Göteborg 23 maj 1998

INSTITUTIONEN FÖR INFORMATIK

MAGISTERUPPSATS IA7400 VT98

VIKTIGA EGENSKAPER HOS EN APPLIKATION VID MIGRERING TILL INTRANÄT

Abstrakt:

Vi står idag inför ett paradigmskifte där företag går från att använda sina intranät som statiska anslagstavlor till att lägga upp kompletta applikationer direkt på nätet. Alla applikationer är inte lämpade för detta, och problemet för företagen är att avgöra vilka applikationer som är det. Vi genomförde en empirisk studie på ett företag, kombinerat med litteraturstudier, och kom fram till att följande egenskaper är direkt avgörande för en applikations lämplighet för en migrering till ett intranät: Om applikationen har en client/server-arkitektur, om data som används av flera användare finns i flera fristående kopior, eller att de användare som är beroende av applikationen inte kommer åt den. För att migreringen skall kunna anses som lyckad hävdar vi att man efter flytten ska ha åtgärdat eller uppnått minst en av dessa egenskaper.

AV

RALF KRAKOWSKI

&

JOHAN DAHLGREN

HANDLEDARE: LENNART PETERSSON

INNEHÅLLSFÖRTECKNING:

1 INLEDNING

Intranät är på frammarsch. Enligt rapporten *Intranets: Internet Technologies Deployed Behind the Firewall for Corporate Productivity* [1996] har idag många större företag egna intranät. På dessa nät finns allt mellan himmel och jord. Dagligen läser vi artiklar om hur mycket pengar storföretagen sparat på att utveckla intranätlösningar [Se bilaga 1]. Många sparar pengar på att använda sig av sina intranät istället för att trycka på papper och distribuera medan andra inför nya system för att förenkla kommunikationen mellan användarna [Se bilaga 1]. Men varför går ingen längre än så? Vi anser att vi idag kan se ett paradigmskifte i intranätutvecklingen. Statiska sidor med information och enkla kommunikationslösningar hör till gårdagen. Vi anser att nästa steg är att migrera redan befintliga applikationer till intranäten. På detta sätt uppkommer ett helt nytt sätt att använda sitt intranät på [Gaines, 1996]. Många företag verkar inte ha förstått att det de håller på med på sina intranät är ungefär vad rymdhunden Laika gjorde i sin sputnik redan 1957. Hon åker med utan att riktigt förstå vilka möjligheter hon har. Precis som Laika fick sin mat i en slang har företagen ungefär vad de tror att de vill ha och de håller sig därför, precis som Laika, ganska nöjda. Men varför ska man nöja sig med det man har? Idag vet vi att tekniken erbjuder oss så ofantligt mycket mer än telefonlistor och bilder på fru och barn. Vi står, som vi redan konstaterat, idag inför ett paradigmskifte på intranät-fronten. Företagen har äntligen börjat ta kontroll över rymdraketen de färdas i och de har så smått börjat pilla på de olika apparaterna de har omkring sig för att försöka komma underfund med hur de kan användas för att göra livet mera behagligt ombord.

Vi har i denna uppsats studerat hur en applikation som idag redan används inom en organisation bör se ut för att kunna migreras till ett intranät. Dessutom har vi tittat på hur resultatet av denna migrering bör se ut för att den, enligt oss, skall ses som lyckad. Vi har analyserat ett antal applikationer och funnit en lämplig kandidat. Där efter har vi migrerat denna till ett intranät och undersökt resultatet. Vi har därefter jämfört resultatet med vår hypotes och dragit ett antal slutsatser.

Eftersom det är ett såpass nytt ämne har det varit relativt svårt att finna bakgrundsmaterial till arbetet [Phanouriou & Abrams, 1997]. Detta är naturligtvis både spännande och jobbigt. Spännande därför att det innebär att vi rör oss i dåligt utforskade områden, och jobbigt därför att det alltid är lättare att utforska något som man har en karta över. Vi har under arbetets gång upptäckt att kartritare och upptäcktsresande är mycket underskattade yrkesgrupper.

2 KAPITELSTRUKTUR

Vårt arbete är indelat på följande sätt:

Vi börjar med att ställa upp vår problemdefinition och presentera vår hypotes i kapitel 3.
För att förstå de möjligheter som intranät erbjuder måste man känna till en del grundläggande detaljer bl.a. vad ett nätverk innebär. Eftersom intranäten använder sig av precis samma teknik som Internet måste man även ha en viss insikt i hur Internet fungerar. Därför ger vi i kapitel 4 en kort beskrivning av bakgrunden till intranät.

Därefter i kapitel 5, följer vår metodbeskrivning där vi förklarar hur vi gått tillväga och varför vi valt just den metod vi valt. Här finns också vår avgränsning där vi begränsar vårt problemområde.
I kapitel 6 kommer vi att redovisa de resultat vi kom fram till under arbetets gång, varför vi valde att bygga just det system vi byggde och varför vi löste de tekniska problemen som vi gjorde.
I slutsatsen, kapitel 7, kommer vi att diskutera dessa resultat i förhållande till vår hypotes och de andra teorier som vi har tagit upp under arbetets gång.

Därefter följer i kapitel 8 en kritisk granskning av vårt arbete där vi också försöker bemöta denna kritik.

Efter att genomfört vår undersökning fann vi att vi hittat fler nya frågor än vi hade besvarat. För den som är intresserad av att fortsätta undersöka detta ämne presenterar vi dessa nya frågeställningar i kapitel 9.

Kapitel 10 innehåller vår referenslista, och arbetet avslutas med en bilaga med ett par exempel på hur företag har kunnat utnyttja sina intranät för att göra stora besparingar.

3 PROBLEMDEFINITION

De flesta större företag har någon form av applikationer för att hantera interna transaktioner som t.ex. beställning av teknikertjänster och liknande. Enligt många av dem som vi kommit i kontakt med under vårt arbetes gång är dessa system ofta lågprioriterade eftersom de inte medför någon direkt inkomst för företaget.
När man väl tar sig tid att utveckla dessa applikationer blir projekten ofta kostsamma beroende på att man kanske inte har en enhetlig datormiljö på sin avdelning vilket innebär att man måste utveckla flera olika versioner av samma system för olika plattformar[Wallström , 1998], eller att man måste distribuera applikationerna till alla underkontor**.** Därför blir det ofta svårt att sälja in dessa system hos företagsledningen, som ofta ser kostnaderna för systemen som allt för stora i proportion till de intäkter de genererar. Företagsledningen anser ofta att det är på kunderna man tjänar pengar, inte på att bygga system för de egna anställda.

Samtidigt har allt fler företag börjat satsa pengar på att utveckla intranät. Enligt en undersökning av Forrester Research i USA presenterad i rapporten *Intranets: Internet Technologies Deployed*

Behind the Firewall for Corporate Productivity [1996] har idag två av tre av företagen på Fortune 500 listan redan intranät, eller planerar att införa dem. På dessa nät har man lagt ut allt möjligt, från personliga hemsidor till enkla databaskopplingar, men det är först nu på senare tid det börjat hända saker på intranät-fronten. Fler och fler företag har börjat intressera sig för de fördelar man kan vinna på att kombinera systemen för interna transaktioner med sina intranät genom att flytta de redan befintliga applikationerna till intranätet. När nu web-tekniken har mognat såpass att det går att bygga avancerade applikationer till en relativt låg kostnad finns det också en bra verktygslåda. Det är detta vi menar när vi talar om ett paradigmskifte; Företagen går från ett intranät med statiska HTML-sidor till ett intranät med interaktiva applikationer som kan fungera som en hjälp i utförandet av det dagliga arbetet [Gaines, 1996].
Om ett företag bara börjar använda denna nya resurs kan de spara stora pengar. Detta i form av t.ex. sänkta distributionskostnader och sänkt underhållskostnad för systemen.[Se bilaga 1] Andra fördelar man kan uppnå är att applikationerna kan bli plattformsoberoende och nå ett större antal användare än tidigare.
Problemet för företagen är nu bara att avgöra vilka applikationer som lämpar sig för denna flytt. Vi har utarbetat en hypotes där vi ställer upp ett par egenskaper som vi anser vara viktiga för att en applikation ska vara lämplig för en migrering till ett intranät.

3.1 HYPOTES

Genom studier av artiklar på området intranät kombinerat med våra egna tidigare erfarenheter av denna typ av utvecklingsarbete har vi utarbetat en teori för att underlätta migrering av applikationer till ett intranät:

För att en applikation ska vinna något på att flyttas till ett intranät skall den uppfylla minst ett av följande krav:

- Applikationen skall ha en client/server-arkitektur [se 4.5.2]
- Data som används av flera användare finns i flera fristående kopior [se 4.5.5]
- De användare som är beroende av applikationen kommer inte åt den [se 4.5.5]

För att migreringen skall kunna anses som lyckad skall man efter flytten ha uppnått eller åtgärdat minst en av dessa egenskaper.

4 BAKGRUND

4.1 Vad är ett intranät?

4.1.1 Definition av Intranät

De finns flera olika definitioner av Intranät. Här följer ett par exempel.

Ett intranät kan definieras som "a secure corporate network with rich functional features of Local Area Netorks (LAN) interconnected by the Internet and/or its technologies and applications" [Winston, Chellapa and Baura, 1998].

Ett intranät är egentligen bara ett Internet i miniatyr. Tekniken är precis den samma, dvs. ett TCP/IP baserat nätverk som länkar samman organisationens människor och information på ett sätt så att människorna blir mer produktiva, informationen mer tillgänglig och navigering genom alla resurser och applikationer enklare än förut [Kyle, Steward & Martines, 1997].

Eftersom Intranät baseras på de öppna standards som finns för World Wide Web (WWW) [se nedan] är de verktyg som behövs för att konstruera intranät- och internet-applikationer de samma. Skillnaden ligger i informationens tillgänglighet. Informationen på ett Intranät är endast tillgänglig för klienter inom organisationen eller intranät-gruppen.
Tidigare har detta skötts via LANs som varit skyddade av en brandvägg [Telleen, 1996] [se 4.1.2].

Telleen menar vidare att det som idag kallas Intranät är till stor del interna Web servrar. Medan LANs berikar "ensamma" datorer genom att koppla dem samman har Wide Area Networks (WANs) expanderat det potentiella geografiska omfånget av LANs. Internet och WWW har sedan tagit konceptet av WANs till nya höjder. Intranät har slutit cirkeln då detta också kallas virtuella LANs på Internet och försöker att slå samman det bästa av två världar. Vad är då grunden för denna typ av nätverk som Intranät baseras på?

4.1.2 Local Area Network

Då man kopplar samman ett antal datorer med hjälp av kablar vilket gör det möjligt för datorerna att kommunicera skapar man ett Local Area Network (LAN). Ett LAN länkar samman datorer inom ett begränsat område, ofta inom en avdelning eller byggnad så att de kan dela resurser. LANs kan konfigureras som "peer-to-peer" eller "client-server". Inom "peer-to-peer" nätverk kan alla datorer komma åt alla resurser på alla datorer som är uppkopplade på nätverket. Varje dator

kan användas som arbetsstation. Denna typ av nätverk brukar vara små, 5 till 6 datorer, för att fungera effektivt.

