekt som genererade meddelandet från början. (Pressman, 1997)

3 Problem

3.1 Inledning

Många av dagens utvecklingsprojekt läggs ner innan de ens blir färdiga eller anses som mindre lyckade efter genomförandet. En stor bov i detta drama är att det arkitekturval man gör inte passar ihop med det system/ de applikationer man väljer att ha. Det verkar råda en koppling mellan den arkitektur som passar bäst ihop med en viss typ av system/ applikationer. (Linthicum, december 1996)

Min problem uppgift kommer ligga i att göra en överblick på ett antal arkitekturer och se på respektive arkitekturs för och nackdelar. Detta för att i slutändan försöka definiera ett antal kriterier som kan underlätta ett arkitekturval för en applikation inom en organisation.

Visserligen finns det många tidigare som gjort jämförelser arkitekturer emellan, men dessa undersökningar behandlar oftast endast två olika lösningar och då ofta inom samma kategori t.ex. Intranät med ActiveX kontra Intranät med Java, DCOM eller CORBA osv. Dessutom brukar dessa jämförelser mer koncentrera sig på små tekniska skillnader medan detta arbete är ute efter att kartlägga övergripande för och nackdelar arkitekturerna emellan.

3.2 Avgränsning

En avgränsning ligger i att endast göra en övergripande fokusering på de olika arkitekturerna och på så sätt undvika att jämföra tekniska detaljer. Dessutom ska arbetet begränsas till att endast undersöka fyra olika sorters arkitekturer, nämligen.

- Klassisk Client/Server
- Statiskt Intranät med HTML, utan stöd för ActiveX teknik
- Dynamiskt Intranät med HTML, med stöd av ActiveX teknik
- Distribuerade komponenter och tittar då på Microsofts lösning med DCOM.

4 Undersökningsmetoder

Idag finns det två stora angreppssätt för hur en undersökning ska gå till väga. Detta beroende på vad man vill få fram och har för avsikt med sitt forskningsarbete. Dessa två är kvantitativ undersökning, där en enkätundersökning är en typiskt angreppsmetod och kvalitativ undersökning, där en ingående intervju kan vara en typisk angreppsmetod. Nedan följer lite mera utförligt beskrivet om kvalitativa och kvantitativa undersökningsmetoder. (Holme och Solvang, 1991)

4.1 Kvalitativ undersökning

Styrkan hos kvalitativa data och metoder ligger i att de visar på totalsituationen och med hjälp av denna helhetssyn kan en ökad förståelse för sammanhang och processer möjliggöras. En mycket viktig del i kvalitativa undersökningar ligger i att skapa en grund för teorikonstruktion. Detta angreppssätt innebär oftast en intensiv granskning av varje undersökningsobjekt. En annan aspekt är att urvalet inte i lika stor grad behöver vara representativt som fallet är i kvantitativa undersökningar. Detta då det är olika personers synpunkter och åsikter man vill åt. På grund av detta kan det till och med vara bra att undersöka några icke representativa undersökningsobjekt för att kunna belysa problemställningen från ett nytt perspektiv. En stor nackdel är dock att man inte utifrån en sådan information kan uttala sig om hur pass väl frågeställningen täcker alla undersökningsobjekt. En kvalitativ undersöknings centrala aspekt är att genom insamling av information dels kan få en djupare förståelse av det problem som undersöks och dels kan beskriva helheten av det sammanhang som problemet ryms i. Vid nyttjande av kvalitativa metoder så är det forskarens egna uppfattning eller tolkning av informationen som står i förgrunden. Detta beroende på forskarens motiv, sociala processer och sociala sammanhang. Detta betyder att det inte går att omvandla dem till statistiska siffror. (Holme och Solvang, 1991)

Exempel på kvalitativa undersökningsmetoder:

- intervju
- fallstudie
- litteraturstudie

4.1.1 Intervju

Nyttjandet av intervju behövs framförallt om det inte finns något nedtecknat material om företeelsen, utan man måste vända sig till människor för att få besked. Vad som menas med en intervju är en situation där en person ställer frågor till en annan. Intervju lämpar sig särskilt bra då man inte är helt säker på vad man vill ha svar på, då möjlighet till omformulering av frågor kan göras under en intervjus förlopp.

Fördelen med att nyttja sig av en kvalitativ intervju är att undersökningen för den intervjuade liknar en vardaglig situation och ett vanligt samtal. Den intervjuade styrs då minimalt, förutsatt att intervjuaren sköter sitt jobb. Intervjuarens uppgift ligger främst i att hålla samtalet inom givna ramar så att problemet belyses. Det handlar i detta fall om att "vaska" fram den information ur samtalet som är intressant. Denna metod kan då

vara särskilt lämplig om undersökaren i förväg inte har full insikt i vilken information som är väsentlig kunskap att utvinna då frågorna kan ändras allt eftersom undersökaren får insikt om vilken information som är den viktiga. Detta till skillnad mot t.ex. enkätundersökning då frågorna inte kan ändras efter de har blivit utskickade till undersökningsobjekten. (Holme och Solvang, 1991)

Utvärdering intervju

Denna metod kan lätt bli svår att genomföra, dels på grund av tidsbrist, då en viss del litteraturstudie ändå lär behövas för att ”rätt” frågor ska kunna ställas som i sin tur tar lång tid i anspråk, dels på grund av att det skulle ta väldigt lång tid att genomföra intervjuer för denna undersökning. Detta då många företag förmodligen skulle behöva intervjuas. Förmodligen skulle en uppdelning av intressanta undersökningskategorier vara, de som utvecklar programvaran, de som ”designar” systemet och de som får programvaran och systemet installerat hos sig. Dessutom är det förmodligen så att inget enskilt företag håller på, eller har erfarenhet av samtliga av de olika typerna av tillämpningar, då de flesta verkar specialisera sig inom sin respektive nisch.

Samtidigt då som det kan vara svårt att hitta företag som håller på med eller har erfarenhet av respektive arkitektur skulle det med stor sannolikhet bli kostsamt då man förmodligen får vara beredd att resa utanför orten, för att få tag i representativa företag vilket då kan innebära en hel del resekostnader.

4.1.2 Fallstudie

Att genomföra en fallstudie innebär att ett fåtal företeelser undersöks angående en mängd avseenden till skillnad mot t.ex. statistisk undersökning då många företeelser gällande ett fåtal avseenden. Det innebär oftast att det kan vara svårt att få en generell uppfattning om ett problemområde. En fördel med fallstudien är möjligheten att sätta in problemet i ett verkligt perspektiv och på så sätt upptäcka information lättare än annars. (Holme och Solvang, 1991)

Fallstudie inom utrednings- och forskningsmetod används framförallt i följande sammanhang: (Holme och Solvang, 1991)

- Som illustration. I detta fall får fallbeskrivningen oftast en pedagogisk och förtydligande funktion.
- Som hjälpmedel att skapa hypoteser. I detta fall är oftast problemområdet relativt okänt eller obearbetat. Den kan också användas för att få en ny infallsvinkel till ett tidigare studerat område.
- Som metod vid aktionsforskning/ förändringsarbete t.ex. i samband med en organisationsutveckling. Forskaren måste då skaffa sig ingående kunskap angående organisationen och dess aktörer, detta t.ex. i form av begrepp och språk.
- Som hjälpmedel för att skapa en ny teori.

Utvärdering fallstudie

Denna metod skulle komma att ligga som ett komplement till litteraturstudien där en befintlig applikation, skolans schemaläggnings system kommer att undersökas och försöka anpassas till respektive arkitektur och på så sätt se på möjligheter och svagheter med de olika arkitekturerna. Visserligen hade det ju varit att föredra att nyttja sig av en riktig fallstudie, detta då annars spekulationer och antagande mer än fakta kommer att spegla denna metoddel. Anledningen dock till att inte försöka sig på en verklig fallstudie hänger till stor del på tidsaspekten, det skulle vara omöjligt att anpassa en applikation av någorlunda komplexitet till fyra olika arkitekturer innan ”deadline” och att försöka leja ut detta arbete är klassat som orealistiskt.

4.1.3 Litteraturstudie

Litteraturstudie innebär en rad speciella problem, framförallt härrörande nyttjandet av källor, vad källan är baserad på, vad som antecknas av det som händer, vad som blir bevarat, syfte med källan, ålder osv. En källa kan förklaras som material som är skriftligt nedtecknat.

En källa vill förmedla sina läsare information, vilken kan vara av olika slag. En uppdelning kan göras i normativ kontra kognitiv information. Normativ information är värderande och kan hittas i t.ex. lagar och förordningar. Kognitiv information däremot är berättande och syns t.ex. i offentlig statistik. Beroende på syftet med arbetet kan det vara lämpligt att nyttja sig av övervägande den ena eller andra formen av information. Om syftet t.ex. är att få en allmän uppfattning om en situation och hur det upplevs kan det vara fördelaktigt att nyttja sig av kognitivt material medan normativt material kan vara lämpligt då förhållningssätt, avsikter och krav är det intressanta. (Holme och Solvang, 1991)

Vid nyttjande av litteraturstudie bör fyra värderingar göras av materialet. (Holme och Solvang, 1991)

- observation
- ursprung
- tolkning
- användbarhet

Observation

Vilka källor kan belysa problemställningen. Det är viktigt att skaffa sig en överblick över vilket material som finns tillgängliga och vilka som kan vara av intresse.

På grund av att det är omöjligt att granska allt material, detta på grund av kostnads- och tidsbrist, så får man vara varse att ens sållningsprocess som till stor del kan råka bero på tillfälligheter lätt kan resultera i ett systematiskt skevt material. Ett exempel är att bara råka nyttja material från en av parterna i en jämförelse.

Ursprung

Här gäller det att bestämma vad eller vem som är upphovet till materialet. Detta för att kunna bedöma vilket samband som finns mellan källan och vad den beskriver.

Tolkning

Tolkning innebär att man ska analysera vad som står i materialet och vad upphovsmannen avsett att säga med det. Målet är att kunna ge en sammanfattande bild av informationsbärande struktur och helhet.

Användbarhet

Hur relevant är materialet för problemställningen. Exempel hur nära eller hur direkt är materialet kopplad till en bestämd situation. En annan viktig fråga är hur trovärdig materialet är. För att svara på trovärdighetsfrågan nyttjar man två analyser, extern och intern analys. Den yttre analysen innebär att göra en jämförelse med annat material. Den inre analysen tittar istället på aspekter som materialets inre överensstämmelse, upphovsmannens subjektiva perspektiv osv.

Utvärdering litteraturstudie

Tillgången på material för de olika arkitekturerna kommer med stor sannolikhet att variera kraftigt rörande tillgång på böcker och artiklar. Detta beroende på att de olika arkitekturtyperna som ska undersökas varierar mycket i hur länge de har funnits. Äldre typer som t.ex. Client/Server gör det lättare att hitta böcker om ämnet men gör att det blir i mycket svårare att hitta artiklar då ämnet inte i samma grad debatteras och granskas lika mycket längre. Böcker har sin stora fördel i att de lättare ger en grundförståelse och är mera heltäckande samtidigt som de ofta är relativt nyanserade gällande åsikter. Nyare arkitekturer som t.ex. DCOM gör dock att det inte finns så många skrivna böcker men å andra sidan så innebär det förmodligen att det lär gå att hitta mycket artiklar då många har åsikter som behöver ventileras. Samtidigt som artiklar ger fördelen i färskhet samt många olika personers perspektiv på saken så är artiklarna ofta väldigt påverkade av författaren.

Detta gör att materialet kan skifta lite grann, men då detta är känt från början bör detta kunna tas med i beräkningarna så att källorna kan studeras på ett relativt objektivt sätt.