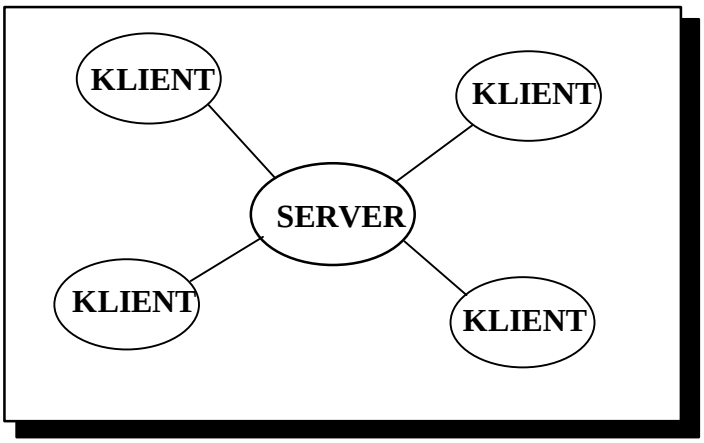

Fig 1. LAN (Local Area Network)

I ett client-server nätverk används en dator, oftast en maskin med hög kapacitet, för att styra flödet av information genom nätverket. Denna maskin kallas server. Servern är ansvarig för applikationer och delad data som alla klienter kan ladda ner och ha tillgång till samtidigt. På detta sätt kan man ha en central databas på en kraftfull server som behandlar all data och klienterna kan använda sina resurser till att visa informationen med ett användarvänligt grafiskt gränssnitt.

Winston et.al. menar att populariteten hos LAN har varit mycket stor i det flesta företag och har gett upphov till flera framsteg inom nätverksteknik och protokollutveckling. Eftersom ett LAN endast sträcker sig över ett mycket begränsat område, som t.ex. en avdelning på ett företag, är de också mycket säkra från intrång.

Men allt eftersom företagen växer och nya kontor startas upp både inrikes och utrikes uppkommer behovet av att bryta ner de geografiska barriärerna hos LANs, vilket leder till att man kopplar ihop sina lokala nät i ett enda stort nätverk som sträcker sig över stora ytor. Man skapar vad som kallas ett Wide Area Network (WAN).

4.1.3 Wide Area Network

Ett WAN kan beskrivas som ett nätverk som sträcker sig över en stor geografisk yta och där en del av nätet ofta utgörs av telenät, vanligt eller ISDN. Funktionaliteten är densamma som för ett lokalt nät (LAN), men kostnaden är högre och kapaciteten lägre [Ewert, 1997]. Vad händer då man vill öppna ett kontor i Australien? Att sätt upp ett WAN som skulle kommunicera via satellit är en stor kostnad även för stora multinationella organisationer. Här finner enligt Winston et.al. Internet sin plats, det ultimata WANet.

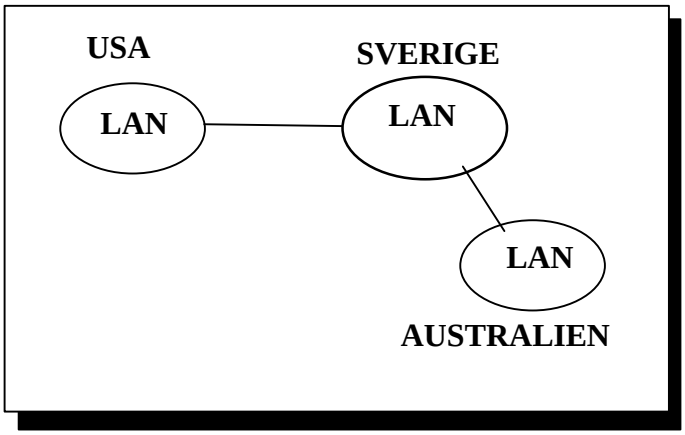

Fig. 2. WAN (Wide Area Network)

Många av de applikationer och tillämpningar som tillkommit sedan WWW exploderade är inriktade mot elektronisk kommers men det visar bara en liten del av de möjligheter som finns med denna teknik menar Winston et.al. De menar vidare att det idag finns en stor “tyst majoritet“, som författarna utrycker det, som redan använder elektronisk media för kommers inom organisationen - det vi idag kallar intranät eller “Enterprise Wide Networks“.

Eftersom vi tidigare konstaterat att ett intranät inte är mer än ett nätverk kopplat och modellerat efter Internet och WWW som bara nås av människor inom organisationen eller gruppen skall vi ge en mycket kort bakgrund till Internet och WWW. Det exakta förfarandet för transport över Internet och dess protokoll beskrivs inte i denna uppsats utan den intresserade hänvisas till David Mayrs “History of the Internet and WWW“ [Mayrs, 1997].

4.1.4 Internet

På 1960 talet utvecklade The U.S. Department of Defense's Advanced Research Projects Agency (ARPA) ett nätverk för att förbättra militär kommunikation. Nätverket som fick namnet Advanced Research Projects Agency Network (ARPANET), användes för att skicka information mellan militären och dess underleverantörer för olika statliga projekt. Under 1970 talet utvidgades nätverket till andra länder, och under1980 talet expanderade nätverket mycket snabbt. Många universitet, forsknings institut och statliga institutioner började koppla upp sig på vad vi idag kallar Internet. De tjänster som användes var bl.a. email, news och gopher.

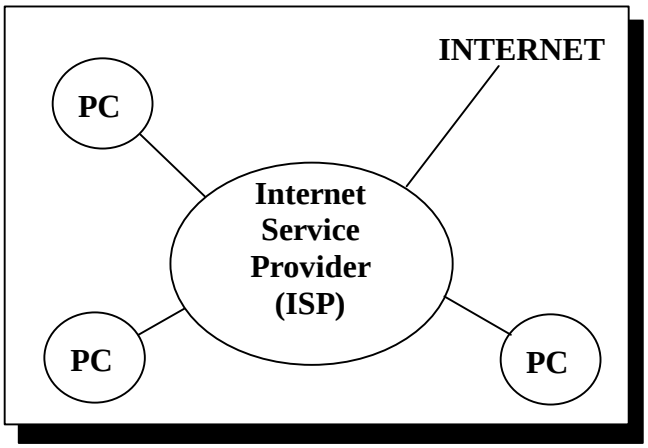

Fig. 3. Internet

4.1.5 World Wide Web

Under 1989 började Tim Berners-Lee, anställd hos the European Laboratory for Particle Physics (CERN) utveckla ett antal protokoll för informationsdistribution på Internet som kallas World Wide Web även kallat WWW.
WWW-protokollen var utvecklade för att underlätta informationsdistribution mellan olika organisationer. Dessa protokoll är plattformsoberoende. Med detta menas att information som skapas på en plattform t.ex. Unix kan presenteras på en annan plattform och den kommer då att se exakt likadan ut.

Målet med WWW var att skapa ett online system som kunde låta icke tekniska användare att dela data utan att konstiga kommandon eller kryptiska gränssnitt. Inom ett par år hade användare utanför CERN börjar skapa sidor för Weben, som WWW också kallas, och mjukvaruföretagen hade börjat bygga läsare. Läsare finns idag från flera företag så som Netscape, Microsoft och Sun. Idag finns det en grupp som kallas The World Wide Web Consortium (W3C) som utvecklar standarder för WWW [Wesley & Wesley, 1996].

4.2 Interna webservrar

Det första steget mot ett Intranät är att sätta upp en intern webserver. Man använder då samma teknik som för att sätta upp en extern webserver mot Internet. Det egna nätverket skyddas mot utomstående med hjälp av brandväggar och andra säkerhetsmekanismer.

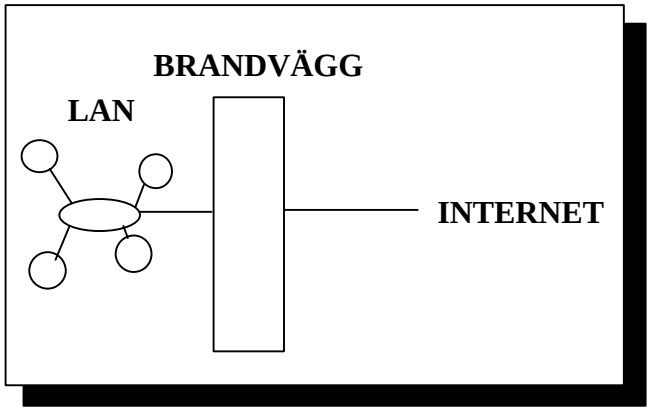

Fig. 4. Nätverk kopplat mot Internet

Data som transporteras över publika nät kan skyddas genom kryptering. På detta sätt kan applikationer på den interna webservern skyddas och bara nås av människor med tillgång till det egna nätverket. Man kan koppla ihop flera LAN inom företaget och använda sig av Internet som kommunikationsmedel. *Det är detta vi menar med ett intranät* [se figur 5]:
Genom ett första steg har de interna webservrarna gått från att bara presentera statisk information till att också bli dynamiska genom sökning i databaser. Det paradigmskifte vi vill påvisa i vår uppsats är nästa steg - att migrera redan befintliga applikationer till de interna webservrarna och därmed införa ett riktigt Intranät. Den ökade spridningen och utvecklingen av dynamiska programmeringsspråk som t.ex. Java, ger ett brett utbud av möjligheter för dagens Intranät. I en heterogen operativsystems miljö, kan Java till stor del eliminera plattformsberoendet. Ett enkelt system skrivet i ett plattformsoberoende språk kan distribueras genom hela organisationen vare sig man använder Unix, Windows 95 eller något annat operativsystem.

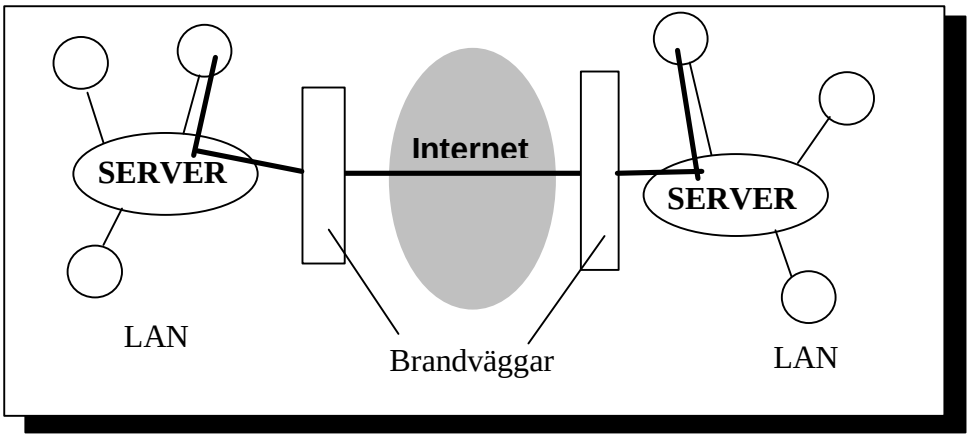

Fig. 5. Intranät

Intranät har precis som andra tekniker som utvecklas snabbt, utvecklats “quick and dirty“ istället för genom väl genomtänkta och fullständiga lösningar. De flesta av dagens intranät eller interna webservrar befinner sig i dagsläget i en situation där de bara har upptagit Internet tekniken istället för att utveckla den menar Winston et.al..

Dagens interna webservrar innehåller till stor del följande tjänster enligt Winston et.al:

E-mail: Intern e-mail till alla anställde inom företaget är ett vanlig tjänst på Intranät. Olika organisationer inför standard SMTP-baserade e-mail system. Dess system fungerar både för internt och externt bruk.

Online publicering: Publicering av företagets dokumentation är den vanligaste företeelsen i användningen interna web tekniker. Nyhetsbrev, lunchlistor och månadsrapporter kan presenteras på nätet istället för att distribueras på det traditionella sättet.

Online sökning: Sökning i företags dokumentation, kataloger, prisuppgifter eller externa länkar finns inom denna kategori.

Applikations distribution: Vanliga applikationer så som Excel makron och mallar kan laddas ner från den interna webserver. Uppdateringar och versionskontroller kan enkelt hanteras p.g.a. central underhållning.