4.2 Kvantitativ undersökning

En kvantitativ undersökning är mycket mera strukturerad än vad en kvalitativ undersökning är och är i mycket större utsträckning präglad av kontroll från forskarens sida. En kvantitativ metod definierar vilka förhållanden som är särskilt intressanta utifrån den frågeställning som är gjord. Dessutom vid nyttjande av en kvantitativ metod behöver intervjuaren känna vilka svar som är tänkbara, i alla fall vid nyttjande av t.ex. enkäter som ofta har alternativa svars alternativ. Statistiska mätmetoder spelar en väsentlig roll i analysen av kvantitativt insamlad data. (Holme och Solvang, 1991)

Exempel på kvantitativ undersökning

- Enkätundersökning

4.2.1 Enkätundersökning

Liksom vid intervjuförfarandet kan denna metod nyttjas då nedskrivet material omkring problemet saknas och går liksom intervju ut på att fråga en person en massa frågor, den stora skillnaden är dock att det inte är en person med under intervjutillfället. Den intervjuade får klara sig med de skrivna instruktioner som kan skickas med tillsammans med svarsformuläret. Enkät jämfört med intervju kräver större förkunskaper om vilken information som är den väsentliga. Detta då frågorna blir statiska, dvs. frågorna kan inte ändras allt eftersom svaren börjar komma in. Enkät är framförallt att föredra då många personer ska frågas då detta blir väsentligt billigare. Dessutom minskar risken för att intervjuaren påverkar den undersökta, medvetet eller omedvetet. Problemet är dock att risken för bortfall blir större. (Holme och Solvang, 1991)

Utvärdering enkätundersökning

Denna metod saknar till stor del intresse för detta arbete, detta i och med att en enkätundersökning framför allt leder till en kvantitativ undersökning, vilket inte är av intresse i det här fallet. En annan aspekt är riskfaktorn, om svarsbortfallet blir stort finns det risk att undersökningen blir oanvändbar. Ytterligare en orsak är att man blir så tidsmässigt uppbunden då man inte kan göra så mycket annat så vida man inte nyttjar flera olika metoder, detta framförallt medan man väntar på inkommande svar. Dessutom kan denna metod också bli relativt kostsam, då utskick och påminnelser till ett tillräckligt stort urval måste göras.

5 Val av metod

Inriktningen med detta examensarbete kommer att vara av hermeneutisk karaktär, dvs. ge insikt och förståelse angående min problemställning och inte försöka bevisa något fenomen. Beroende på denna inriktning så är det sannolikt att kvantitativa metoder av olika slag inte kommer att tillföra detta arbete något av intresse. Detta har resulterat i att t.ex. enkäter av olika slag inte kommer att användas som någon metod och även inte ingå i nedanstående selektionsfas. Detta gör dock att det ändå finns vissa olika metoder inom den kvalitativa området att välja bland. Vid val av olika kvalitativa metoder har olika kriterier som är intressanta fått vägas mot varandra. Dessa är:

- **Tidsåtgång:** Beroende på examensarbetes begränsade tidsrymd och fasta deadline är det viktigt att man verkligen hinner genomföra arbetet.
- **Kostnad:** Eftersom inga extra finansiella medel finns att tillgå så är det viktigt att kostnaden är låg.
- **Åtkomst:** Vissa källor kan vara svåra att få fram och få tillgång till att nyttja.
- **Krav på förkunskap:** Ibland kan det behövas viss bakgrundskunskap om ett ämne för att kunna få fram ny information med hjälp av en speciell metod.

Det har framförallt funnits tre intressanta alternativ, som är:

- Litteraturstudier. Denna metod blir nog förhållandevis billig. Det finns relativt stor tillgång till material då det finns många källor att välja bland, olika bibliotek, bokaffärer, Internet, artikeldatabaser osv. vilket väsentligt ökar åtkomstsmöjligheter till material. Dessutom går det att påverka tidsåtgången då man t.ex. inte behöver vänta på någon annan som fallet t.ex. blir vid en intervju. Litteraturstudie kräver låg förkunskap, då endast ungefärliga kunskaper krävs för att man ska hamna någorlunda rätt bland materialet.
- Intervju. Denna metod blir nog även denna hyfsat billig, något dyrare än litteraturstudie dock. Detta i och med resor fram och tillbaka samt ev. telefonräkningar. Åtkomsten är också stor, då det finns många företag inom datasektorn. Tidsåtgången kan nog vara rätt varierande beroende på vilka företag man får tag i och hur upptagna dessa är. Kräver viss förkunskap, då man måste vara någorlunda insatt för att kunna ställa relevanta frågor.
- Teoretisk fallstudie. Detta som alternativ till en verklig fallstudie med implementation då detta skulle bli orealistiskt svår för alla ovanstående kriterier. Denna metod innebär att tidsåtgången kan hållas nere relativt mycket då den kommer att nyttjas i samband med litteraturstudien. Tillgången på samma program i de fyra olika arkitekturerna lär dock vara svårt, kostnaden lär dock kunna hållas nere i och med att det görs teoretiskt. Kräver någorlunda förkunskaper, då man annars inte vet och kan känna igen vissa karakteristiska drag arkitekturerna emellan.

Nedan finns en sammanfattande graf över fördelar och nackdelar med respektive metod.

	Litteratur studier	Intervjuer	Teoretisk fallstudie
Tids åtgång	—	—	+
Tillgång	+	+	—
Kostnad	+	—	+
Krav på förkunskap	+	—	—

Utifrån dessa övervägande kommer litteraturstudie och teoretiskt fallstudie att nyttjas för detta arbete.

6 Källor

Detta kapitel är uppdelat i två delar. Det första tar upp intressanta bokkategorier för detta arbete och den andra delen tar upp de böcker och tidskrifter som nyttjats.

6.1 Intressanta källor

Åtminstone en bok om Client/Server, detta då alla arkitekturer som ska jämföras bygger på någon form av detta tänkande. Dessutom ska en lite äldre bok nyttjas då det dels inte på den tiden fanns så många varianter, utan framförallt grundtanken och dels tydligare se på brister som kunnats åthjälpas senare. Som komplettering ska även en något nyare bok nyttjas, detta för att se på varianter som vuxit fram och vilka problem man framförallt vill få bukt med de nya varianterna. (Elbert och Martyna, 1994; Renaud 1996)

Åtminstone en bok om Internet/Intranät för att förstå varför denna variant av Client/Server är så populär idag, dels se på skillnader mot traditionell Client/Server och på vinster respektive eftergifter man får göra om man väljer denna arkitektur (Tittel, 1997)

Det kan vara svårt att hitta bra litteratur angående ActiveX och dynamiskt Intranät, detta då ActiveX först lanserades 1996, men de böcker som finns att tillgå tillsammans med boken om Intranät/Internet och artiklar angående de båda ska räcka. . Den bok som nyttjats för denna avdelning är: (Chappell, 1996)

För DCOM kan det vara svårt att hitta bra böcker då detta är så väldigt färskt, och kommer framför allt att söka artiklar inom detta område för att få en djupare insikt. Problemet är att det mesta materialet som finns angående detta lär finnas på Microsofts hemsidor, vilket får tas med i beräkning då DCOM är en Microsoft produkt. . Den bok som nyttjats för denna avdelning är: (Sessions, 1998)

6.2 Kort om källor som valts

COM och DCOM av Roger Sessions

Detta var den enda bok som gick att hitta angående DCOM. Denna bok gav möjlighet till en lite större helhet än vad som tidigare erhållits genom att bara läsa lösryckta artiklar. (Sessions, 1998)

Understanding ActiveX and OLE av David Chappell

Detta var en väldigt tekniskt och faktaspäckad bok om COM och ActiveX, kanske lite för tekniskt inriktad då den till stor del verkar vara riktad mot programmerare, men gav ändå en hel del förståelse kring varför och inte bara att. (Chappell, 1996)

ActiveX för dummies av Kurt D Fenstermacher

Denna bok innehöll vissa intressanta och bra bitar för grund och förståelse kring ActiveX men annars inte något vidare då den främst riktade in sig på att beskriva hur man gör websidor med hjälp av ActiveX. (Fenstermacher, 1997)

Intranät bibeln av Ed Tittel och James Michael Stewart.

Denna bok var bra för att få en heltäckande, introducerande bok till Intranät teknologin, fördelar och nackdelar gentemot att nyttja en Client/Server miljö. Dessutom innehöll boken väldigt många websideadresser för vidareläsning. (Tittel, 1997)

Client/Server computing av Bruce Elbert och Bobby Martyna

En äldre bok om Client/Server (1994). Fördelarna med denna arkitektur var framförallt gentemot stordator miljö och det gick att se för och nackdelar med arkitekturen tydligare då det ännu inte hade utvecklats så mycket hård och mjukvara för att förenkla tillvaron med Client/Server. (Elbert och Martyna, 1994)

Introduction to Client/Server systems av Paul E. Renaud

En nyare bok om Client/Server (1996). Mer hjälputrustning och mera avancerade lösningar har dykt upp. Boken är även lagd på en ytligare nivå än boken Client/Server computing som väldigt mycket går ner på en teknisk nivå. Annars är även denna bok mest inriktad på fördelar med Client/Server lösningar gentemot stordator systems lösningar. (Renaud, 1996)

Software Engineering, fourth edition av Roger S. Pressman

Denna bok användes som kursmaterial i programvaruutvecklings kursen och var ämnad att nyttjas för att se om det gick att hitta problem med att utveckla, testa osv. olika former av distribuerade applikationer. (Pressman, 1997)

7 Genomförande

Detta kapitel är uppdelat i fem olika delar. Den första delen inleder med allmän information om nätverk, för och nackdelar med en sådan lösning. Detta för att samtliga av de undersökta arkitekturerna är avsedda för nätverk. Del två tar upp och behandlar Client/Server, del tre statiskt Intranät , del fyra dynamiskt Intranät och del fem DCOM.

7.1 Allmänt nätverk

Då samtliga arkitekturer är byggda som nätverksarkitekturer så beskrivs nedan lite allmänna aspekter att observera angående nätverk. Nedanstående material om nätverk är hämtat från (Renaud, 1996; Elbert och Martyna, 1994).

Fördelar små nätverk

- Ingen större fara med bandbredden, behöver inte bry sig om att ha de effektivaste konfigureringarna och protokollen.
- Servrarna behöver ej vara så kraftfulla då de ej måste kunna hantera databehandling från så många klienter.
- Enklare att administrera och konfigurera, behöver ej göras på så många komponenter.
- Den höga graden av samverkan i ett Client/Server nätverk kan göra det svårt att endast uppgradera en viss del av systemet, vilket förmodligen inte gör det mycket extra dyrare och krångligare att uppgradera resten, om nätverket är litet. (Renaud, 1996)
- Uppdatering och hantering av data blir smidigare när det inte behöver skötas på så många servrar.

Nackdelar små nätverk

- Lönar sig inte att köpa in avancerad programvara, kan inte fördela kostnaderna på så många användare. Detsamma gäller med specifik hårdvara.
- Ofta skapat med avsikten av att ha ett litet LAN vilket gör det svårt att skala upp. (Renaud, 1996)

Fördelar stora nätverk

- Ej så kostsamt per användare att köpa in avancerad programvara eller hårdvara som kan medföra underlättnader i verksamheten.
- Ofta enklare att skala upp och man har förmodligen haft det i baktanke vid införandet av nätverket.
- Viss verksamhet kräver det för att alla anställda ska kunna ta del av samma information.

Nackdelar stora nätverk

- Svårt att administrera, förmodligen utspritt samt många komponenter, på många platser.
- Dyrt måste ha kraftfulla servrar och effektiva nätverksprotokoll för att klara av belastningen från många användare.
- Problem med bandbredden, många användare som ska dela på den.
- WAN innebär ofta problem i sig, nyttjar bl.a. analoga signaler. (Renaud, 1996)
- Många komponenter innebär många felkällor.
- Svårt och dyrt att ändra plattform, operativsystem osv.
- Kräver större krav på välfungerande grupprogram vilket kan vara dyrt att köpa in

Några generella problem vid uppskalning

- Bandbredden, fler ska dela på nätverket.
- Fler komponenter som ska underhållas.
- WAN långsammare, analogt och mer felbenäget än ett LAN. (Renaud, 1996)
- Problem med att undvika WAN genom att koppla ihop flera LAN med hjälp av Internet resulterar i bl.a. en röra av olika lösningar, säkerhetsproblem, krav på ”brandväggar”.

7.2 Client/Server

Nedanstående material är hämtat från (Tittel och Stewart, 1997). Det fanns vissa förutsättningar för att Client/Server tänkandet skulle bli möjligt, nämligen

Skapandet av ”intelligent” persondator utrustning under mitten av 70- talet. Dessa var baserade på relativt billig mikrodator teknologi och var förstärkt med generella operativsystem och intelligent databehandling. Detta bidrog till att persondatorn blev en realitet. Detta nya koncept om persondatorer revolutionerade tankesättet om system och man började inse möjligheterna med datornätverks arkitektur. Innan var all kommunikation implementerad som ett separat monolitiskt undersystem. Vidare var varje gränssnitt mellan applikation och dess "kommunikations delsystem" skräddarsytt för varje applikation, vilket gjorde det svårt att skriva dessa applikationer. Datornätverket började växa fram i slutet av 70-talet.

Dessa saker tillsammans möjliggjorde att Client/Server arkitekturen kunde börja växa fram i mitten av 80- talet som alternativ till stora och inflexibla stordatormiljöer.

Normalt sett så är klienten ämnad till datapresentation och applikationshantering medan servern sköter databasåtkomst och kontroll.

Client/Server behöver några eller alla av följande element i distribuerad datorisering: (Khanna, 1994)

- Nätverkskoppling mellan klient och server

- Samverkan mellan klient och server
- Någon form av RPC mekanism
- Lagrade procedurer och utlösare
- Stöd för SQL (Structured Query Language)
- Datareplikering
- Transaktionshantering
- Data caching
- Fjärradministrering av data
- Säkerhet

Client/Server administrering är omfattande och inkluderar bl.a. följande aktiviteter: (Renaud, 1996)

- Konfigurering: inventering, installationer, konfigurering och versionskontroll
- Felisolering och reparation: identifiering, lösning och uppspårning av fel, eventuell backup hantering och återhämtning
- Säkerhetsadministration: kryptering, lösenord, fysisk säkerhet (stöld, brand osv.), säkerhetspolicy (vem har tillgång till vad och vem kan ändra inställningar)
- Prestanda aspekter: realtid, statistik över trafikvolym, resursutnyttjande, "trafikstockning" på nätet
- Kostnadsadministrering: lokalisering av kostnader för användning: uppkopplingstid, nätverksdatalagring, utskrivna sidor osv.

Så mycket som 73 procent av kostnaderna i att äga och sköta ett Client/Server system består av personalkostnader ej teknik kostnader. (Renaud, 1996)

Dessutom på grund av den korta livscykeln på datorutrustning så krävs mycket konfigurering och installation av den ständigt nyinköpta maskinvaran. Detta var särskilt synligt vid tvålagers arkitektur då varje klient var tvungen att konfigureras om mot varje ny server som installerades.

Fördelar Client/Server

Nedanstående material är hämtat ur (Renaud, 1996; Elbert och Martyna, 1994).

- Väl utprovad, har varit med länge och testats ordentligt. Har också lett till att det finns mycket programvara och protokoll att välja mellan för att bäst passa en organisations lösning. Det finns dock många standarder för t.ex. protokoll (ingen enhällig standard som till skillnad mot Intranät med TCP/IP och HTTP).
- Kan minska nätverksbelastningen om man distribuerar datan på ett riktigt sätt
- Det finns många avancerade program som t.ex. Lotus Notes och Microsoft Exchange som kan bidra med en mängd funktioner som t.ex. databassäkerhet och replikering.

- Möjliggör avancerade gränssnitt
- Klienten kan ta hand om viss databehandling och på så vis minska belastningen på servern (dela på jobbet) vilket ett av syftena är med Client/Server.
- Finns specialiserade protokoll för olika uppgifter, t.ex. SQL anrop för databashantering.

Nackdelar Client/Server

Nedanstående material är hämtat ur (Renaud, 1996; Elbert och Martyna, 1994).

- Medför ofta att man låser möjliga plattformar då programvaran som används oftast inte är möjlig på flera olika plattformar och kräver vissa operativsystem osv.
- I och med komplexiteten att administrera och underhålla ett Client/Server nätverk så vill man göra standard konfigureringar hos klienterna för att spara tid. Detta leder till motsträvighet att ha många speciella konfigureringar för speciella syften.
- P.g.a. plattforms beroende så måste denna aspekten iakttas vid införskaffning av ny programvara. Detta så att programmet kommer att fungera i det befintliga systemet.
- I och med komplexiteten så måste man tänka på skalbarheten redan vid införandet eftersom att det kan vara svårt annars efteråt att försöka skala upp. (Renaud, 1996)
- Då det är krångligt att administrera och konfigurera ny programvara till Client/Server system kan det vara vanligt att uppdateringar av programvaran sker paketvis. Nackdelen kan vara att man får vänta tills man hittar mer än en intressant programvara som ska köpas in. (Renaud, 1996)
- Versionskontroll så att alla klienter nyttjar samma version av en applikation är ett omfattande arbete i en Client/Server miljö.

Traditionell Client/Server fungerade väl, så länge affärslogiken hölls enkel och då affärslogaritmerna inte ständigt ändrades. På denna tid existerade heller ingen kommunikation mellan servrarna utan endast klient till serverkommunikation. Men snart blev affärslogiken så komplex att den inte kunde hantera på en PC, dessutom blev klientsystmen för svåra och arbetskrävande att underhålla då affärslogiken ofta behövde uppdateras. Detta tillsammans med Internet aspekten, där klienten endast hade hand om presentations delen och affärslogiken hamnade på servern som man inte hade så stor kontroll över ledde till ett tryck på att möjliggöra server till serverkommunikation samt att dela upp applikationen i tre olika lager, nämligen: (Sessions, 1998)

- Gränssnittslager
- Affärslogik
- Datahantering

Stegvis p.g.a. teknisk kringutrustning som t.ex. RPC, lagrade procedurer, objektorienterade databaser, Domaincontroller, server till server kommunikation och grafikservrar så bryts den strikta Client/Server modellen upp. Därför har en uppdelning gjorts i följande två delar.

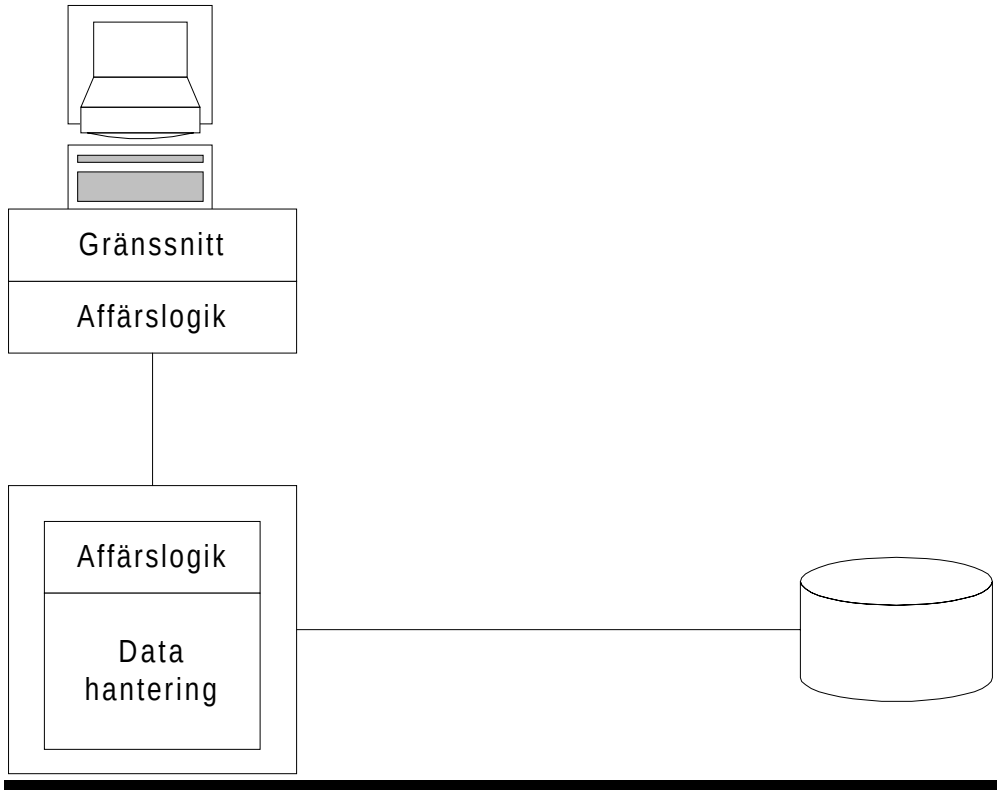
- Tvålagers arkitektur

- Trelagers arkitektur

7.2.1 Tvålagers arkitektur

Nedanstående material är hämtat ur (Renaud, 1996; Tittel och Stewart, 1997). Nyttjande av tvålagers arkitektur innebär att affärslogiken inte ligger som ett separat lager utan spänner över både server och klientdatorn. Tvålagers arkitektur lämpar sig främst på mindre nätverk då dess fördelar i enkelhet och billighet framkommer, samtidigt som miljön inte normalt blir för komplex för att denna ska kunna lösas med denna arkitektur. Dessa fördelar försvinner i större nätverk då komplexiteten samtidigt ökar och det blir svårt att administrera en sådan här arkitektur. Dessutom försvårar en tvålagers arkitektur möjligheten att skala upp på ett smidigt sätt.

Nedan finns en bild som beskriver hur en applikations olika delar är uppdelade i en tvålagers Client/Server arkitektur.



Figur 7 Tvålagers arkitektur

Fördelar Tvålagers arkitektur

- Endast klienter kan kommunicera, alltså ingen server till server kommunikation, vilket i sig gör saker mer komplexa att sköta.
- Billigt och enkelt, en tvålagers arkitektur innebär förmodligen ett litet nätverk och då behöver man inte lägga ut pengar på extra servrar och avancerade protokoll dessa emellan.
- Data lagras förmodligen centralt, en enda server för att hantera all data, vilket leder till lättare administration

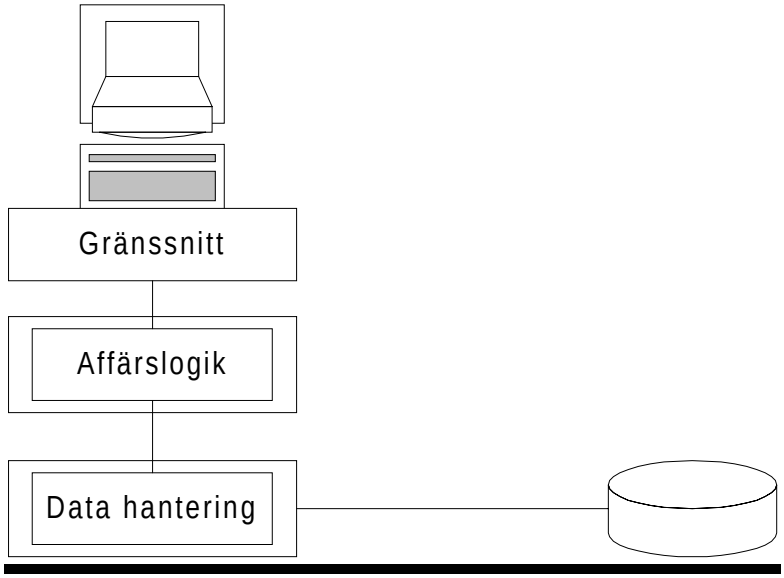
Nackdelar tvålagers arkitektur

- Klarar ej så många användare. Fungerar ej i större LAN, Passar ej för mer än 10 användare. (Tittel och Stewart, 1997)
- Kan ej fördela arbetsbelastningen på flera olika servrar, vilket i sig kanske inte är så stor nackdel i ett litet nätverk då arbetsbelastningen ändå inte lär vara så stor.
- Ej tacksam att skala upp då varje klient måste konfigureras för att anropa en server på ett riktigt sätt, kan finnas flera servrar t.ex. filserver, printerserver, faxserver osv. Varje klient måste konfigureras för varje existerande server.
- Sårbart, om t.ex. databasservern kraschar, finns det förmodligen ingen reserv.

7.2.2 Trelagers arkitektur

Den i tvålagers arkitekturen gemensamma del för en applikation, affärslogiken, isoleras till en egen separat maskin vilket gör den lättare att sköta. I och med att all affärslogik nu ligger på en separat maskin och inte ligger fördelat på varje applikation kan man införa ett extra lager servrar för att kunna dela på arbetsbördan. Denna uppdelning kan t.ex. göras genom replikering, då identisk data kan finnas på flera olika servrar för att minska belastningen eller genom något som kallas ”partionerad data” som innebär att olika servrar inte innehåller någon information som någon annan server gör och att det finns en extra server som känner till vilken data som finns på vilken server. Detta för att kunna avlasta belastningen på nätverket. Om en viss avdelning oftast använder sig av en viss data kan den läggas på en server som fysiskt finns nära deras klientdatorer. En trelager arkitektur innebär en mängd olika sätt att lägga upp sin databas på, alla med sina för respektive nackdelar vilket gör det lättare att special anpassa för just den specifika organisationens miljö. (Renaud, 1996)

Nedan finns en bild som beskriver hur en applikations olika delar är upp delade i en trelager Client/Server arkitektur. Fyrkanten runt affärslogik och datahanterings rutorna ska symbolisera att de är avskärmade från övriga delar och kan därmed även finnas fysiskt placerade på olika platser, t.ex. en specialserver för affärslogik och en speciell server för datahanteringen.