Som vi tidigare nämnt skall ett riktigt Intranät slå samman det bästa från Internet och LAN. Det finns flera sätta att införliva funktionaliteten hos WWW med det interna företagsnätverket. Det fullständiga Intranätet skall ha alla de funktioner som ett LAN har. Detta inkluderar bl.a. fildelning, printerdelning och applikationsdelning, men det finns idag inget säkert sätt att dela dessa funktioner på Internet enligt Winston et.al.

Det första steget är dock att se på möjligheten till att migrera redan befintliga applikationer till det existerande intranätet. För att kunna ställa upp någon form av kriterier för hur väl en applikation lämpar sig för en flytt måste vi först ge en kort beskrivning av de olika typer av applikationer som finns.

4.3 Vilka typer av applikationer finns det?

Vi har identifierat två huvudtyper av online-applikationer som används i dag. Med online-applikationer menar vi applikationer där människor sitter och arbetar mot någon form av data som uppdateras samtidigt som personen skriver in ändringarna, till skillnad från batch-hantering där alla ändringar som matats in under dagen genomförs vid en bestämd tidpunkt. Dessa två olika typer har vi valt att beteckna med de engelska orden Standalone och Client/Server. Client/Server

applikation skall i detta fall ej förväxlas med client/server nätverk. Här följer en beskrivning av dessa applikationstyper.

4.3.1 Standalone-applikationer

Med standalone applikationer menar vi de applikationer där all behandling sker i samma process på en lokala maskin. Exempel på standalone-applikationer är tex Photoshop och andra beräkningstunga program där stora mängder data måste kunna behandlas i realtid.
Den stora fördelen med standalone-applikationer är att man inte behöver kunna kommunicera med andra användare eller datorer för att genomföra sitt arbete. På detta sätt slipper man de problem med långsamma kommunikationer och avbrutna förbindelser som ibland kan uppstå när man arbetar över ett nätverk.
En annan stor fördel med standalone-applikationer är att användaren har en egen version av programmet på sin dator vilket innebär att han kan spara ner sina egna personliga inställningar för programmet så att det är anpassat efter hans eller hennes behov.
Nackdelen är att det blir dyrt att uppgradera eller förändra applikationen. Eftersom alla användare har sin egen kopia av programmet så måste man vid en uppdatering gå ut till varje enskild maskin och manuellt installera den nya programvaran, något som kostar både pengar och tid för företaget.
För att komma ifrån detta problem kan man istället använda sig av en client/server-arkitektur:

4.3.2 Client/Server-applikationer

Client/Server refererar till en typ av databearbetning baserad på ett “begäran-tilldelnings“ tänkande där kommunikationen mellan datorerna baseras på skickandet av meddelanden [Kalakota, 1997]. Skickandet av dessa meddelanden sker mellan kommunikationsprocesser.

Klienten är en process (ett program) som sänder ett meddelande till en serverprocess (program), där klienten begär att servern skall utför en uppgift. Klienten består vanligtvis av gräsnssnittsdelen av applikationen där bl.a. validering av data som användaren matat in i applikationen sker. Den klientbaserade processen är den del av applikationen som användaren ser och interagerar med. En av klientens nyckelegenskaper är det grafiska gränssnittet mot användaren, dvs det som användaren ser av programmet, och därmed påverkar hans uppfattning av hela systemet. En serverprocess (ett program) fullföljer klientens begäran genom att utföra den begärda uppgiften. Serverprogram brukar få begäran från klienter och kan då utföra databasförfrågningar och uppdateringar, sammanställa data och leverera ett svar till klienten. Serverprocessen brukar finnas på en annan maskin på nätverket. Hårdvaruplattformen och operativsystemet i ett client/server system är oftast inte det samma hos klienten och servern. Klienten och servern har fundamentalt olika krav på maskinresurser i form av processor hastighet, minne, disk utrymme och kapacitet och in och ut enheter.
Det finns flera stora fördelar med client/server-tekniken.

För det första är den öppen. Den gör det möjligt för organisationer att distribuera data över nätverk med hjälp av grafiska arbetsstationer, servrar och stordatorer. Client/server modellen ger möjligheter till att anpassa behoven för användarna med hänsyn till var man befinner sig och vilka resurser man har. Joseph O. Nattey [1996] pekar bl.a. på följande fördelar med client/server modellen:

- Samarbete – nyckelkomponenter i systemet arbetar tillsammans för att uppnå ett gemensamt mål.
- Utbyggbarhet – alla nyckelelementen kan bytas ut när så erfordras för att bygga ut eller förändra systemet utan att andra delar av systemet påverkas.
- Anpassningsbarhet – Ny teknik kan enkelt inkorporeras i systemet.
- Tillgänglighet – Datan kan nu kommas åt via företagets nätverk vilket kan ge fler användare tillgång till informationen.
- Säkerhet – All verifiering av användaridentiteter och andra säkerhetsspärrar kan läggas på servern.

Han menar vidare att i teorin ser client/server ut som en ypperlig arkitektur. Det gör det möjligt att snabbt skapa grafiska applikationer. Under ytan finns dock oväntade kostnader menar Nattey. Han pekar på följande nackdelar:

- Tekniken är i dagsläget varken mogen eller helt stabil. Den är heller inte helt enkel att sätta ihop.
- Eftersom många idag fortfarande inte är insatta i hur client/server-arkitekturen fungerar investerar många företag i client/server-system i onödan eller gör felsatsningar.
- Ju mer distribuerat nätverket blir desto större blir säkerhetsriskerna.
- Client/server är en teknik under utveckling och därför finns det i dagsläget ingen standard för hur de ska se ut. Detta innebär att det på marknaden finns en uppsjö av produkter från olika tillverkare som har löst de olika problemen på sina egna sätt vilket får till följd att det kan bli svårt att koppla ihop ett nyinköpt system med företagets redan befintliga system.

Client/server applikationer kan i sin tur delas in i två- och treskikts lösningar.

4.3.3 Skillnaden mellan 2- och 3-skikts client/server-applikationer

En tvåskikts client/server lösning innebär att man har en klient som går direkt mot någon central resurs, som t.ex. en databas eller liknande. På detta sätt får man en enkel struktur som är lätt att hantera. När man bygger en tvåskiktslösning så måste all logik ligga i klienten vilket innebär att den blir relativt stor, vi får en *fet klient* [se nedan]

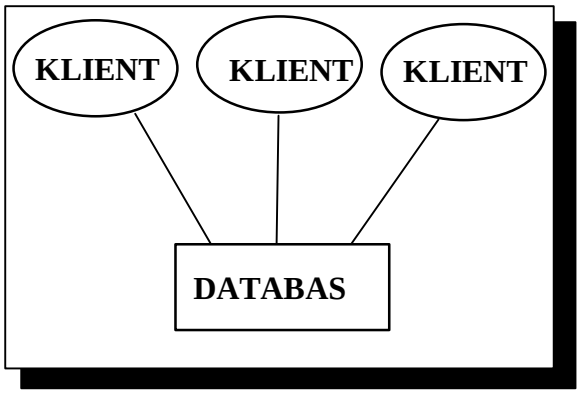

Fig. 6. Tvåskikts client/server

En treskiktslösning innebär att man har ett mellanled mellan klienten och databasen, som tar hand om inkommande förfrågningar, skickar dem vidare till databasen och vidarebefordrar resultatet till klienten. Här kan man lägga större delen av programlogiken i servern, vilket innebär att vi får en *tunn klient* [se nedan].

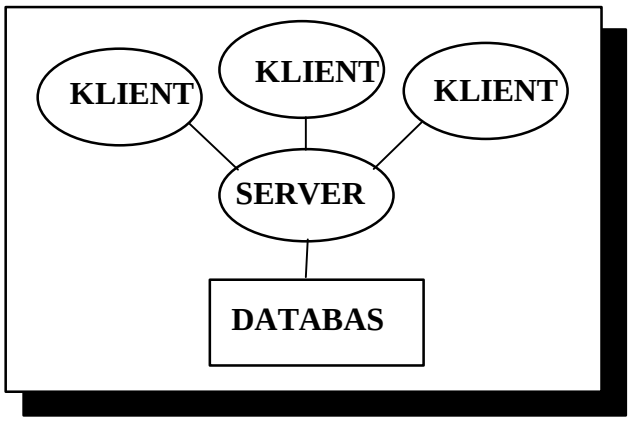

Fig. 7. Treskikts Client Server

4.3.4 Skillnaden mellan tunna och feta klienter

Idén med tunna klienter är en av hörnstenarna i client / server tänkandet. Med en tunn klient menas att det ligger lite eller ingen logik i användarens del av systemet, utan all databehandling sker på servern. Med en tunn klient uppnår man flera fördelar:
Eftersom största delen av allt arbete utförs av servern så kan man dra nytta av dess kapacitet. De maskiner som används som servrar har oftast prestanda som vida överstiger de PCs som står på folks skrivbord och det torde vara logiskt att utnyttja denna prestanda.

4.3.5 Client/server och intranät - knyta ihop säcken

4.5.5.1 Fördelar med Client/server

Hos client/server modellen kan man finna flera fördelar. Patrick Smith [1994] belyser några av de punkter som vi anser är viktiga hos denna modell:

1. Förbättrad data delning

Data som är samlad som en del av den normala arbetsprocessen och underhållen på en server är omedelbart tillgänglig för alla användare som har rättigheter till denna data. Transparenta nätverk, dvs. nätverk där användarna inte behöver bry sig om den tekniska funktionaliteten hos nätverket, försäkrar att data är tillgänglig samtidigt för alla behöriga användare.

2. Integrerade tjänster

I client/server-modellen har man full tillgång till all information man är behörig att använda. Man behöver inte byta till terminalläge eller logga på en annan process för att komma åt information.

3. Delande av resurser över olika plattformar

Client/server-modellen ger möjligheten att implementera öppna system. Användaren kan utnyttja resurser över plattformsgränserna. Sålunda kan användare erhålla service och transparent tillgång till tjänster som tillhandahålls av databas-, kommunikations- och applikationsservrar. Applikationen är helt skild från operativsystemet och använder sig bara av dess resurser.

4. Data och bearbetning kan ske oberoende av plats

Vi har rört oss ifrån den maskin-centrerade databearbetningens era under 1970 och 1980 talet till en ny era där PC användare har krävt system som är användar-centrerade [Gaines, 1996]. Förut, så loggade användaren på stordatorn eller minidator. Syntaxen för varje plattform var unik. Idag kräver användarna ett standardiserad utseende. Användaren vill kunna logga på applikationen från sin dator utan att bry sig om var eller hur datan bearbetas.

5. Centralisering

Genom att centralisera databaser och andra resurser kan man öka deras tillgänglighet för användarna. Med tunna klienter går man tillbaka till 70- och 80-talens terminaltänkande. Man centraliserar databehandlingen och förser användarna med tunna klienter som inte har någon egen logik. Samtidigt behåller användarna sina PCs och på så sätt kan man få det bästa både vad gäller distribuerat och centraliserat tänkande.