Figur 8 trelagers arkitektur

Fördelar trelagers arkitektur

Nedanstående material är hämtat ur (Renaud, 1996).

- Klarar att hantera fler användare och större LAN och även WAN.
- Högre grad av skalbarhet, klienterna konfigureras bara mot första lagret servrar.
- Servrar kan dela på frågebearbetningen genom t.ex. replikering.
- Möjliggör att distribuera ut data, genom t.ex. ”partionerad data” för att öka nätverkseffektiviteten och dataöverföringen.
- Större möjlighet att anpassa informationslokalisering efter verksamheten genom att t.ex. nyttja någon av följande datahantering.
    - Replikering
    - Partionering
    - Reorganisering

Nackdelar trelagers arkitektur

Nedanstående material är hämtat ur (Renaud, 1996; Tittel och Stewart).

- Medför server till serverkommunikation vilket är komplext.
- Knappast lönsamt i mindre LAN, några extra servrar kanske inte kan motiveras för t.ex. 5 användare.

- Replikering medför arbetskrävande uppdatering, då den måste ske på flera platser.
- Partionerad data medför att alla inblandade servrar måste fungera.
- Ju mer distribuerat systemet är desto svårare är det att administrera
- Svårt att skala upp, innebär ofta flera servrar som en ny klient kommer att göra anrop mot och alltså måste konfigureras för. Desssutom server till serverkommunikation som kräver extra inställningar.

Komplexiteten att sköta kommunikation mellan olika servrar kräver hjälpmedel. Ett sådant hjälpmedel är t.ex. en Domaincontroller.

Domaincontrollern är en server som sköter all kommunikation och vidarebefordring av tjänster till övriga servrar, dvs. alla anrop och förfrågningar går genom denna server. Detta resulterar i att klienterna endast behöver konfigureras för en server och konfigureringen för att kommunicera med övriga servrar görs på ett ställe, nämligen i Domaincontrollern. Detta tillsammans med att den annars så komplexa server till server kommunikationen nu kan döljas helt från klienten gör att ett system blir mycket enklare att skala upp än tidigare då antalet omkonfigureringar minskar väsentligt.

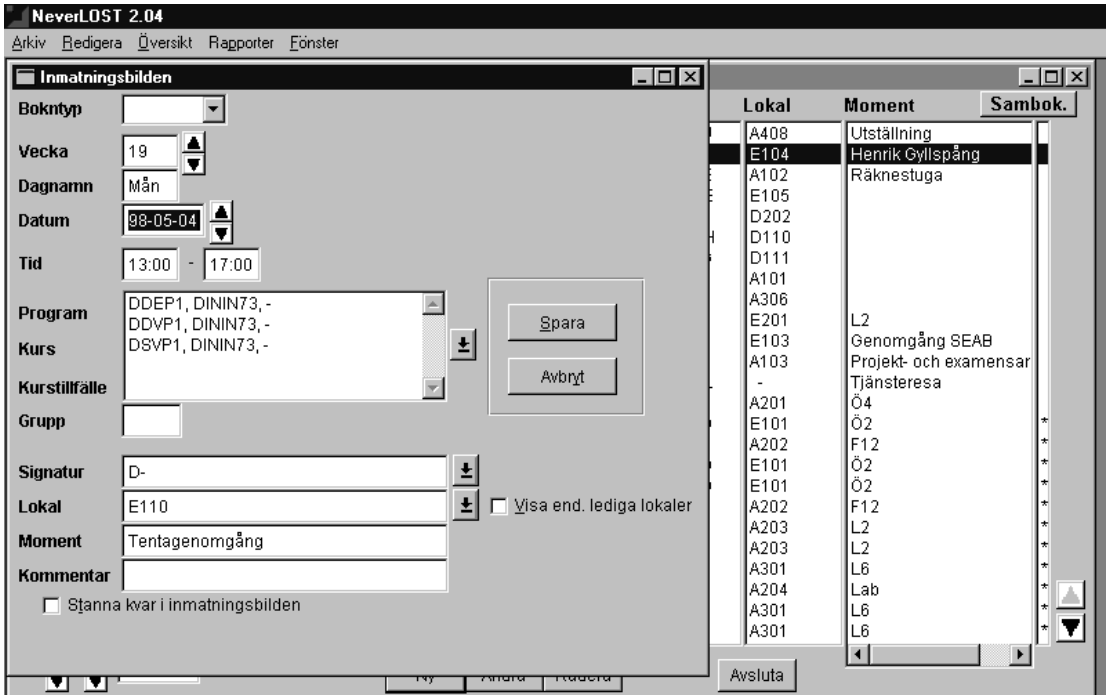
Det finns dock nackdelar med att nyttja sig av en sådan här lösning.

- Sårbart om Domaincontroller brakar ihop, då brakar allt ihop såvida ingen backup finns.
- Införskaffning av dyr programvara alternativt komplex konfigurering hos Domaincontroller för att vidarebefordringen till övriga servrar ska gå rätt till.

7.2.3 Exempel på en Client/Server lösning

Detta kapitel innehåller resultat från den teoretiska fallstudien, som nämndes i metod kapitlet.

Nedan finns en bild som är en skärmdump av Skövde Högskolas schemaläggningssystem. Detta program finns och hanteras inne på institutionen och har därmed automatiskt begränsad åtkomst från övriga på skolan system. Programmet innehåller så väl möjligheter till sökningar bland gamla reservationer som inmatning av nya. Bilden nedan visar bara en av många gränssnitt i programmet och visar hur gränssnittet ser ut vid registrering av salsbokningar, och då programmet innehåller flera olika sidor för olika funktioner ökar inlärningstiden på programmet men samtidigt förmodligen även funktionaliteten. På grund av att det i programmet finns möjlighet redigera i datan så krävs någon form av säkerhet så att inte vem som helst kan komma in och ställa till oreda. På skolan är detta löst genom att det endast finns på institutionens dator och därmed utan räckhåll för t.ex. studenter, men skapar problem om den personen är sjuk, då programmet bara finns på en plats. En sådan här lösning medför både att man slipper bry sig om säkerhetsaspekter som t.ex. lösenord samtidigt som utomstående måste få informationen angående reserverade salar via något annat media.



Figur 9 Bild på gränssnitt av exempelsystem

7.3 Statiskt Intranät

Att nyttja Intranät innebär att nyttja en verksamhetsintern användning av Internetteknik som protokollen TCP/IP och HTTP (Hyper Text Transfer Protocol).

Uppdelningen blir klient, som representeras av en webläsare samt webserver

Detta tankesätt medför att man omfördelar den traditionella synen på Client/Server där klienten ska ha hand om datapresentation och applikationshantering och där servern

hanterar dataåtkomst och kontroll, till att klienten nu endast ska ha hand om datapresentationen, kallas normalt även för en tunn klient. Detta att jämföra med stordator system med en kraftig central dator som är hopkopplade med en uppsättning "dumma" terminaler.

Klienten (dvs. webläsaren) klarar bara av att hantera två kommandon, get och post som ser till att ”önskad” websida hämtas och presenteras på skärmen. Användningsområden blir på det här viset något begränsade, och nyttjas framförallt för att presentera, samla in och söka bland information.

All verksamhet över nätverket sker m h a en webläsare, inklusive databashantering, vilket i sin tur medför att det blir enklare för användarna då de bara behöver hålla reda på ett gränssnitt. Nackdelen är dock att klienten inte har kapacitet att avlasta servern från databehandling. Det skript som finns tillgängligt i tidigare generationer av webläsare är nämligen bara CGI- skriptet, vilket endast kan köras på server maskinen.

En stor fördel dock med införandet av Intranät är att de för tillbaka den arkitekturella enkelheten till verksamheten. Den saliga blandningen av patentskyddade utvecklingsverktyg, gränssnitt, plattformar och nätverksprotokoll såväl som brist på standard för Client/Server lösningar har gjort denna arkitektur till en av de mest komplexa att utveckla och bygga applikationer till. (Linthicum, maj 1996)

Fördelar statiskt Intranät

Nedanstående material är hämtat ur (Bark, 1997; Black, 1996; Nordling, 1997; Tittel och Stewart, 1997).

- Intranät möjliggör en naturlig ingång i sitt interna Internet genom det globala Internet för kunder och leverantörer, detta brukar kallas för extranät. Denna ide att man vill släppa in utomstående in i sitt egna nätverk ställer höga krav på säkerhet, framförallt vilka som får komma in och vad de får göra där, vilket brukar benämnas för brandväggar.
- Intranät är "lätt" att lära sig, många har nyttjat Internet tidigare, möjligheten till hyperlänkar osv. dessutom i och med bristen på funktionalitet i klient programvaran så blir de lättare att lära.
- Lätt att skala upp, bara ange IP-adressen för nyinstallerad webserver, URL:er (Uniform Resource Locator) kan anges dynamiskt i efterhand i t.ex. hyperlänkar. En ny klient behöver bara installera en webläsare och tillförordnas en IP-adress, Internet är ett bra exempel.
- Förhållandevis smidig att administrera, även för många användare. Webläsaren blir en generell klient för alla sorters tillämpningar. (Nordling, 1997)
- Många funktioner kommer gratis, t.ex. E-post, IRC (Internet Relay Chat), nyhetsgrupper för att nämna några.
- Relativt billigt, generella produkter samt möjlighet att nyttja shareware och liknande innan man köper programvara.
- Plattforms oberoende, möjligt att t.ex. köra UNIX, Mac:ar, WindowsPC i samma Intranät miljö.
- I och med Intranätets plattforms oberoende så ökar möjligheten att återvinna befintlig utrustning.

- Intranätteknologin är baserat på öppna standarder. Man binder sig inte till ett programföretag. (Nordling, 1997)
- Inte speciellt prestandakrävande för klientmaskinerna. Klientens uppgift i ett Intranät är bara att presentera data och ställer därmed inga vidare krav på prestanda hos klient datorn som t.ex. stor hårddisk, kraftig processor, mycket minne osv. (Bark, 1997) Detta är dock teoretiskt då man får anta att en klientdator används till annat än bara kommunikation över nätet och då behöver klienten naturligtvis lite bättre kapacitet.
- Krångligheten med traditionell Client/Server system, med versionskontroll, distribution av mjukvara och system administrering försvinner. (Black, 1996)

Nackdelar statiskt Intranät

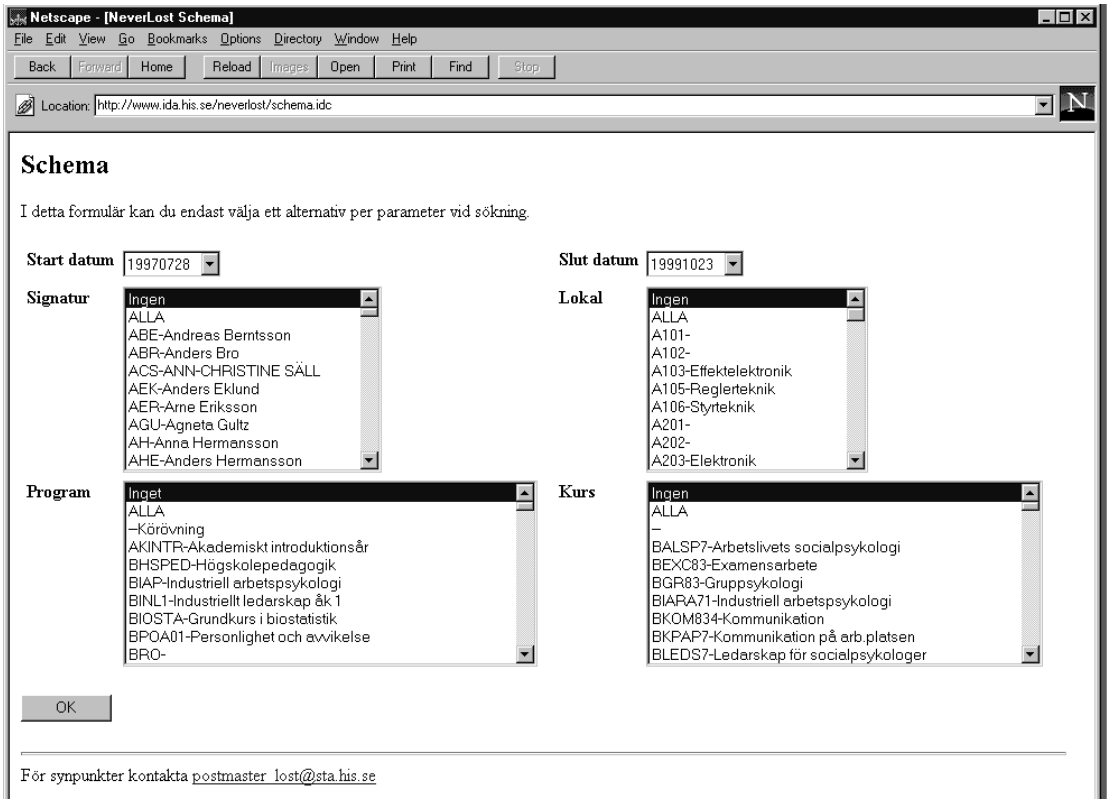
Nedanstående material är hämtat ur (Fenstermacher, 1997; Nordling, 1997; Renaud, 1996; Tittel och Stewart, 1997).

- En nackdel är att det kommunikationsprotokoll som Intranät är uppbyggt kring, TCP/IP är nätverks krävande, vilket innebär att bättre prestanda än normalt krävs
- En annan är att säkerheten är förhållandevis låg. komplexa åtkomstsinställningar (ställa in vilka IP-adresser som får tillgång till vad, istället för att man tillhör olika behörighetsgrupper), brandväggar mot Internet osv.
- Intranät når inte upp till samma funktionalitet som många existerande grupprogram exempel Lotus Notes och Microsoft Exchange. (Tittel och Stewart, 1997)
- All information som ska finnas i nätet måste först konverteras om till HTML format så att läsaren kan visa sidan.
- Servern kan lätt komma att belastas hårt då klienten endast kan visa data och inte har någon möjlighet till datahantering.
- Saknar möjlighet till olika databas strukturer så som t.ex. replikering
- HTML standarden är för löst ”specad” och klarar t.ex. inte av positionskontroll, vilket kan resultera i att objekten på en websida kan hamna lite huller om buller. Layout skiljer även från webläsare till webläsare, vilket alltså ger extrajobb för en webdesigner. (Fenstermacher, 1997)
- IP- adresser är relativt krångliga att administrera. (Nordling, 1997)

7.3.1 Exempel på statisk Intranätlösning

Detta kapitel innehåller resultat från den teoretiska fallstudien, som nämndes i metod kapitlet.

Bilden nedanför är en skärmdump av webvarianten av skolans schemaläggningssystem. Problemet med bilden nedan är att den liksom fallet normalt är i statiska websidor att det inte går att skriva in nya bokningar i systemet samt att den inte kan uppdateras dynamiskt, utan måste ladda hem en helt ny sida. Programmet nedan fungerar endast som en databassökare. Man anger sina sök kriterier i rullgardinslistorna och trycker sedan på ”Ok” knappen. Vad som sen händer är att informationen skickas över till servern som tar hand om databerarbetningen och skickar sedan tillbaka resultatet i form av en lista. En stor fördel dock är att som tidigare nämndes, det går inte att redigera i programmet och boka upp salar, vilket lätt hade kunnat skapa kaos då det här programmet finns tillgängligt för alla. Detta betyder alltså att man inte behöver bry sig om att lägga in rättigheter eller lösenord som förmodligen hade krävts om programmet hade varit i någon annan form än statisk HTML.



Figur 10 Bild över gränssnitt på exempelsystem i webversionen

7.4 Dynamiskt Intranät

Ett dynamiskt Intranät omfördelar synen på vem som skall göra vad gentemot annars i ett Intranät. Detta så att klient och server mer kommer att likna en vanlig Client/Server arkitektur än ett statiskt Intranät. Klienten blir ”fetare” i och med att t.ex. ActiveX komponenter möjliggör viss datahantering hos klienten lokalt. Dessutom kan även serversidan förstärkas i och med lanseringen av Active Server Pages som är en

komplettering till det tidigare skriptspråket på serversidan, nämligen CGI- skript. (Linthicum, Active Platform 1997)

Ett dynamiskt Intranät möjliggör skript och programkörning i webläsaren m h a ActiveX och Java, vilka å andra sidan i sin tur ställer högre krav på programvara. T.ex. ActiveX kräver Internet Explorer 3 eller senare version som i sin tur ställer vissa krav på prestanda och plattform. Java applets i sig ”slukar” dessutom en hel del processorkraft samtidigt som den vill ha Netscape 4 eller senare för att vilja vara med.

Dynamiskt Intranät med hjälp av ActiveX teknologi utökar möjligheterna i ett Intranät och man kan få tillbaka lite av den funktionalitet som man tidigare hade i traditionella Client/Server applikationer. Nackdelen dock är att t.ex. ActiveX vill ha webläsaren Internet Explorer 3 eller senare samt operativsystemen Windows 95 eller NT för att fungera. Detta medför alltså att man genom att nyttja ActiveX blir väldigt plattformsberoende. Flexibiliteten får offras då man vill ha lite högre komplexitet och funktionalitet.

Fördelar dynamiskt Intranät

Nedanstående material är hämtat ur (Fenstermacher, 1997; Karpinski, 1997; Linthicum, Active Platform 1997).

- Den tidigare omständliga processen med att konvertera om all data till HTML-format behövs ej längre då ActiveX klarar av att hantera flera vanliga format som bl.a. Excel och Word format. Detta åstadkommes med hjälp av Active Dokument som är en ActiveX teknologi. (Karpinski, 1997)
- Lättare att skaffa program (ActiveX komponenter) då dessa bara är att ladda ner från internet, många komponenter är även shareware eller freeware.
- Det finns en rad program som kan nyttjas för att utöka funktionaliteten i t.ex. webläsaren. Ett exempel är ”Adobe Acrobat Reader” som klarar av att visa avancerad utformning och ger möjlighet att visa formatet PDF som ej Active Dokument klarar av normalt. (Fenstermacher, 1997)
- Med nyttjande av redan färdig ActiveX komponenter blir det lättare att skapa snygg layout och multimedia på websidan. Sidan blir på det här sättet uppbyggt av objekt och man behöver inte som tidigare ladda hem en ny sida för att kunna göra en liten ändring utan laddar bara hem omritningen för respektive objekt vilket i sin tur leder till minskad nätverkstrafik och avlastning på servrar som inte behöver utföra CGI-skript för att räkna ut något. Detta kan ofta göras lokalt hos klienten m h a ActiveX skript. Detta medför att man t.ex. kan ha en klocka på sin websida som ritar om sig varje sekund, vilket annars skulle innebära stor nätverksbelastning om hela sidan var tvungen att skickas från servern eller t.ex. ständig uppdatering av börskurser.
- Möjlighet till lokal exekvering av program och kan alltså t.ex. kryptera ett lösenord lokalt i webläsaren innan man skickar ett meddelande över nätet.
- En ActiveX komponent kan se till att ett formulär är riktigt ifyllt istället för att som normalt låta servern upptäcka detta och skicka tillbaka sidan så att processen kan göra om, vilket leder till minskad nätverksbelastning.
- Möjliggör gruppverksamhet med hjälp av cookies och intelligenta serverskript som kan erhållas med t.ex. Active Server Pages som är ett senare alternativ till CGI skriptet.

Nackdelar dynamiskt Intranät

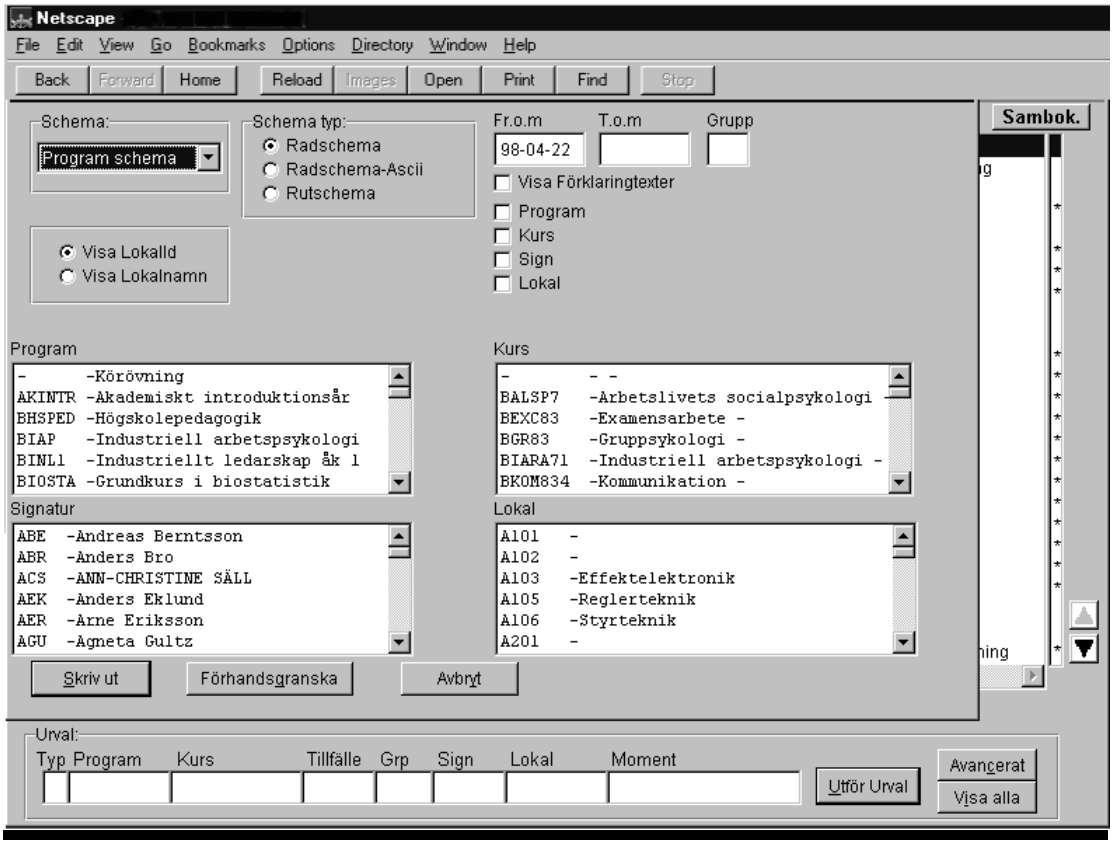
Nedanstående material är hämtat ur (Andersson, 1997, Linthicum, Active Platform 1997; Linthicum, ActiveX Security 1997; Patrizio, 1997).

- Nyttjande av ActiveX medför dock vissa säkerhetsaspekter. Man vet t.ex. inte vad ett nerladdat ActiveX program gör. Det kan t.ex. formatera om hårddisken eller något annat obehagligt. Microsoft som lanserat ActiveX litar istället på något som kallas för authenticode och på människans goda vilja att inte lansera "elaka" program. (Linthicum, ActiveX security 1997)
- Java applet som är ett alternativ till ActiveX tillåter visserligen ingen bearbetning mot maskinvara och då slipper man även AcitveXs säkerhetsproblem (se 2.2.5), men missar prestanda och effektivitet då en Java applets resultat inte kan lagras lokalt, utan måste exekveras över nätet varje gång den ska nyttjas.
- ActiveX fungerar i dag endast om klienten körs på en 32-bitars Windowsmiljö med hjälp av Internet Explorer 3 eller senare mot en Internet Information Server, vilket tenderar att begränsa Intranätets annars så stora plattformoberoende. (Patrizio, 1997)
- Internet Explorer 3 eller senare som behövs för möjliggöra ActiveX över nätet kräver viss prestanda på klientmaskinen för att det hela skall fungera smärtfritt.
- Om man vill använda sig av CGI skriptets komplement, Active Server Pages så krävs Windows NT.
- ActiveX komponenter laddas automatiskt ner på hårddisken för att öka effektivitet, vilket samtidigt kräver utrymme på en hårddisk. (Andersson, 1997)

7.4.1 Exempel på möjlig lösning i ett dynamiskt Intranät

Detta kapitel innehåller resultat från den teoretiska fallstudien, som nämndes i metod kapitlet.

På grund av existerande ActiveX komponenter som t.ex. knappar, inmatningsfält, ”checkboxar” kan man med ActiveX återskapa ett gränssnitt med väldigt likartade funktioner gentemot en vanlig Client/Server applikation. Dessutom i och med att ActiveX möjliggör lokal skriptexekvering och uppdatering av enskilda objekt istället för hela websidor, kan utseendet uppdateras dynamiskt beroende på vad som anges i t.ex. rullgardinslistorna eller andra alternativ som markeras. En nackdel är att ActiveX inte kan köras över t.ex. UNIX vilket innebär att en kompletterande lösning skulle behöva göras för att alla i nätverket (såvida det nu inte bara existerar Windows maskiner i nätverket) ska kunna nyttja applikationen.