4.5.5.2 Client/server i form av webb-teknik

Genom att använda sig av webb-teknik uppnår man en mycket lägre underhållskostnad. Eftersom användarna inte har några egna klienter i egentlig bemärkelse utan bara laddar ner dem från en central server när de behöver användas finns det egentligen bara *en* fysisk existens av programkoden. Vid en eventuell uppdatering behöver man bara ändra på ett enda ställe oavsett hur många användare som sitter i den andra änden.
Samtidigt skiftar man ansvaret för informationen till användarna. Tidigare har all data skickats ut till alla användare oavsett om de var intresserade eller inte. Detta sätt att agera kallar Steven L. Telleen, Ph.D en push-mentalitet (Push mentality)[Telleen, 1996], dvs ledningen anser sig veta vad användarna vill ha och skickar ut den informationen till dem. Om man inte vet vad användaren vill ha så skickar man istället allt som man *tror* att han vill ha. I företagsekonomiska termer så skapar man först en tillgång och förväntar sig sedan en efterfrågan.
Om man däremot har anammat det nya sättet att tänka har man vad Telleen kallar en pull-mentalitet (pull-mentality), dvs att användarna själva måste söka upp den information de är intresserade av. När informationen efterfrågas så görs den tillgänglig. Man märker att det finns en efterfrågan och först då skapar man tillgångarna.
Enligt Telleen arbetar de flesta företag och privatpersoner idag enligt push-tänkandet. Vi förutsätter att andra ska förse oss med information och vi får den också. Problemet är bara att vi som människor inte klarar av all denna information utan drabbas av information-overload och blir stressade av att inte veta om vi råkat missa någon viktig nyhet.
Genom att samla all data till en central plats kommer man ifrån de problem som redundans i datan medför [Date, 1994]. Om man bygger sitt system i enlighet med tankarna om centralisering av de resurser som lämpar sig för det så vinner man också den stora fördelen att man lätt kan ansluta nya användare till systemet.

Man borde ta chansen att komplettera det nuvarande intranätets telefonlistor, nyhetsbrev och disskusionsgrupper med ett gränssnitt till alla lämpliga system. Man kan integrera client/serversystem genom att ersätta klienten med en webbläsare och på så sätt få ett enda gränssnitt mot sina system. En stor fördel med intranät är just det gemensamma användargränssnittet vilket reducerar efterfrågan av upplärning [Holtz, 1996] .
Om man så önskar, kan man istället för att göra om servern, lägga till en IP-baserad mellanvara mellan webbläsaren och server menar Kent Edquist [Edquist, 1998]. Vidare kan man fånga upp information avsedd för dumma terminaler och mata informationen till användarnas webbläsare istället. Genom att bygga broar mellan databaser och webbservrar så att användarna på ett intuitivt sätt kan komma åt informationen genom intranätet, gör att chansen är stor att företagets standardsystem blir populärare genom att ge användarna ett enklare och framför allt gemensamt gränssnitt. Edquist pekar på ett antal fördelar man får om man vill gå över till lösningar byggda på webbteknik:

- Man får en miljö som är mer homogen och väsentligt enklare att underhålla och modernisera.

- Användarna får ett gränssnitt till all företagsintern information oavsett om det är enkla telefonlistor eller komplex information som uppdateras i realtid från företagets administrativa system.

- Man hjälper företaget att förbereda sig för att offensivt använda dessa nya teknologier inom nya områden som elektronisk handel, dator- och telefonintegrerade säljstödssystem eller extranätbaserade system som hårdare integrerar företagets partners till företaget.

Införandet av en client/server arkitektur, som ett intranät innebär, ger inte bara *tekniska* förändringar, utan även organisatoriskt uppkommer förändringar. Shel Holtz [1996] menar att organisationen vinner styrka på effektiv delning av information, istället för att undanhålla den. Idag ser vi på organisationen genom ett filter av traditionella affärsstrukturer och processer baserade på en modell av affärer som fanns under den industriella ekonomin. Hjärtat i denna ansats är den traditionella top-down organisationen. Varje person på varje nivå skulle bara ha så mycket information så att han/hon kan utföra sitt arbete. I årtionden har, enligt Holtz, mellanchefer undanhållit information för att behålla styrka och makt. Denna modell har fungerat bra inom organisationer som producerat varor, och för vilka de överordnade faktorerna inom produktionen varit land, arbetskraft och kapital. I den nya informationsekonomin har land, arbetskraft och kapital ersatts som primära faktorer av produktionen av information. Även organisationer som producerar varor har anammat egenskaperna hos informationsekonomin. Detta betyder inte att de slutat producera varor, utan det innebär att information numera är den viktigaste byggstenen i tillverkningsprocessen enligt Shel Holtz. Genom ett intranät och dess varierande komponenter (disskusionsgrupper, email och så vidare) rör sig informationen fritt. Informationen kan inte undanhållas och ägas.

Den största fördelen med att använda en client/server-lösning är dock att den fungerar på samma sätt som Internet/intranät. När man via sin browser läser en sida på WWW arbetar man enligt en client/server-metod: Någonstans i världen finns en web-server som tillhandahåller tjänsten att visa en viss sida med information i form av t.ex. Java-applets eller ren text. Denna server anropas av browsern (klienten) med en begäran om att skicka över information (web-sidan). På precis samma sätt fungerar det på ett intranät. Eftersom WWW-tekniken redan är uppbyggd enligt denna modell är det mycket lämpligt att även bygga de applikationer som ska användas via någon form av WWW-baserat nät enligt samma princip.

5 METOD

5.1 Kvantitativa metoder

Att använda sig av kvantitativa metoder innebär som bekant att man samlar in stora mängder data och sedan sammanställer dem med någon form av statistiska metoder, för att på så sätt kunna dra slutsatser ur sitt material. Dessa metoder lämpar sig främst då man har att göra med t.ex. enkätundersökningar eller tekniska mätresultat, på vilka man vill utföra någon form av beräkningar för att få fram medelvärden och liknande. Eftersom vårt ämne inte är av den typen anser vi att det enbart skulle vara slöseri med tid att försöka genomföra en sådan undersökning.

Samtidigt skulle det kunna vara på sin plats med en enkätundersökning bland landets större IT-företag för att undersöka hur de har utnyttjat sina intranät. En enkät med frågor av typen "Vilken typ av applikationer anser Ni är lämpliga att lägga ut på Ert intranät?" skulle kunna ge oss en fingervisning om vad de flesta företag anser vara lämpliga applikationer för intranät. Efter att ha läst ett par gamla examensarbeten förstod vi att alla företag inte är intresserade av att tala om för sina konkurrenter vad de har för strategier för sina intranät, utan de betraktar detta som affärshemligheter som de helst håller för sig själva. Företaget vi skrev applikationen för är t.ex. väldigt måna om att ingen utomstående ska få tillgång till informationen som visas. Om företagen skyltar med sina flotta intranät och vilken information de har där så skulle detta kunna locka konkurrenter eller andra till att försöka komma åt informationen genom att t.ex. hacka sig in på nätet. Ett ännu värre hot är att konkurrenter kan bygga egna system baserade på det system man tagit fram vilket får till följd att det egna företaget tappar den konkurrensfördel som dess intranät kanske givit.
En annan anledning till att vi inte använt oss av kvantitativa metoder är tidsaspekten. För att en enkätundersökning ska vara statistiskt giltig måste den vara baserad på ett stort undersökningsunderlag och för att få ett sådant så skulle vi bli tvungna att skicka ut en stor mängd enkäter eller göra väldigt många intervjuer och det har vi inte tid med, eftersom vi även har ägnat en tid åt att skriva själva applikationen. Även om vi hade hunnit skicka ut alla enkäterna och fått tillbaka dem i tid så hade vi nog inte hunnit sammanställa materialet och dra några statistiskt giltiga slutsatser utifrån det.

5.2 Kvalitativa metoder

När man jobbar med människor på det sätt som vi i vår bransch måste göra så kan man inte spika en design från början och sedan slaviskt följa den hela vägen till färdig produkt. I stället så måste man låta designen växa fram under arbetets gång och man får inte vara rädd för att ändra eller kanske helt förkasta de planer man hade från början [McCullough, 1998]. När man använder sig av kvalitativa metoder så spelar alltid forskaren själv en stor roll i undersökningen. Alla data tolkas av forskaren och hur man än försöker låta bli så kommer man aldrig att helt kunna avstå från att lägga med egna värderingar i slutrapporten [McCullough], men detta är inte alltid av ondo. Om forskaren är en väl ansedd man inom sitt område så kan det istället vara en positiv sak

att han har lagt sina egna värderingar och åsikter i rapporten, eftersom han betraktas som expert på det han gör.
Nackdelen med en kvalitativ metod är att man inte rakt av kan överföra resultaten till andra företag. Inte ens om företagen är mycket lika till utseende och bransch kan man vara säker på att man kommer att lyckas med projektet bara för att man gör på samma sätt som sist när det gick bra. Om man baserat sina beslut på kvantitativa metoder kan man i alla fall med en viss grad av säkerhet säga att resultatet kommer att bli godtagbart [McCullough]. Men som vi tidigare nämnt så anser vi fortfarande att en kvalitativ studie ger mer detaljerade uppgifter och eftersom vi också måste ta hänsyn till människor så väljer vi denna metod.

För att kunna genomföra denna detaljerade kvalitativa studie och kunna avgöra om vår hypotes stämmer överens med verkligheten bestämde vi oss för att gå empiriskt till väga och bygga en egen applikation för ett företags intranät. Detta kombinerat med literaturstudier av tidigare arbeten på områdetborde kunna ge oss svar på huruvida vår hypotes stämmer överens med verkligheten. Vi gick ut till ett antal företag och hörde oss för var vi bäst skulle kunna utföra vår undersökning.

5.3 AVGRÄNSNING

I vårt arbetet har vi valt att endast använda oss av Sema Group för att testa våra teorier. Anledningarna till att vi valde just Sema är flera:

1. Stort, multinationellt företag
2. Resurser
3. Intresserade av ny teknik
4. Engagerade kontaktpersoner

1. Stort multinationellt företag

Anledningen till att vi valde ett stort företag som etablerat sig i hela Europa är att vi tror att det är den typen av företag som satsar mest på sina intranät (2 av 3 företag på Fortune 500 listan har som vi redan nämnt idag intranät eller planerar att införa dem). Små företag med bara ett par anställda har ingen större nytta av att satsa på intranät [Holz, 1996] och därför kommer de antagligen att vara mer tveksamma till att låta oss få fria händer vid utvecklandet av vår applikation. Om vi hade valt ett litet företag så hade vi antingen blivit tvungna att försöka hitta ett med ett redan befintligt intranät, eller också sätta upp ett själva. Inget av dessa alternativ var egentligen realistiskt, eftersom vi hade en begränsad tid att utföra arbetet på.
De flesta medelstora företag har antagligen redan intranät, men de har antagligen inte personal att avvara för att handleda två studenter som gör sitt examensarbete, så vi valde att försöka hitta ett stort företag.
Att Sema Group är multinationellt påverkade kanske inte hur vi utvecklade vår applikation, men det är ju alltid roligt som programmerare att veta att den applikation man byggt kan användas av folk över hela Europa.

2. Resurser

Eftersom Sema är ett såpass stort företag (16.000 anställda över hela världen och en omsättning på ca 15 miljarder kronor per år) så innebär det naturligtvis att det finns gott om resurser för att utföra försöksverksamhet. Eftersom web-tekniken är en såpass ny företeelse och därför också dras med en hel del barnsjukdomar så är det få företag som hittills vågat satsa på att utveckla applikationer i för den.
Tack vare att Sema är såpass stort som det är så finns det alltid gott om kunnigt folk i närheten när man får problem, och eftersom alla var väldigt positiva till vårt arbetet så kunde vi räkna med att få det stöd vi behövde.
Materiella resurser i form av datorer och telefoner är andra saker som man bara måste ha för att kunna utföra denna typ av arbete och sannolikheten att få tillgång till dem är avsevärt högre på ett företag med stora resurser än på ett med få tillgångar.