Figur 11 bild över möjligt gränssnitt på exempelsystem i ett dynamiskt Intranät.

7.5 DCOM

Redan från början baserade sig OLE på två antaganden, enligt (Box, 1996), nämligen

- Objekt som finns i separata adressutrymmen med separata trådar för konton kan ibland behöva samarbeta för att kunna lösa vissa uppgifter. Detta resulterar i att de flesta program varken är rena klienter eller rena servrar, utan en hybrid av båda. De betraktas som klienter när de anropar "någon annans" objekt och som server förser med metodanrop å andras programs vägnar

- De flesta gränssnitts metoderna i COM är synkrom och har blockerings semantik i klienten. Programmet måste vänta på att metoden ska bli färdig innan den återvänder till proxyn. Eftersom att en klient även kan vara en server, särskilt en server som behövs av det ursprungliga metodanropet när den agerade som klient, behövs vissa tekniker för att tillåta återgång till klienten för att undvika deadlock.

I dagens operativsystem så är processerna avskärmade från varandra. En klient som vill anropa en komponent som finns i en annan process kan inte anropa denna komponent direkt, utan måste använda sig av någon form av processintern kommunikation som tillhandahålls av operativsystemet. Denna kommunikation möjliggör COM på ett totalt osynligt sätt. COM översätter anropet från klienten och vidarebefordra det till komponenten i den andra processen. Om nu klienten och den eftersökta komponenten skulle råka befinna sig på två olika maskiner så ersätter DCOM den lokala processinterna kommunikationen med ett nätverksprotokoll (ORPC) och varken klienten eller komponent är medvetna om att kopplingen dem emellan är lite längre än vanligt.

DCOM protokollet, känt som ORPC är en uppsättning definitioner som förlänger standard protokollet, (DCE RPC) och det framförallt inom två särskilda områden, enligt (Horstman och Kirtland, 1997), nämligen.

- Hur anrop mot avlägsna objekt görs.
- Hur objektreferenser representeras, skickas och underhålls.

ORPC protokollet har designats särskilt för DCOMs objektorienterade miljö och specificerar hur anrop görs över nätverket och hur objektets referenser representeras och underhålls.

Medan COM är en specifikation för att bygga "interoperable" komponenter så är DCOM helt enkelt ett högnivånätverksprotokoll designat att möjliggöra COM-baserade komponenter att samverka över ett nätverk. DCOM väljer automatiskt det bästa underliggande nätverksprotokollet baserat på de nätverksprotokoll som finns tillgängliga på klient och server maskinerna. (Eddon och Eddon, 1998)

I normala fall när man börjar implementera en distribuerad applikation i ett nätverk måste man uppmärksamma flera designaspekter: (MSDN, 1998)

- Komponenter som samverkar ofta bör placeras nära varandra.
- Vissa komponenter kan bara köras på vissa maskiner eller på vissa specifika platser.
- Mindre komponenter ökar flexibilitet vid utveckling men även trafiken över nätverket.
- Större komponenter minskar trafiken över nätet men minskar även flexibiliteten vid utveckling.

Med DCOM är dessa kritiska designproblem lätta att komma runt, då utvecklingsdetaljer inte är specificerade i källkoden. Detta gör att det med DCOM är enkelt att ändra hur klienten är kopplad till komponenter och hur komponenter är kopplade till varandra. Dessa komponenter kan dynamiskt bli förflyttade utan omarbetning eller omkompilering. Det enda som behövs är att uppdatera "Windows registret", filsystemet eller databasen där varje komponent finns. (MSDN, 1998)

DCOM gör det lätt att skriva distribuerade applikationer som (MSDN, 1998)

- Skalar, från minsta endatormiljö till största poolmiljö av servrar
- Ger rik, symmetrisk kommunikation mellan komponenter
- Kan robust expanderas för att möta nya funktionella krav
- Tar fördel av existerande färdigprogram (COM och ActiveX)
- Har medfödd säkerhet
- Kan effektivt utvecklas och administreras
- Kan nyttjas med vilket protokoll och integreras i vilken plattform som helst

Enkelhet, den stora spridningen och standardInternetprotokoll som HTTP, gör Internet till en ideal teknologi med att länka komponenter som ej finns på samma maskin. HTTP är enkelt att programmera, har naturligt stöd för flera plattformar, tillgänglighet och universell namngivning. DCOM möjliggör dessa applikationskomponenter att exekvera över Internet, vilket gör DCOM är idealisk för att bli en Internet teknologi då, enligt (Microsoft Corporation, 1997), DCOM:

- Är transport neutral: DCOM tillåter komponent kommunikation med hjälp av t.ex. TCP/IP, UDP/IP, IPX/SPX, Appletalk och HTTP.
- Har medfött stöd för Internet protokoll och nyttjar ActiveX teknologin.
- Möjliggör distribuerad Java: Då DCOM är språkoberoende kan Java applets liksom ActiveX komponenter kommunicera direkt över Internet.
- Är evolutionär teknologi: DCOM stödjer även komponenter skrivna i andra språk, t.ex. C, COBOL, BASIC och PASCAL att kommunicera över Internet.
- Stödjer gemensamma komponenter för webläsare och webserver: Då ActiveX komponenter kan bäddas in i webläse baserade applikationer möjliggör DCOM en rik applikations infrastruktur för distribuerade Internet applikationer genom att använda den senaste webläse teknologin.

DCOM är en uppsättning Microsoft koncept och program gränssnitt där vilket klient programobjekt kan anropa tjänster från vilket server programobjekt som helst hos en annan dator (eller samma) i nätverket. Till exempel kan man skapa en websida som innehåller ett skript eller program som kan exekveras, inte på websidans server utan på en annan mera specialiserad server i nätverket. Vid nyttjande av DCOM gränssnitt så kan en websidas webserverprogram skicka vidare ett RPC till ett specialiserat serverobjekt som utför den nödvändiga databehandlingen och skickar resultatet till websidan.

DCOM i sig är inte så underbar. Många hävdar (Roger Sessions) att t.ex. CORBA slår DCOM gällande distributionsbiten, men just DCOMs förmåga att skapa en helhetsvision med återvinningsbara komponenter med hjälp av COM och ActiveX, lokaliseringsfrihet m h a COM och även då DCOM har möjlighet att hitta rätt komponent, distribution över Intranät och Internet m h a TCP/IP, UDP/IP, IPX/SPX, Appletalk och HTTP. DCOM Stödjer interaktivitet över nätet genom att stödja ActiveX komponenter och skript och även Java applets och skript. Dessa aspekter tillsammans hjälper till att höja DCOM. (Sessions, 1998)

Fördelar DCOM

Nedanstående material är hämtat ur (Brockschmidt, juni 1996; Sessions, 1998).

- DCOM integrerar Internet certifikatsbaserad säkerhet med duktig NT baserad säkerhet och kombinerar de bästa av två världar.
- Då DCOM har vuxit fram ur COM, som är den världsledande komponent teknologi idag, så kan existerande investeringar i COM baserade applikationer, komponenter, verktyg och kunskap nyttjas för att ta steget in i den standardbaserade distributionsdatoriseringen.
- Effektiv versionshantering. Med COM och DCOM kan klienten dynamiskt fråga om komponentens funktionalitet. I stället för att visa sin funktionalitet som en enda grupp av metoder och egenskaper kan en COM komponent te sig olika för olika klienter. Om nya drag adderas till en komponent så påverkar dessa inte en äldre klient som ej känner till detta. (Brockschmidt, juni 1996)
- DCOMs oberoende av lokalisation gör det enkelt att distribuera komponenter till datorer och erbjuder på så vis ett enklare och smidigare skalbarhet.
- Underlättar design problem genom bl.a. lokalisationsoberoende av komponenter
- Mycket medfött stöd för att nyttja Internet teknologi och stödjer distribuerade komponenter över Internet.

Nackdelar DCOM

Nedanstående material är hämtat ur (Sessions, 1998).

- Konceptet är nytt, ej vältestat och utprovat.
- DCOM lösningen inte mycket lönt utan Microsoft Transaction Server. (Sessions, 1998)
- Fungerar visserligen på andra plattformar än Windows, och ska stödja fler inom sinom tid, men övriga koncept som bör användas tillsammans med DCOM för att skapa något mer än bara ett protokoll är avsedda för Windowsmiljö (ActiveX, COM o.s.v.). En annan aspekt är att DCOM är framför allt avsedd för NT plattformar, finns visserligen för t.ex. Windows 95 också men då i och med att mycket av DCOMs säkerhet baserar sig på NTs säkerhet, vilket bl.a. Windows 95 saknar mycket av.
- COM objekt är lite krångligare att skapa än t.ex. C++ klasser, men ska underlättas i och med lanseringen av COM+.
- Dyrt att införa DCOM, behöver så mycket specifik utrustning, NT servrar, Microsoft Transaction Server osv. Kan t.ex. inte återanvända gammal utrustning om den inte stämmer in i Microsofts vision.
- Mycket mera komplext än t.ex. ren Client/Server eller Intranät då klienten känner till en hel del om server, detta för att kunna hitta objekten rätt. Detta gör det svårare och komplexare att t.ex. bygga och konfigurera om .

- DCOM är en Microsoft produkt och Microsoft stödjer ej stordator tänkandet vilket ofta har varit lösningen för avancerade databaser. Microsoft i och med NT server lanserar dock ett alternativ i form av något som kallas för ”clustering”, vilket innebär att flera mindre kraftfulla datorer ska kunna fungera ihop och på så vis kunna matcha stordatorers kapacitet. Clustering är dock inte är så välutvecklat ännu, klarar för tillfället bara två servrar som samverkar, men utlovar att flera ska kunna samverka inom kort.

7.5.1 Exempel på möjlig DCOM lösning

Detta kapitel innehåller resultat från den teoretiska fallstudien, som nämndes i metod kapitlet.

I detta fall är det inte utseendet som behöver påverkas, utan istället möjligheten att distribuera ut komponenterna över nätverket och funktionalitet som därmed följer. Till exempel i och med att ett DCOM objekt kan anropas kan det på så sätt köras av ett annat objekt. Detta innebär möjlighet att skapa objekt på klient sidan som kan kommunicera med objekt på servern. Klient objektet kan läsa över information i realtid, direkt från server objektet som kanske har en permanent koppling mot en databas. En annan fördel är att distribuerade applikationer är mycket stöd för att skalas. Andra fördelar som t.ex. lokalisationsoberoende, återanvändning av komponenter (om applikationerna är uppbyggda kring COM eller ActiveX komponenter), möjlighet till att hantera olika personer med olika rättigheter genom COM och DCOMs förmåga att visa upp olika funktioner och egenskaper för olika klienter, kan också nyttjas med fördel.

8 Analys

Client/Server har funnits länge. Tekniken började växa fram under mitten av 80- talet vilket innebär att konceptet har haft tid att mogna, noggrant testats samt att det finns mycket programvarualternativ speciellt för denna miljö.

Intranät är även den en väl beprövad metod, då Internet som kommer från amerikanska ARPANET som växte fram under slutet av 1960- talet och började under 1980- talet få fotfäste för civila tillämpningar. WWW och Intranät är dock nyare.

ActiveX som mycket ligger bakom ett dynamiskt Intranät eller Internet har visserligen inte så lång tid bakom sig då termen lanserades först i början av 1996. ActiveX har dock i praktiken haft längre tid för att mogna då det framväxt och förfinats rakt ur OLE tekniken. Det finns idag många existerande ActiveX komponenter som direkt fungerar efter nerladdning. Nackdelen är dock att teknologin inte hunnit testas och mogna lika länge som t.ex. Client/Server eller ett statiskt Intranät.

DCOM bygger som sagt på COM tekniken och förlänger bara den teknikens funktion till att fungera över ett nätverk. Problemet är att helheten DCOM tillsammans med ActiveX, COM- objekt osv. inte har haft tid att testas och utvärderas, vilket gör att det blir mycket spekulationer angående denna arkitektur.

Som sagt var och en av de olika arkitekturerna har sina för respektive nackdelar, nedan följer en liten jämförelse arkitekturerna emellan.

8.1 C/S kontra Statiskt Intranät

Det går att göra mera avancerande applikationer i en Client/Server miljö. Intranät är framförallt till för att visa och distribuera information.

Client/Server arkitekturen har stor flexibilitet, allt från ett litet tvålagers nätverk bestående av en databasserver och ett fåtal användare som blir billigt och smidigt att installera och sköta, då valfritt protokoll kan användas o.s.v. (Intranät kräver TCP/IP och HTTP) till att ha en avancerad flerlagers arkitektur med komplicerad server till server kommunikation och avancerade specialprogram och hårdvara.