3. Intresserade av ny teknik

På företaget har man precis börjat experimentera med nya "heta" tekniker som t.ex. Java, Javascript och Perl för web-utveckling. Detta passade oss utmärkt, eftersom även vårt intresse ligger inom denna sektor. Eftersom man på företaget redan hade ett intresse för denna nya teknik så kände vi att vi skulle ha en god chans att få relativt fria händer vid utvecklingsarbetet. Hade vi valt ett annat företag med en mer konventionell syn på mjukvaruutveckling med allt vad det innebär av fördomar och farhågor gentemot att "koppla upp sig på nätet", så kanske de hade varit mer restriktiva med att ge oss tillgång till servrar och stordatorer.

4. Engagerade kontaktpersoner

.
De personer som vi möttes av på företaget visade stor entusiasm över våra idéer och vi kände att vi skulle kunna få det stöd vi behövde för att utföra vårt arbete. Under arbetets gång har det också visat sig att vi valde rätt, för folk har alltid tagit sig tid att hjälpa oss när vi fått problem.

6 RESULTAT

6.1 Val av applikation

6.1.1 Sökta kriterier hos applikationen

Då vi valt en partner för samarbete, undersökte vi deras utbud av applikationer. Vi sökte en applikation som i enlighet med vår hypotes skulle uppfylla åtminstone ett av dessa krav:

- Applikationen skall ha en client/server-arkitektur [se 4.5.2]
- Data som används av flera användare finns i flera fristående kopior [se 4.5.5]
- De användare som är beroende av applikationen kommer inte åt den [se 4.5.5]

Vi ansåg att LedningsInformationsSystemet LIS var ett bra val då det uppfyllde flera av de punkter vi satt upp. Nedan följer en beskrivning av LIS och motivering till val av denna applikation.

6.1.2 Introduktion till LIS

Ledningsinformationssystem (LIS) är ett system för en speditionsfirma som presenterar information om antal levererade enheter för olika distrikt i Sverige. Användaren kan välja att visa information om distrikt, orter och kontor. Efter att användaren valt vad han/hon vill titta på presenteras resultatet i ett diagram. Vidare kan användaren göra ytterligare val med hänsyn på olika distributionstyper. LIS är uppdelat i flera delar. De två huvuddelarna är vecko- och dag – statistik, där veckostatistiken visar antalet leveranser per vecka och dagstatistiken visar leveranser per dag över en period på sextio dagar räknat från dagens datum.

6.1.3 Teknisk beskrivning av LIS

LIS olika delar är i sin tur uppdelade i delar. Varje del består av två delar. En del som användaren har på sin dator och en del som finns på en lokal server i användarens nätverk. Användarens del består av ett program skrivet i Visual Basic. Där finns både ett grafiskt gränssnitt och själva programlogiken. Programmet kommunicerar med en databas på en server i det lokala nätverket. *Applikationen kan klassas som en tvåskikts client/server applikation enligt vår definition. Den uppfyller då det första villkoret i vår teori.*
Från databasen hämtar klienten (dvs. den del som användaren har på sitt skrivbord) all data som användaren söker efter. Från början distribuerades klienterna ut till användarna med hjälp av disketter, men nu kan de distribueras och uppgraderas automatiskt via företagets nätverk. Databasens uppdatering är lite krångligare. Varje natt körs det ett jobb på stordatorn (Se figur 8: 1) som använder sig av dess databas (2). Programmet genererar en databastabell. Denna tabell flyttas till en central server (3) och översätts till Microsoft Access-format. Microsoft Access är den typ av databashanterare som finns ute hos de olika användarna i landet. Databastabellen skickas sedan via ftp ut till alla kontor runt om i landet. Den placeras på deras server (4) i deras lokala nätverk och på morgonen kan användaren hämta data från kontorets lokala databas med hjälp av sin Visual Basic-klient. På detta sätt får alla kontor varsin kopia (5) av denna tabell.

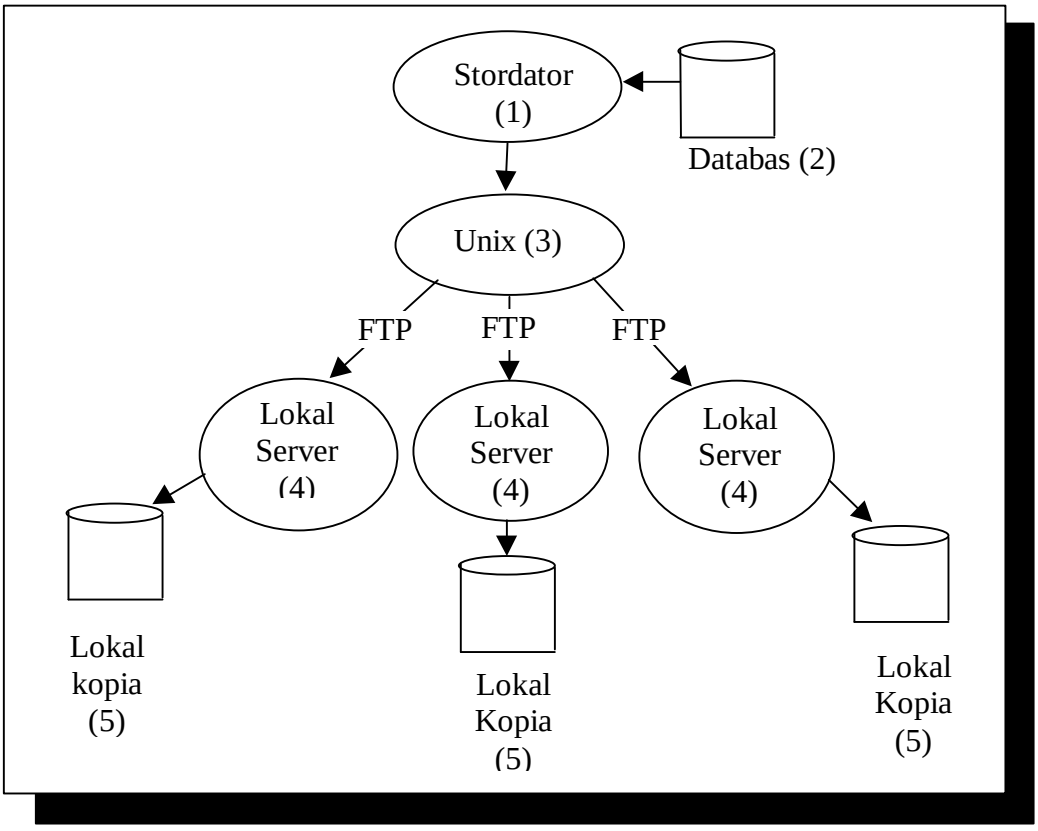

Fig. 8: Distribuering av databasen i det befintliga LIS

Här uppfylls även det andra villkoret i vår teori, nämligen att flera användare har var sin databas med samma data. Dessutom är tillvägagångssättet för att skapa databasen komplicerat och känsligt för driftsstörningar.

Man hade i dagsläget inte bestämt sig för vilka som skulle ha tillgång till applikationen. Därför kan vi inte uttala oss om huruvida den uppfyllde det tredje villkoret i vår hypotes.

Vi ansåg därför att applikationen hade större delen av de egenskaper som vi satt som viktiga för applikationens lämplighet för en migrering till ett intranät, och valde därför att försöka flytta densamma till företagets intranät.

6.1.4 Uppdelning av LIS

LIS är som vi tidigare påpekat indelat i ett antal delsystem. Vi valde att endast bygga den del som hanterade dagstatistiken. Att vi valde att försöka implementera just dagstatistiken berodde på flera saker:

- Den innehöll ofarliga data
- Det var ett litet, begränsat problem
- Intressanta programmeringstekniska problem

Ur företagets synvinkel var det den del av systemet som var mest “ofarlig“ eftersom det som visas inte uttrycks i kronor utan i antal leveranser per dag och dessa data är inte lika känsliga som de där faktiska förtjänster redovisas.
För vår del passade uppgiften utmärkt eftersom det var ett litet, begränsat problem som vi beräknade att vi skulle hinna genomföra på den utsatta tiden. Om vi skulle lyckas bygga denna del skulle vi även kunna bygga resten om vi fick mer tid på oss, och därför såg vi den som en lämplig startpunkt. Rent programmeringsmässigt var det också en intressant uppgift. Ur programmeringssynpunkt är det alltid roligt att få arbeta med den senaste tekniken inom ett område.

6.2 Beskrivning och val av verktyg

När vi funnit en applikation som vi ansåg lämplig att migrera till företagets intranät började vi undersöka vilket verktyg som bäst passade våra syften.
Idag finns det ett antal färdiga hjälpmedel för att bygga denna typ av applikationer. De är anpassade för ganska skilda användningsområden. Vi utgick i från fem olika typer av verktyg. Nedan presenterar vi de olika verktygen för att sedan motivera vårt val för att kunna åstadkomma en lyckad migrering:

6.2.1 HTML (HyperText Markup Language)

För att kunna publicera information för global distribution behöver man ett universellt språk, ett slags modersmål som alla datorer kan förstå. Publiceringsspråket som används på WWW är HyperText Markup Language (HTML).
HTML ger enligt Dave Raggett, Arnaud Le Hors och Ian Jacobs [1997] användaren möjlighet att:

- Publicera online-dokument med titlar, texter, tabeller, listor, foton etc.
- Hämta online-information via hypertextlänkar genom att klicka sig fram.
- Skapa formulär för att genomföra transaktioner med applikationer var som helst i världen, t.ex. för användning i sökning av information eller att beställa produkter.
- Inkludera videoklipp, ljudklipp och andra applikationer direkt i dokumentet.

För att det ska gå att koppla formulär mot någon funktionalitet måste det till någon form av programmering bakom själva HTML dokumentet, och den brukar oftast vara någon form av CGI-skript [se nedan], t.ex. ett Perl-skript.

Dave Raggett et.al. menar att de flesta borde hålla med om att HTML dokument borde fungera bra över olika webläsare och plattformar. Att åstadkomma kompatibilitet sänker kostnaderna för utvecklingen av WWW-lösningar eftersom man endast behöver utveckla en version av dokumentet. Om detta inte eftersträvas finns risken att WWW kommer att utvecklas till en värld av inkompatibla format, vilket tillslut reducerar WWWs kommersiella potential för alla deltagare. I varje ny version av HTML har man lagt stor möda på att de inkludera de nya funktioner som utvecklarna tagit fram. HTML har utvecklats med visionen att alla typer av informations-teknologi skall ha möjlighet att använda information på WWW: PCs av alla olika märken och prestanda, mobiltelefoner, datorer med hög och låg bandbredd och så vidare.

6.2.2 CGI (Common Gatway Interface)

Ett statiskt dokument skapas i princip en gång och förändras sedan bara då författaren gör manuella ändringar [Ericsson, 1997]. Ett dynamiskt dokument förändras över tiden, antingen varje gång det används, under tiden det används, eller med jämna intervall (t.ex. 3 ggr/dag). Ordet "dynamik" kan tolkas på flera olika sätt menar Ericsson vidare. Gemensamt för dessa är dock att de avser något som är föränderligt, snarare än konstant.

Dokumentet kan skapas då användaren begär det (baserat på data från användaren).
Ett exempel på sådan information är resultatet från en sökning via en "sökmaskin", där användaren skickar en "fråga" till en server, och får ett svar, som beror av frågan, tillbaka.

Informationen som ligger till grund för sådana svar är uteslutande lagrade i andra format än HTML enligt Mikael Ericsson. Anledningen att välja ett annat format för lagringen är enkel: HTML är varken avsett för eller lämpligt för direkt uppdatering, förändring eller sökning; det är i

många avseende för begränsande och svårhanterat. Därför väljer man hellre att hantera det innehåll som skall förmedlas i form av t.ex. databaser (t.ex. Oracle, FileMaker, Access eller DB2) och genererar HTML-dokument när det behövs.
Det finns många olika situationer och uppgifter där statiska dokument inte "räcker till". Viss information måste hållas "fräsch" för användarna varje gång de läser ett dokument. Vissa uppgifter kräver att användaren matar in information själv, och först därefter skapas ett dokument som visas för användaren.