I och med att datahantering sker hos klienten måste även varje klient utrustas med hyfsad prestanda. Detta kan lätt bli onödigt dyrt om nätverket inte används till mycket annat än att distribuera information. Ett Intranät dock som har all sin datahantering förlagd på server behöver bara lägga ner krut på servermaskinerna och kan på så sätt spara pengar, såvida användarna nu inte ändå kräver kraftfulla datorer i andra syften.

Ett Intranät är relativt lätt för ovana datoranvändare att lära sig. Dels är gränssnittet detsamma (i alla fall funktionalitet och menyval) för alla nätverksapplikationer dels har allt fler personer testat att kopplat upp sig mot Internet och tekniken är densamma. Samtidigt så finns väldigt mycket av ”look a like” känsla i de flesta program i dag (för att man ska känna igen sig). Ett bra exempel är alla Windows program.

Ett Intranät är plattformsberoende och låter flera olika maskiner (Mac:ar, UNIX, PC) att fungera tillsammans över nätverket.

Sammanfattning: En Client/Server verksamhet är att föredra i de flesta fall, såvida inte det enda syftet med nätverkskommunikationen är att kunna söka upp och presentera data.

8.2 C/S kontra Dynamiskt Intranät

Fortfarande går det inte att göra lika avancerade applikationer i dynamiskt intranät som i en Client/Server miljö.

ActiveX innebär återanvändningsbara program och priset behöver då inte bli så högt vid köp. Dessutom finns det gott om freeware bland ActiveX komponenter.

Dynamiskt Intranät med stöd av ActiveX är bra vid öppnande mot Internet, möjliggör interaktivitet över nätet. Ett exempel är betalning över nätet med bl.a. Sparbanken och Nordbanken.

ActiveX begränsar den annars så fina tanken med plattforms oberoende, som fallet är i ett statiskt Intranät. Detta då i dagsläget endast Windows 95 och NT stödjer ActiveX.

Sammanfattning: Kan vara en svår avvägning, det finns mycket beprövad programvara för en Client/Server miljö men allt mer programvara görs för olika lösningar i ett dynamiskt Intranät. Ett dynamiskt Intranät kan dock vara en fördel om datavanan hos de anställda inte är så stor, då gränssnittet i en webläsare är det samma för alla nätverksapplikationer, en nackdel är att nyttjande av ActiveX i det dynamiska Intranätet gör att man låser sig för Microsofts lösningar.

8.3 C/S kontra DCOM

Client/Server är inte lika komplext att administrera, då klienten inte behöver ha någon vetskap om hur servern fungerar.

DCOM bygger naturligt på Internet teknologin likväl som återanvändning av komponenter som COM och distribution av komponenter över nätverk med ActiveX.

DCOM är förmodligen lite överdrivet i ett litet nätverk, då mycket utrustning och programvara krävs även vid ett litet nätverk.

Client/Server har större möjlighet att återanvända gammal utrustning, då det finns protokoll för de flesta sorters olika plattformarna. DCOM dock fungerar i dag endast på Windows NT och Windows 95.

Sammanfattning: DCOM konceptet är helt klart intressant, men det har inte haft någon tid att mogna och utprovas som Client/Server miljön har. DCOM kräver visserligen en större insats i kapital men kan gott betala tillbaka sig vid ett normalt stort nätverk, då det finns bra stöd för t.ex. skalbarhet och konfigurering.

8.4 Statiskt kontra Dynamiskt Intranät

I ett statiskt Intranät råder fortfarande plattforms oberoende medan ActiveX kräver Windows95 eller NT samt Internet Explorer 3 eller senare för att fungera.

Nätverk och server arbetsbelastning kan minskas då ActiveX möjliggör viss datahantering lokalt hos klienten.

Även om Intranätet endast används som informationsmedie så kräver en klient som nyttjar ActiveX hårddiskutrymme då ActiveX komponenterna laddas ner och körs lokalt i klientmaskinen.

ActiveX underlättar markant skapandet av websidor då det finns många komponenter som är redo att bara klistras in i en websida, dessutom slipper man att konvertera all data till HTML- format då Active Dokument som är en ActiveX teknologi stödjer flera vanliga format, däribland Word och Excel.

Sammanfattning: Såvida man nu inte är rädd för att förlita sig på Microsoft som leverantör så verkar fördelarna med ett dynamiskt Intranät tala för sig själv.

8.5 Statiskt Intranät kontra DCOM

DCOM kräver Windows 95/NT för att fungera vilket inte ger någon stor valfrihet, detta gentemot ett statiskt Intranäts plattformoberoende.

DCOM möjliggör mycket mer komplicerade applikationer över nätverket.

DCOM är förmodligen mycket dyrare att installera då det som sagt kräver maskin- och mjukvara av specifik sort medan man i ett statiskt Intranät förmodligen kan återanvända mycket hårdvaruutrustning.

DCOM har naturligt stöd för COM och ActiveX likväl som stöd för TCP/IP protokoll, dessutom kan DCOM nyttja säkerheten från både NT och Intranät.

Sammanfattning: Kräver verksamheten ett avancerat nätverk med avancerade nätverksapplikationer är DCOM att föredra, men i och med DCOMs ”start avgift”, gällande kostnad och komplexitet bör det läggas under noga övervägande om inte ett statiskt Intranät räcker.

8.6 Dynamiskt Intranät kontra DCOM

DCOM möjliggör med avancerade applikationer samtidigt som båda kräver Windows miljö, visserligen klarar sig ActiveX i Windows 95 också, men det skall även gälla för DCOM nu också.

Kostnaden, det blir förmodligen mycket billigare att satsa på ett dynamiskt Intranät, såvida verksamheten nu inte kräver allt för avancerade nätverksapplikationer.

DCOM bygger både på COM liksom på Intranät/Internet protokollen vilket borde göra att mycket kan återanvändas vid en uppskalning från dynamiskt Intranät till en DCOM arkitektur.

Intranät med stöd av ActiveX är mycket mera uttestad och vanligt förekommande på marknaden än vad DCOM är.

Sammanfattning: Fråga om krav på komplexitet och på om man vågar satsa stenhårt på en inte allt för väl testad och granskad arkitektur.

9 Diskussion

Arkitekturlösningen dynamiskt Intranät är granskat ur en synvinkel där ActiveX nyttjas. Ett Intranät baserat på Java teknologin skulle alltså inte ha exakt samma styrkor och svagheter som angivits i den här rapporten. De stora huvuddragen är dock de samma då båda lösningarna kom till av samma syfte. Jämförelser dessa två lösningar emellan för att skapa ett dynamiskt Intranät finns dessutom redan i mängder. En annan anledning till att jag tittade på ActiveX lösningen och inte Java ligger i att DCOM bygger på samma principer som ActiveX, nämligen COM.

Undersöker på lösning av distribuerade komponenter genom DCOMs synvinkel. Andra lösningar finns som t.ex. CORBA men liksom i fallet ovan med dynamiskt Intranät finns det redan många jämförelser dessa två lösningar emellan. Visserligen är både DCOM och CORBA en specifikation för implementering, men DCOMs specifikation är mycket hårdare och varierar följaktligen inte lika mycket som en CORBA lösning kan göra.

I och med denna undersöknings förhållandevis grunda och breda karaktär bör den kunna nyttjas som ett första steg vid val av en lämplig arkitektur att bygga sitt system och sina applikationer kring. Detta första val ska syfta till att begränsa antalet lösningar som behöver undersökas närmare t.ex. om första valet resulterar i att man vill nyttja dynamiskt Intranät så kan man direkt gå in på nästa undersökning vilken kan resultera i om man vill nyttja ActiveX eller Java teknologi osv.

Nedan följer en uppspaltning av olika faktorer och hur de olika arkitekturerna förhåller sig till de olika faktorerna, därefter finns även en graf som beskriver hur de olika arkitekturerna förhåller sig till varandra, dvs. vilken arkitektur som är bäst inom respektive område. Det är uppenbart att det förmodligen inte blir någon rättvis jämförelse om man bara räknar antal poäng, men har ändå undvikit att försöka vikta olika kategorier då ändå verksamheten och den speciella situationen avgör vilken faktor som är den viktigaste för just det specifika fallet. Dock finns det inget som säger att man inte kan vikta de olika faktorerna själv i efterhand för att på så sätt få fram den mest passande lösningen.

Kostnad:

- DCOM dyrast, kräver speciell plattform, NT eller Windows 95 maskiner samt Microsoft Transaction Server, för att underlätta administeringen.
- Client/Server har billigare hårdvara än både DCOM och dynamiskt Intranät med ActiveX då Client/Server inte kräver specifik plattform. Å andra sidan är mjukvara mycket dyrare till Client/Server än till DCOM och ActiveX, då dels DCOM och ActiveX möjliggör återanvändning av mjukvara och dessutom finns som ”freeware” för många program.
- Ett statiskt Intranät är förmodligen det billigaste alternativet då dels olika sorters plattformar kan nyttjas tillsammans i och med protokollen plattforms oberoende, dessutom behöver klienterna i ett statiskt Intranät inte vara så kraftiga då all databehandling sker på servern.

Kostnad för underhåll:

- Client/Server är dyrast och krångligast, finns ofta många speciella gränssnitt och konfigureringar i nätverket, Client/Server har inget naturligt stöd för distribution av

mjukvara eller versions hantering. Dessutom omständligt att uppdatera informationen då den oftast finns på olika platser rent fysiskt.

- En DCOM arkitektur består av flera komponenter, som i sig kan vara felorsaker men å andra sidan är där för att underlätta hantering av systemet.
- Dynamiskt Intranät med hjälp av ActiveX kan innebär att en isolerad testmaskin behövs, så att ej skadliga komponenter laddas ner och körs lokalt i det interna Intranätet.
- Ändring av websidor måste ske i specialprogram mot servern, vilket kan ta tid då HTML- språket är så löst specat och bl.a. har problem med positionshantering, så att en sida kan se olika ut beroende på vilken bläddrare som nyttjas. Dessutom innebär nyttjande av ett statiskt Intranät att all information måste omarbetas till HTML- format.

Användarvänlighet:

- I en Client/Server arkitektur har programmen olika gränssnitt, vilket gör att det blir svårare att lära sig all del olika gränssnitten, att notera dock är att man försöker skapa ”look alike” känsla mellan olika program. Ett bra exempel är alla Windows program, samtidigt finns förmodligen inbyggt stöd via F1 knappen samt möjlighet till kund support då programmen förmodligen har en återförsäljare.
- I ett statiskt Intranät är gränssnitt generellt för alla program, då gällande funktionalitet och menyval men kan variera i utseende beroende på vem som designat websidan, saknar dock inbyggd hjälp via F1-knappen och möjligheten till kund support då bläddrare kan laddas ner gratis eller till en liten kostnad över nätet. Dessutom i och med Intranät applikationers enkla komplexitet blir programmen ”lätta” att lära sig (i och med att det inte finns så många olika funktioner som man måste lära sig att hantera)
- I ett dynamiskt Intranät är också gränssnittet generellt och kund support saknas, på samma sätt som i ett statiskt Intranät. Dock kan ett skript på websidan skicka iväg begäran på en hjälpfil eller något liknande.
- I DCOM är förhållandet som för dynamiskt Intranät med ActiveX, vid nyttjande av webläsare.

Lämplig nätverksstorlek och Skalbarhet:

- Ett statiskt Intranät passar bra i de flesta storlekar, skalar bra från väldigt litet LAN till enormt WAN (Internet).
- Ett dynamiskt Intranät passar liksom ett statiskt Intranät i de flesta storlekar, att observera dock är att p.g.a. specifika maskin varukrav kan det bli förhållande vis dyrt i ett väldigt litet nätverk.
- Client/Server finns i olika varianter. En 2- lagers arkitektur kan vara ett väldigt billigt alternativ i ett litet nätverk, skalar dock inte så bra. En 3- lagers arkitektur passar för lite större nätverk och skalar bättre än ett 2- lagers Client/Server.
- DCOM skalar bra från väldigt litet till väldigt stort nätverk, nackdelen är att det blir en förhållandevis dyr lösning i ett väldigt litet nätverk.

Syfte:

- Statiskt Intranät nyttjas framför allt till att sprida och söka bland information.
- Client/Server ser till att arbetsbelastningen delas mellan server och klient.
- Dynamiskt Intranät nyttjas dels till att sprida och söka information men även t.ex. samla in information.
- DCOM se till att arbetsbelastningen sker på det effektivaste stället.