CGI är ett samlingsnamn för program som exekveras på servern där de ligger, och klienten ser enbart resultatet av denna exekvering. Detta innebär att man slipper de långa laddningstider som förknippas med t.ex. Java, men istället blir man helt beroende av hur snabb förbindelse man har, samt hur snabbt servern kan exekvera programmet.

6.2.3 Javascript

De flesta webläsare kan använda sig av ett flexibelt och lätt programmeringsgränssnitt via skriptspråket Javascript. Det ger möjligheter till att över plattformsgränserna skapa skript för händelser och objekt. Det ger författaren möjligheter att kontrollera händelser som uppstarter, avslut och då användaren trycker på musknappar. Det är löst baserat på språket Java men är till skillnad från Java inget “riktigt“ programmeringsspråk [Flanagan & Shafer, 1998]. Därför kan det upplevas som något förvirrande att de har liknande namn. Det är främst avsett för att skapa snygga grafiska effekter, som t.ex. knappar som ändrar färg när man rör musen över dem, men det går även att skapa mer avancerade saker som formulär och liknande. För den som redan kan HTML är det ett relativt relativt enkelt språk att lära sig.

6.2.4 Active X

Active X är Microsofts eget koncept för att framställa interaktiva websidor [Tall & Ginsburg, 1996]. Active X är ett paraplybegrepp, dvs spänner sig över ett stort område. Java är ett programmeringsspråk medan Active X är en plattform för komponentbaserad utveckling. Som läget är nu så stödjer endast Internet Explorer detta till 100 procent, vilket inte är konstigt då det är en produkt från
Microsoft. I jämförelse med t.ex. Java så finns det uppenbara säkerhetsrisker med Active X. En Active X komponent kan enkelt programmeras för att t.ex. radera användarens hårddisk.

6.2.5 Java

Java har blivit ett av de mest omtalade programmeringsspråken på senare år. Sun, som lanserat språket, har skänkt ut det gratis för att på så sätt snabbt kunna sprida det över världen.

Till skillnad från de ovanstående är Java ett komplett programmeringsspråk. I dag används språket främst för WWW-utveckling och det har därför fått en uppsjö med funktioner som är speciellt anpassade för just web-hantering [Eckel, 1998].

6.2.6 Motivering till val av verktyg

Eftersom applikationen skulle migreras till intranätet och skulle använda sig av en webläsare som gränssnitt måste vi använda oss av någon form av HTML. Att bara skriva vårt program i HTML hade inte fungerat eftersom det språket inte erbjuder den funktionalitet vi krävde. Med HTML kan man bara skapa statiska sidor och man kan inte med enbart HTML kod kommunicera med andra resurser så som databaser och applikationer. Då användaren interagerar med applikationen skall data presenteras grafiskt. Användaren vill, enligt företaget, oftast bara se en liten del av all den data som finns i databasen. Om vi bara skulle använda oss av HTML skulle vi vara tvungna att skapa applikationen på så vis att all data skulle skickas med vid varje uppstart av applikationen. Detta skulle leda till onödigt långa väntetider vid starten.
För att göra HTML dynamiskt kan man som vi redan nämnt använda sig av CGI. CGI är ett populärt sätt att lösa denna typ av problem, men även denna teknik faller till föga. Då en av förutsättningarna är att göra applikationen så lik som den tidigare versionen var, ger inte HTML tillsammans med CGI den funktionalitet som vi kräver för att applikationerna skall uppfattas som likvärdiga både utseende- och funktionsmässigt.
Javascript är som redan tidigare nämnts ett enkelt skriptspråk främst avsett för att göra snygga grafiska effekter på en hemsida och vi anser inte att det har den funktionalitet som vi efterfrågade när vi skulle skriva vår applikation. Trots de grafiska möjligheterna saknas möjligheterna till koppling till andra resurser så som databaser.
Active-X fungerar bara tillsammans med webläsaren Internet Explorer från Micro$oft. Eftersom detta innebär att alla som använder andra webläsare än Microsofts egen inte kommer att kunna köra applikationen bestämde vi oss på ett tidigt stadium för att bortse från denna teknik.
Vi hade kunnat använda en kombination av HTML, CGI och Javascript för att bygga systemet, men i så fall hade vi kunnat få problem när det gäller säkerheten. Inget av dessa språk har något inbyggt stöd för kryptering av information som skickas över nätet, och eftersom användare skulle behöva logga på systemet med användarnamn och lösenord så skulle detta innebära att lösenorden skulle skickas i klartext över nätet. Detta är ingen bra lösning eftersom det då är enkelt för en utomstående att komma över ett giltigt användar-ID med tillhörande lösenord och ta sig in i systemet den vägen. Den del av systemet vi valde att implementera innehöll som sagt inga topphemliga data, men inom en nära framtid planerar man att implementera hela LIS, och då kommer dessa viktiga säkerhetsaspekter in i bilden.

Efter att ha avfärdat alla de andra teknikerna återstod bara Java. Till skillnad från Javascript och Active X är det ett riktigt programmeringsspråk, samtidigt som det har alla de grafiska funktioner vi behöver för att kunna bygga applikationen så att den liknar den gamla så mycket som möjligt. Att bygga client/server-applikationer i Java är enligt våra egna erfarenheter inga större problem, och vi ansåg oss därmed kunna bygga applikationen helt i enlighet med de kriterier vi ställt upp i vår hypotes. Samtidigt finns det inbyggt i språket ett stöd för kryptering och dekryptering av data som ska skickas över nätet.

Utöver dessa fördelar har Java också en hel del andra egenskaper som gör det lämpat som utvecklingsverktyg för denna typ av applikationer [Flanagan, 1997]:

1. Plattformsoberoende
2. Multitrådat
3. Objektorienterat
4. Anpassat för WWW

1. Plattformsoberoende

Att språket är plattformsoberoende innebär att det ska gå att köra på vilken maskin som helst som har en Java-tolk. Tanken är att man bara ska behöva skriva sina applikationer EN gång och sedan kunna använda dem på vilken plattform som helst. Detta uppnår man genom att språket först kompileras till sk. "Byte-kod" som är maskinoberoende, dvs den blir likadan oavsett om man kompilerar sin kod på en Unix-maskin, en PC eller en Mac. Denna kod kompileras i sin tur av den maskin som ska köra programmet till kod som maskinen kan förstå. På detta sätt kan alltså alla maskiner i teorin köra samma program, något som företag med många olika miljöer kan tjäna stora pengar på. Ericsson räknar t.ex. med att de har sparat in halva utvecklingskostnaden för Tacos, ett ärendehanteringssystem som byggdes i Java, genom att inte behöva utveckla separata versioner för PC och Unix [Wallström, 1998].

2. Multitrådat

Multitrådning innebär att språket klarar av att hålla flera processer igång samtidigt, genom en form av time-sharing, dvs att processorn ägnar en mycket kort tid åt varje process i tur och ordning, vilket får till följd att varje process upplever det som att den har hela processorn för sig själv. På detta sätt kan man t.ex., som i vårt fall, ha en server som samtidigt kan hantera en stor mängd klienter utan att det uppstår några krockar. Vi ansåg att detta var en mycket stor fördel eftersom den nuvarande applikationen används av många användare samtidigt och denna egenskap vill vi även behålla efter migreringen.

3. Objektorienterat

En av de stora fördelarna med objektorientering är som bekant att det är enkelt att återanvända kod [Skansholm, 1996]. Om det nu visar sig att försöket slår väl ut och vårt projekt blir lyckat så kommer någon att få fortsätta bygga vidare på systemet för att till slut implementera hela LIS. Eftersom vi har använt ett objektorienterat språk så underlättar detta för den eller dem som ska vidareutveckla vår applikation.
Vår applikation är inte någon avancerad konstruktion, utan ganska lätt att begripa sig på, men när systemet väl är färdigbyggt så kommer det att bli ganska stort. Då underlättar det att språket är objektorienterat eftersom man då kan använda sig av inkapsling (information-hiding) [Skansholm] för att endast visa de delar av ett objekt som är intressanta för andra objekt. Man kan alltså använda sig av en sorts "black-box"-teknik, där man som programmerare inte behöver

bry sig om vad som händer inne i ett objekt, utan kan koncentrera sig på vad man skickar in och vad som kommer ut.

4. Anpassat för WWW

Java har vunnit större delen av sin popularitet mycket därför att det är anpassat för Internet / intranät. Plattformsoberoendet innebär att man kan skriva en liten applet för sin hemsida som sedan alla (i teorin) kan använda, oavsett vilken typ av dator man har när man surfar. Vidare innehåller språket många roliga funktioner och effekter som är speciellt framtagna för att underlätta arbetet med att skapa web-applikationer, om det nu är det man är intresserad av att göra. Samtidigt finns det en uppsjö av mer "seriösa" tillämpningar, som t.ex. RMI och Corba [se nedan], som bidrar till att språket blivit såpass populärt som det faktiskt är, trots alla sina barnsjukdomar (mer om dessa senare).

Utöver dessa tekniska fördelar har Java även en del andra intressanta egenskaper som bidrog till att vi såg det som det optimala verktyget för vårt projekt:

Enligt en färsk undersökning gjord av IDC & Giga Information Group presenterad i Computer Sweden [Wallström, 1998] är Java det språk som växer snabbast i världen just nu. Störst är det i USA, med Skandinavien på en klar andraplats. Enligt denna undersökning håller idag 45% av de intervjuade företagen (i USA) på med, eller utvärderar en Java-satsning, vilket är en ökning med 11 procent jämfört med förra året. En stor anledning till Javas popularitet anser man i artikeln är det faktum att många programmerare redan kan C++, och Java har ärvt mycket av sin struktur och syntax från detta språk. Detta är en stor fördel även för oss eftersom det underlättar för andra programmerare när de vill gå in och uppdatera eller ändra i vår kod.

6.2.7 Beskrivning av tänkbara lösningar

När vi väl bestämt oss för att använda Java som utvecklingsspråk återstod frågan hur vi rent tekniskt skulle gå till väga. Eftersom vi valt att använda Java fanns det tre olika sätt vi kunde välja att gå till väga för att implementera systemet:

1. Corba
2. RMI
3. Sockets

1. Corba

Corba (Common Object Request Broker Architecture) är egentligen ingen fristående produkt, utan det utgör en koppling mellan program över ett nätverk, som ett intranät, eller över Internet. Man kan tänka sig Corba som ett hav som förenar en massa små öar i ett litet ö-rike. När öborna har lärt sig att bygga båtar så kan de enkelt färdas över vattnet till varandra [se figur 9]. Men Corba är mycket mer än så; När man använder Corba för att kommunicera mellan t.ex. ett C++ program och ett Java-program så jobbar de respektive programmerarna enbart inom sitt eget

språk och behöver därför inte bry sig om hur man ska "översätta" datan som skickas för att den ska förstås av det andra programmet. Det sköter Corba om självt [Curtis, 1997].

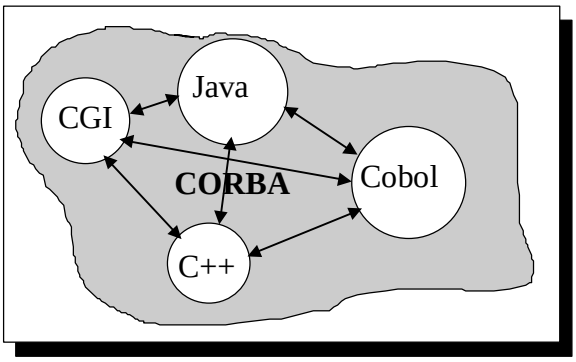

Fig. 9: Corba

2. RMI

RMI (Remote Method Invocation) kan man säga är Suns eget svar på Corba. Skillnaden är att det endast är ett Java till Java interface. Där Corba kan liknas vid ett hav som förbinder ett antal öar, kan RMI mer liknas vid en järnväg, som förbinder två orter. Det går att få två applikationer skrivna i olika språk att kommunicera via RMI, men då måste man ha en Java-"anpassare" (Java adapter) som ligger tillsammans med applikationen skriven i det främmande språket, som tar hand om dess utdata och konverterar den till något format som kan förstås av Java-applikationen i andra änden. [se figur 10] [Curtis]

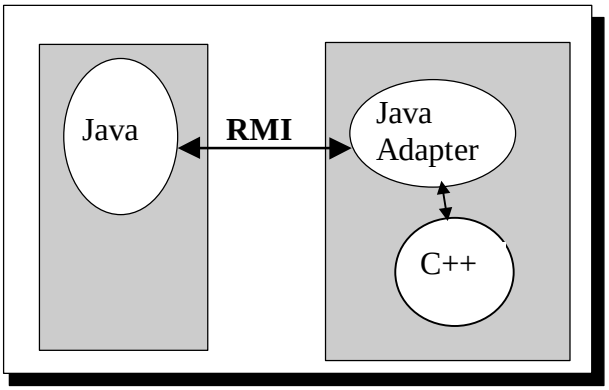

Fig. 10: RMI

3. Sockets

Ett tredje alternativ är sockets. En socket kan liknas vid en telefonförbindelse som tillåter två processer att tala med varandra. Till denna socket kan man skriva text-strängar, precis som till en vanlig fil. Skillnaden är bara att där en fil lagras på disk eller i minnet på den maskin där man exekverar koden, så skickas det man skriver på socketen över till maskinen i andra änden av förbindelsen, som i sin tur läser den precis som om den kom från en lokal fil [Flood, 1997].

En annan stor fördel med sockets är att det går oerhört snabbt. Eftersom man bara skickar ren t.ex.t så blir det i vårt fall aldrig mer än ett par kilobytes som skickas och över ett intranät med många megabits per sekund i överföringshastighet så blir dessa överföringshastigheter försumbara [Flood].

Den tredje stora fördelen är att sockets är en gammal beprövad teknik. Sockets har använts sedan datorernas barndom och är därför mycket pålitliga [Flood].

6.2.8 Val av teknisk lösning

Då vi analyserade den befintliga applikationen och hur vi skulle kunna migrera den fann vi framför allt två faktorer som var av stor vikt vid valet av teknik:

- Det var ett litet begränsat problem.
- Data skulle inte ändras utan bara läsas.

Corba är, som vi sagt tidigare, ett sätt att skicka objekt mellan applikationer och vi ansåg att lära sig detta separata verktyg skulle ta onödigt lång tid för att vara motiverat. Dessutom vet vi av egen erfarenhet att en RMI-lösning endast kommer till sin fulla rätt när man arbetar med stora system, där hundratals objekt ska kommunicera med varandra över nätet. Corba och RMI är dock båda mycket bra lösningsalternativ om appikationen skulle vara större och mer komplicerad. Skillnaden mellan Corba och RMI och deras för- respektive nackdelar är i sig ett område stort nog för en magisteruppsats och för den intresserade så hänvisar vi till just ett sådant arbete [Duplancic & Lindberg, 1998]. För vår del räcker det att säga att en lösning där Corba eller RMI hade ingått skulle blivit mycket mer avancerad än vad nöden kräver
Sockets däremot är en relativt enkelt lösning som finns i det flesta språk. Vi ansåg att detta vore det enklaste och snabbaste sättet att lösa detta begränsade problem.

6.3 Konstruktionen av systemet

Vi började med att lägga upp en plan för hur vi skulle disponera arbetet och vi anser att vi följde den mycket bra.
Den såg ut på detta sätt:

- Analys 1 vecka
- Design 1 vecka
- Konstruktion av systemet 6 veckor
- Uppdateringar och rättande av fel 2 veckor.

En av våra första tankar var att bygga en tvåskiktslösning med en direktkoppling mellan Applet-klienten och stordatorn. Detta visade sig dock omöjligt eftersom stordatorns operativsystem inte hade något stöd för Java. På grund av säkerhetsproblem är Java designat så att en Applet endast kan koppla upp sig mot samma server den laddas ifrån. Detta för att man inte ska kunna länka vidare från en Applet till en annan sida och ställa till med problem för någon annan. Efter sommaren skulle man installera en ny version som då även kommer att ha detta Javastöd vilket innebär att man då kommer att kunna bygga en tvåskiktslösning. Vi bestämde oss alltså för att bygga en treskiktslösning med en serverdel som vi lade på den Unix-maskin som går som intranät-server för företaget.
På detta sätt kom vi också ifrån den inneboende oviljan hos företagsledningen att koppla stordatorn direkt till nätet. Denna ovilja är inget som är specifikt för vårt företag. I en artikel i Computer Sweden [Eriksson, 1998] skriver Mikael Eriksson om hur storföretagen idag är mycket rädda för att koppla sina stordatorer till Internet, trots att det enligt IBM som bygger maskinerna inte finns några som helst hinder, varken säkerhetsmässigt eller prestandamässigt. Istället menar James West, Europachef för IBM i samma artikel, att det enbart rör sig om en psykologisk spärr hos företagsledningarna.

Efter att vi analyserat och gjort ett design förslag upptäckte vi under själva konstruktionsfasen att en del av våra idéer och förslag skulle vara ogenomförbara. Bl.a. används i LIS en trädstruktur för att representera hur de olika kontoren är indelade i distrikt som i sin tur är indelade i områden. Denna trädstruktur ville vi gärna behålla eftersom den är en av grundpelarna i det gamla systemet och vi ville få det nya att bli så likt det gamla som möjligt. Problemet var bara att i dagsläget så ingår inte denna typ av komponent i Java. Vi ställdes alltså inför valet att antingen bygga en egen rutin för att hantera denna typ av strukturer, eller också försöka hitta en från ett tredjepartsföretag. På grund av tidsbristen valde vi det senare och hittade en lösning hos Symantecs Visual Café vilken vi importerade. Då upptäckte vi ett nytt problem. Trots att denna rutin var skriven för att fungera i alla webläsare så fungerade den inte i Microsofts Internet Explorer 3.0, som oturligt nog visade sig vara den browser som de flesta använde på företaget. Detta fick till följd att vi blev tvungna att skapa en ny version av applikationen enbart för Internet Explorer. Eftersom vi inte kunde använda Symantecs trädstruktur fick vi skapa en egen variant med ett något annorlunda gränssnitt. Kopplingen mellan servern och databasen var ganska enkel att konstruera. Den största delen gick åt att göra det grafiska gränssnittet så bra och likt det gamla systemet som möjligt. Vi anser dock att vi hölls oss mycket bra till den tidsplan vi lagt upp. Det största problemet är att olika webbläsare tolkar Java koden olika.

6.4 Beskrivning av det nya LIS

Det nya LIS vi byggde fungerar på följande sätt:
När någon är intresserad av dagstatistiken går han eller hon till en web-sida på företagets intranät. Här ligger den Java-applet (Se figur 11: 1) som vi byggt, och som fungerar som klienten i vårt system. Där väljer användaren precis som tidigare vilket område eller kontor man är intresserad av och trycker på ”uppdatera”-knappen. Då skickas en begäran via intranätet till företagets intranät-server (2) som ligger på en Unix-maskin, där den behandlas och skickas vidare till stordatorn (3) som plockar ut de efterfrågade uppgifterna ur databasen (4) och returnerar dessa till

Unix-servern. Från intranät-servern skickas dessa uppgifter ut till den aktuella applet-klienten där de visas i form av ett diagram.

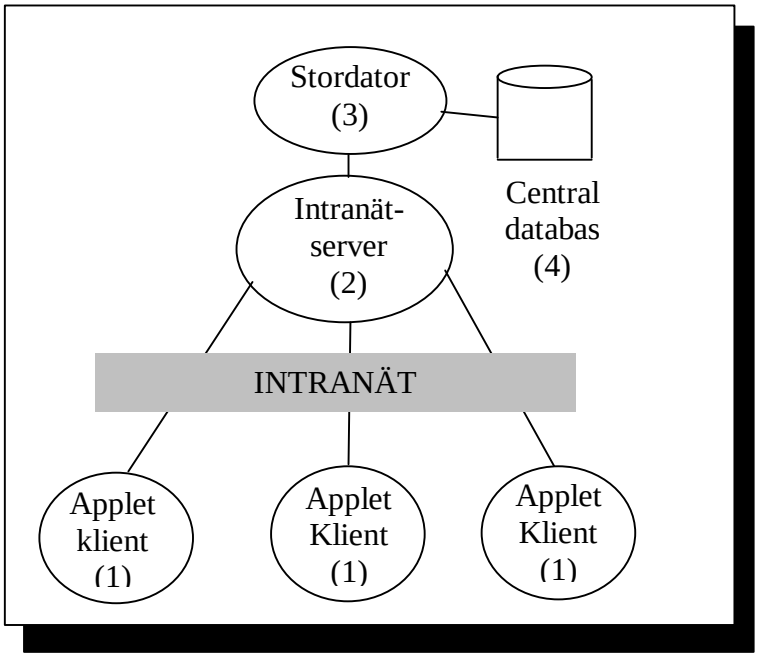

Fig. 11: Strukturen hos det migrerade LIS

När vi migrerat LIS till intranätet hade det följande egenskaper:

Treskikts client/server:

Att vi byggde en treskiktslösning berodde på ett antal faktorer:
Som företagets datormiljö ser ut idag så hade vi inte kunnat bygga en tvåskiktslösning.
Operativsystemet för stordatorn stöder i dagsläget inte Java, men detta är något som kommer att finnas från och med nästa release som kommer någon gång i sommar.
Genom att bygga en treskiktslösning kom vi också ifrån problemet med företags ovilja att koppla sina stordatorer direkt till nätet [Eriksson, 1998].

Genom att ha en server baserad på en Unix-maskin uppnår vi en mycket hög grad av tillförlitlighet. Unix-datorer är som bekant sedan länge kända för sin höga pålitlighet, och detta kan vi nu dra nytta av.
I dagsläget har vår server varit igång i tusentals timmar utan att visa tecken på problem.

Tunna klienter

Den klient vi byggde för systemet låg i storlek på ungefär 20kB, och i sammanhanget är det att betrakta som mycket tunt. Genom att bygga den på detta sätt försäkrar vi oss om att nedladdningstiden minimeras vilket förhoppningsvis kommer innebära att applikationen

uppfattas som snabb och smidig vilket var ett av huvudsyftena med att flytta applikationen till intranätet.

Central data

Nu använder sig applikationen av datan som ligger lagrad direkt på stordatorn. På detta sätt sparas mycket tid jämfört med tidigare då datan skickades ut till de olika regionskontoren via FTP varje natt, oavsett om någon var intresserad eller inte. Nu fungerar systemet istället mer i enlighet med Telleens "pull"-mentalitet, dvs kunden måste själv efterfråga informationen innan han får den.

Tillgänglighet

Genom att applikationen nu ligger på företagets intranät kan alla som är behöriga komma åt informationen. Samtidigt blir det mycket enkelt för nya användare att börja navända applikationen. Tidigare var man tvungen att få ut en tekniker som installerade programmet, men nu räcker det att man som ny användare får URL:en till den HTML-sida där Appleten ligger.