Effektivitet

- DCOM låter applikationerna genom sin distribution köras där de ger mest nytta och kan på sätt minska nätverksbelastningen. Nackdelen är att DCOM behöver ett sätt att få reda på när en komponent inte nyttjas längre och därmed kan ”dödas”, vilket kräver extra kommunikation över nätverket. (Nyttjar visserligen något som kallas för delta ping, vilket ska vara relativt ”intelligent”)
- I en Client/Server miljö delar klienten och servern på arbetet. På så vis slipper servern bli överbelastad och klienten behöver inte vara extremt kraftfull för att klara av uppgiften. Nätverksbelastningen kan variera en del beroende på vilken form av protokoll som nyttjas. Dessutom nyttjas oftast ”flergränssnitts applikationer” vilket betyder att man kan växla vyer och sidor utan att behöva ladda sidorna över nätet.
- I ett statiskt Intranät sker all databearbetning hos servern och klienten tar bara hand om presentationen. Vid förändring på en sida måste hela sidan laddas över på nytt även om bara en liten detalj ska ritas om eller ändras. TCP/IP protokollet i sig är dessutom förhållandevis nätverksbelastande.
- I ett dynamiskt Intranät kan liksom i Client/Server arbetet fördelas mellan klienten och servern. En annan fördel gentemot ett statiskt Intranät är dessutom att endast den delen (objektet) på en sida behöver laddas hem vid en förändring på sidan vilket kan hjälpa till att minska nätverksbelastningen, dock nyttjas TCP/IP protokollet som är förhållandevis nätverksbelastande.

Stöd för flera plattformar:

- I Client/Server beror det till stor del på vilka protokoll som nyttjas, ofta används dock patenterade protokoll vilket minskar möjligheten att köra på flera plattformar.
- I statiskt Intranät råder stort stöd för körning på flera plattformar då protokollen som nyttjas är öppna.
- I dynamiskt Intranät nyttjas visserligen samma protokoll som i ett statiskt Intranät men vissa hårdvarukrav finns, då ActiveX kräver körning i 32- bitars Windows miljö.
- I DCOM finns ännu större stöd för olika protokoll, men här liksom i dynamiskt Intranät återfinns hårdvarukraven.

Mjukvaru utbud

- Client/Server är väletablerad och det finns mycket avancerad mjukvara till denna arkitektur.

- I ett statiskt Intranät medföljer mycket programvara i och med webläsaren som t.ex. E- post, FTP osv.
- I ett dynamiskt Intranät finns det en hel del mjukvara genom alla nerladdningsbara ActiveX komponenter samt den typen av program som medföljer webläsaren.
- För DCOM gäller samma sak som för ett dynamiskt Intranät och kan dessutom ta nytta av existerande COM investeringar.

Applikations utvecklings stöd.

- Till Client/Server finns mycket väl utprovade utvecklings verktyg. Samtidigt finns det dock en sådan uppsjö att ett riktigt utvecklingsverktyg kan vara svårt att hitta.
- Till Intranät fungerar inte de traditionella utvecklingsverktygen detta då en webapplikation skiljer sig fundamentalt från traditionell Client/Server applikation.
- För DCOM applikationsutvecklare skiljer sig inte utvecklingen från utveckling av COM applikationer, detta då DCOM är en evolution ur COM och fungerar enligt samma principer.

Nedan finns en sammanställning och värdering av de olika arkitekturerna, baserat på det material som finns i punkterna i kapitel 9.

Underhåll	4	3	1	2
Användar vänlighet				
Hjälpfunktioner och online	1	4	3	3
Inlärnings tid	4	1	2	2
Storlek och skalbarhet				
Litet	1	2	3	4
Stort	4	1	1	2
Skalbarhet	4	1	1	2
Ändamål				
Informationsspridning	3	2	1	1
Affärs applikationer	3	4	1	1
Komplexa applikationer	1	4	2	1
Effektivitet				
Arbets fördelning	2	4	2	1
Nätverks belastning	3	4	2	1
Plattformar				
Stödjer flera olika i samma nätverk	3	1	4	4
Flera olika typer i olika nätverk	2	1	3	3
Mjukvara				
Existerande utbud	1	3	2	1
Kostnad utveckling	4	3	2	1
Tidsåtgång utveckling	4	3	2	1
Tillgänglighet verktyg	1	2	2	1
Beprövad				
Erfarenhet (versionsnummer)	1	2	3	4

9.1 Slutsats

Samtliga av de undersökta arkitekturerna har sina för respektive nackdelar och det är upp till varje specifik situation att se efter ”vad man vill ha” och vad man anser vara viktigast. Men kort att säga om respektive arkitektur. Statiskt Intranät bör framförallt endast användas som distributions verktyg för information mellan användarna. Client/Servers stora fördel ligger i att det är väletablerat, men i takt med allt fler varianter försvinner även denna fördel, nackdelen ligger i bristen på användarvänlighet för ovana datoranvändare och komplexitet i att administrera. Dynamiska Intranät lösningar fördelar ligger i att det möjliggör exekvering av avancerade program i det användarvänliga gränssnittet ”webläsaren” och därmed även över t.ex. Internet. Nackdelen ligger framförallt i begränsningar gällande fungerande plattformar som normalt nyttjandet av TCP/IP protokollet tar bort. DCOM stödjer både ActiveX och

bl.a. TCP/IP protokollet och ger då samma för och nackdelar som ett dynamiskt Intranät, detta vid nyttjande av en webläsare som nätverksprogram. Dessutom i och med DCOM distribuerade karaktär kan effektiviteten ökas då komponenterna kan köras där de gör bäst nytta.

9.2 Fortsatt arbete

Detta arbete skulle mer eller mindre kunna göras igen, fast då med skillnaden att fler eller andra arkitekturer skulle undersökas. Detta då det här arbete begränsats till att endast undersöka fyra olika arkitekturer.

Ett annat arbete skulle kunna vara att göra en motsvarande undersökning fast i intervjuform, detta för att få reda på hur en selektionsfas går till i dag, vilka kriterier som är viktiga samt att få olika typer av personers (systemutvecklare, programmerare osv.) varierade syn på vad som är funktionibelt och genomförbart.

Ytterligare ett arbete skulle kunna gå ut på att verkligen implementera en applikation i de olika arkitekturerna för att svart på vitt se vad som är möjligt och inte samt vilka problem som uppstår vid respektive lösning.

9.3 Erfarenheter

Det har varit tungt att göra litteraturstudier, dels blir det mycket material att läsa samt att det finns motsättningar mellan vissa material, dels beroende på ålder och dels beroende på källa. Hade nog varit nyttigt att kombinera denna metod tillsammans med intervjuer, detta för att på ett tidigare stadie få reda på intressanta kriterier att jämföra de olika arkitekturerna emellan. En del avgränsningar saknades t.ex. skulle en tidsavgränsning gjorts, detta då för bl.a. ActiveX och DCOM lite äldre material väldigt snabbt blev inaktuell. Det var även svårt att jämföra ett så luddigt och omfattande begrepp som Client/Server, detta dels då övriga undersökta arkitekturer i någon form kan ses som en utveckling av just Client/Server. Känner i efterhand att mer förkunskap skulle kunnat ge en bättre problem avgränsning och en lättare undersökning.

Mycket material har fått tas med en nypa salt, då i stort sett samtliga författare som har skrivit om en ny teknologi (arkitektur) har tenderat att skönmåla den nya lösningen. Det har dock inte varit så svårt att få en någorlunda neutral syn, då nästkommande teknologi har jämförts mot de tidigare och därmed har de föregående teknologierna fått både fördelar och nackdelar beskrivna. Exempelvis så höjdes Client/Server lösningar till skyarna i jämförelse med stordator miljöer, men sen fick Client/Server lösningarna en del kritik i boken om Intranät som då höjdes, osv. Dock så har det inte funnits någon arkitektur efter DCOM, hoppas ändå fått med en rättvis beskrivning.

I och med att det insamlade materialet är byggt på litteraturstudier medför detta att resultatet är uppbyggt endast baserat på min förståelse och tolkning av det lästa materialet. Detta i sin tur medför att arbetet står och faller mycket beroende på om jag har uppfattat materialet på ett riktigt sätt. Ett bra sätt att komma runt detta hade varit att komplettera litteraturstudierna med intervjuer för att dels få klartecken att man har

hängt med någorlunda samtidigt som man skulle kunnat få en annan (insatt) persons åsikter om vilka aspekter som är väsentliga att undersöka.

Referenser

Andersson, Jonas (nr.1 1997), PC World, I fokus

Bark, Mats (1997), Intranät i organisationens Kommunikation, Konsultförlaget, Uppsala

Bernstein, Beth Gold (februari 1998), Database Programming and Design,

Black, Brian (oktober 1996), Internet Systems (ett DBMS supplement), the Internet

Box, Don (maj 1996), Microsoft Systems Journal, Introducing Distributed COM and the New OLE Features in Windows NT 4.0

Brockschmidt, Kraig (maj 1996), Microsoft Systems Journal, How OLE and COM solve the problems of Component Software Design

Brockschmidt, Kraig (juni 1996), Microsoft Systems Journal, How COM Solves the Problems of Component Software Design, Part II

Chappell, David (1996), Understanding ActiveX and OLE, Microsoft Press, Redmond, Washington

Davis, Stephen R. (1994), C++ för dummies, IDG AB, Stockholm

Eddon, Guy och Eddon, Henry (mars 1998), Microsoft Systems Journal Homepage, Understanding the DCOM Wire Protocol by Analyzing Network Data Packets, <http://www.microsoft.COM/msj/>

Elbert, Bruce och Martyna, Bobby (1994), Client/Server Computing Architecture, Applications, and Distributed Systems Management, Artech House Inc., Norwood

Fenstermacher, D. Kurt (1997), ActiveX för dummies, IDG AB, Stockholm

Holme, Idar Magne och Solvang, Bernt Krohn (1991) Forskningsmetodik, studentlitteratur, Lund

Horstman, Markus och Kirtland, Mary (1998), Microsoft Systems Developers Network, DCOM Architecture

Karpinski, Richard (mars 01, 1997), CMPnet, For Web Interactivity, ActiveX Marks the Spot—Love it or hate it, ActiveX is sure to have an impact on the future of distributed computing, <http://www.techweb.com/se/directlink.cgi?NTG19970301S0076>

Khanna Raman, Editor (1994), Distributed Computing, Prentice Hall Inc.

Linthicum, David S. (maj 1996), Database Management System, Rise of the Intranent

Linthicum, David S. (december 1996), DataBase Management System, Client/Server Collapse

Linthicum, David S. (september 1997), Byte, Active Platform

Linthicum, David S. (september 1997), Byte, ActiveX Security

Microsoft Corporation, (1997), DCOM: The Distributed Component Object Model, A Business Overview, <http://www.williams.cs.ncat.edu/Mswhite/DCOM.htm>

MSDN (1998), Microsoft System Developer Network, DCOM Technical overview whitepaper

Referenser

Nordling, Erling (nr.5 1997), Datateknik, Intranet

Patrizio, Andy (september 15, 1997), CMPnet, proprietary Nature Slows Adoption Of ActiveX—Acceptance of the component technology is largely limited to intranet projects, <http://www.techweb.com/se/directlink.cgi?IWK19970915S0002>

Petroutsos, Evangelos (1997), Interactive web publishing, Research Triangle Park, NC: Ventana

Pressman, Roger S. (1997), Software Engineering, fourt edition, The McGraw-Hill Companies Inc.

Renaud, Paul E. (1996), Introduction to Client/Server Systems, second edition, John Wiley & Sons Inc.

Rogerson, Dale (1997), Inside COM, Redmond, Washington

Sessions, Roger (1998), COM and DCOM, Chickester:Wiley, New York

Tittel, Ed och Stewart, James Michael (1997), Intranät Bibeln, IDG Sweden Books, Stockholm

Förkortningar

ARPANET: Advanced Research Projects Agency Networks

CGI: Common Gatway Interface

COM: Component Object Model

DCOM: Distributed Component Object Model

DLL: Dynamic Linked Library

GUI: Graphical User Interface

HTML: Hyper Text Markup Language

HTTP: HyperText Transfer Protocol

IP: Internet Protocol

IRC: Internet Relay Chat

LAN: Local Area Network

OLE: Object Linking and Embedding

OMG: ObjectManagementGroup

ORB: Object Request Broker

RPC: Remote Process Call

SQL: Structured Query Language

URL: Uniform Resource Locator

WAN: Wide Area Network

WWW: World Wide Web