Likhet med det gamla systemet

En av grundtankarna med vår flytt av systemet var att det skulle vara så likt det gamla som möjligt. Vi gjorde ett par kosmetiska förändringar men lämnade huvudfunktionerna intakta vilket fick till följd att alla som använt det gamla systemet kände igen sig.

6.5 Företagets reaktioner:

På företaget fanns som vi redan nämnt en öppenhet för nya idéer och tekniker, men trots denna öppenhet verkar det fortfarande finnas ett stort motstånd hos företagsledningen att öppna upp denna typ av informationskällor för alla på företaget. Detta verkar vara något som går igen hos företagsledningar över hela världen [Telleen, 1996].
Av egen erfarenhet vet vi nu att ett företags ledning ibland har svårt att inse vilka uppenbara fördelar de kan vinna genom denna nya typ av system. Istället för att se fördelarna med systemet fokuserade man på hur mycket det kommer att kosta företaget att ha anställda som sitter och "surfar" på företagets intranät.
Från företagets sida var man mycket nöjd med applikationen. Samtidigt kom man med invändningen att om alla på företaget skulle få tillgång till systemet så skulle de varje dag ägna två minuter åt att se hur det hade gått dagen innan, och två minuter multiplicerat med antalet anställda multiplicerat med antalet kronor per minut de anställda får i lön skulle innebära kostnader på tusentals kronor varje dag, och detta kunde man inte tillåta.
När man resonerar på detta sätt anser vi att man har missat ett flertal viktiga faktorer:

- Lägesrapporter
- Tävlingsmomentet
- Utbildningsfaktorn

6.5.1 Lägesrapporter

Alla som gjort lumpen vet att det är mycket påfrestande att inte veta vad som pågår. Därför fick man redan tidigt lära sig att det är mycket viktigt att regelbundet upplysa sina soldater om läget, även om det inte har hänt något. Att veta att det inte har hänt något är mycket bättre än att inte veta något alls, för då är risken stor att man börja fantisera ihop skräckscenarion och det hela slutar med att man får magsår. Detta är inte någon företeelse specifik för militära förhållanden, utan något som gäller generellt för hela vårt samhälle. Att som anställd ha en chans att se hur man ligger till jämfört med tidigare perioder är mycket viktigt för att förstå ledningens handlande, som annars kan tyckas vara helt okontrollerat. Om man själv kan se att försäljningssiffrorna har rasat sedan förra månaden är sannolikheten större att man fogar sig när ledningen kommenderar helgarbete för att komma ikapp.

6.5.2 Tävlingsmomentet

Alla människor är tävlingsmänniskor på någon nivå, och det gäller även anställda på företag. Oavsett vad man håller på med så kommer folk alltid att vilja tävla mot sig själva och andra och som företagsledare är det mycket dumt att inte dra fördel av denna inneboende egenhet hos människan. Om man som anställd kan se sitt resultat i klara siffror kommer man ohjälpligen att hamna i en tävlingssituation med sig själv. Man vill alltid se om man kan slå rekordet på 45 behandlade fakturor på en minut, eller om man svarva till hjulet med en felmarginal på under två millimeter. På detta sätt kan man höja produktiviteten hos de anställda utan att de kommer och vill ha högre lön, eftersom de gör detta extraarbete av egen fri vilja.

6.5.3 Utbildningsfaktorn:

En av de stora fördelarna med ett intranät är att användarna snappar upp nyttig kunskap om företaget genom att bara “lattja runt“ [Telleen, 1996]. Om man låter de anställda “leka“ lite med företagets intranät i sin egen takt tror vi att de i mycket högre grad kan tillgodogöra sig den information som finns där än om man hade tvingat dem att sitta där under en noga övervakad kurs.

Vi valde till slut i samråd med beställaren att lägga upp applikationen på en olänkad sida. Dvs. att man inte kan komma åt sidan om man inte kan den exakta URL-adressen till den. Denna adress har sedan väl utvalda personer på företaget fått för att testa systemet. Efterhand som folk inser fördelarna med systemet så tror vi att användandet kommer att spridas automatiskt.

7 SLUTSATS

I vår hypotes lade vi fram teorin att en applikation ska ha följande egenskaper för att vara lämplig för en migrering till ett intranät:

- Applikationen skall ha en client/server-arkitektur [se 4.5.2]
- Data som används av flera användare finns i flera fristående kopior [se 4.5.5]
- De användare som är beroende av applikationen kommer inte åt den [se 4.5.5]

Vi valde att testa denna hypotes genom att själva välja ut en applikation som uppfyllde dessa krav och sedan migrera den till ett företags intranät med hjälp av web-teknik. Vi valde att utföra försöket på Sema Group eftersom vi ansåg att de hade rätt inställning till denna typ av utveckling. Applikationen vi valde att flytta (LIS) hade de två första egenskaperna. Bland alla de olika tekniker som stod till vårt förfogande valde vi att använda oss av Java eftersom det innehöll all den funktionalitet som vi behövde, samt att det även har stöd för krypterad kommunikation över nätverk. Vi valde vidare att bygga systemet med hjälp av sockets, i stället för med RMI eller Corba, eftersom det var ett litet och begränsat system vi skulle bygga. Konstruktionen av systemet gick relativt smidigt, med ett par undantag beroende på barnsjukdomar i språket och det faktum att Microsofts webb-läsare Internet Explorer inte är 100% kompatibel med Java-standarden. Trots detta var man på företaget mycket nöjda med applikationen. Vi anser således att migreringen var lyckad, och att vi därför kan hävda att de egenskaper som vi anser att en applikation ska ha för att vara lämpad för en migrering till ett intranät är relevanta. Att applikationen har en client/server-struktur redan från början innebär att man enkelt kan överföra den till webb-miljö eftersom Internet/intranät-tekniken i grund och botten är baserad på client/server-tänkande. Genom att lägga resurser som t.ex. databaser på en central server kommer man ifrån de problem som redundans i datan innebär, och på intranätet kommer alla anställda som har tillgång till nätet åt applikationen. Naturligtvis innebär inte en centralisering av resurser alltid en förbättring, men vi menar ändå att för just denna typ av applikationer överväger fördelarna nackdelarna. I andra specifika fall är det naturligtvis upp till företagen själva att avgöra vilken lösning som är bäst för dem.

8 KRITISK GRANSKNING:

Kan man fästa någon större trovärdighet vid vår uppsats? Nedan följer ett par saker man skulle invända mot det vi har skrivit, och vi har försökt att besvara denna kritik utifrån våra egna åsikter och erfarenheter.

Att vi har visat att vår hypotes stämmer behöver inte innebära att de egenskaper som vi nämnt som viktiga för en applikations lämplighet att flyttas till ett intranät är de enda som spelar in.

Naturligtvis inte. Men avsikten med denna uppsats var inte att skapa ett heltäckande regelverk. De egenskaper vi har tagit upp i detta arbete *är* mycket viktiga, och det har vi också påvisat i detta arbete.

Borde vi inte ha genomfört en grundlig enkätundersökning bland landets större IT-företag?

Det finns ett flertal nackdelar med denna typ av undersökningar som gjorde att vi avstod från att utföra en enkätundersökning. Först och främst hade vi en mycket begränsad tid att röra oss med, vilket fick till följd att vi inte kunde utföra alla de moment som vi hade önskat. Att sätta ihop ett statistiskt giltigt frågeformulär är ingen enkel uppgift, utan snarare något som hade kunnat vara målet för ett helt eget examensarbete.
Som vi redan nämnt är många företag tveksamma till att lämna ut information om sina affärsstragegier. Eftersom ett intranät kan erbjuda stora konkurrensfördelar om man lyckas med sin satsning så uppfattas de som affärshemligheter som vill undanhålla från sina konkurrenter, och vi ansåg därför att chansen till att få tillräckligt många och detaljerade svar för att kunna basera ett examensarbete på dem var relativt liten. Vi valde alltså att själva bygga ett system och redogöra för våra egna erfarenheter och åsikter.
Det enda vi saknar på den här punkten är det faktum att en större undersökning hade kunnat ge oss fler infallsvinklar till problemet. Att vi bara koncentrerat oss på *ett* företag kanske har inneburit att vi har missat en del intressanta aspekter som vi kanske hade uppmärksammat om vi hade gjort en ordentlig förundersökning.

Uppsatsens resultat och slutsatser baseras till största delen på våra egna slutsatser. Är de verkligen trovärdiga?

Självklart väger inte våra åsikter lika tungt som en professors, men vi vågar ändå hävda att de har ett visst mått av sanning i sig. Efter fyra års studier på Systemvetarprogrammet så anser vi oss vara kvalificerade att kunna dra egna slutsatser och komma med förslag om hur en applikation bör se ut för att vara lämpad att flytta till ett företags intranät.

Eftersom det ämne vi studerat är såpass nytt har det inte hunnit skrivas så mycket om liknande projekt. Vi blev således mer eller mindre tvungna att söka vår egen väg. Detta har både varit en förbannelse och en välsignelse, och som vi sade i inledningen: Vi har upptäckt att kartritare och upptäcktsresande är mycket underskattade yrkesgrupper.

De delar av rapporten som inte utgörs av egna slutsatser och åsikter baseras till 99% på amerikanska undersökningar. Kan man verkligen överföra dessa rakt av till svenska förhållanden?

Som vi redan nämnt finns det inte särskilt mycket skrivet om vårt ämne sedan tidigare. Därför har vi blivit tvungna att plocka delar ur många arbeten för att få fram ett underlag för vår rapport. Tyvärr har det vad vi kunnat se inte skrivits mycket alls om svenska förhållanden, något som vi naturligtvis beklagar. Vi har alltså blivit tvungna att ta skeden i vacker hand och använda det som erbjuds, dvs. främst amerikanska forskningsrapporter. Under arbetets gång har vi försökt att vara kritiska mot det vi läst och försökt överföra det till våra svenska förhållanden, men det är ganska troligt att vi har misslyckats någonstans på vägen och använt oss av rent amerikanska värderingar och åsikter som universellt giltiga.

Kan man verkligen hävda att det är ett paradigmskifte på gång inom intranät-tänkandet?

Att det är tekniken som har gått såpass mycket framåt de senaste åren att den tillåter de nya användningsområden för intranät som vi målat upp i denna uppsats är ganska uppenbart. Kan man då tala om ett paradigmskifte i egentlig mening? Vi anser att det vi börjat ana i företagens sätt att se på sina intranät är så radikalt skilt från det tidigare synsättet att vi anser det befogat att tala om ett paradigmskifte. Att gå från vad Telleen kallar en push-mentalitet där man mer eller mindre mekaniskt förser användarna med en massa information som de kanske inte vill ha, till en pull-mentalitet där användarna själva kan välja vilken information de är intresserade av, genom att utveckla en helt ny typ av applikationer för intranät anser vi vara ett tillräckligt stort steg för att kunna tala om ett paradigmskifte.

Är Java verkligen det optimala språket för intranät-applikationer?

Både ja och nej. Som bekant är Java i dagsläget inte så stabilt som man skulle kunna önska. Denna uppfattning delar vi med de flesta andra utvecklare [Wallström, 1998] och det är enligt samma artikel en allmän uppfattning att Java först om fem år kommer att vara stabilt nog för att utveckla affärskritiska applikationer i. Detta är en kalkylerad chansning från vår sida, men vi tror ändå att vår applikation är tillräckligt liten och enkel för att fungera smärtfritt. Baserat på våra egna erfarenheter kan vi inte idag rekommendera någon att bygga system i Java, annat än som ren försöksverksamhet eller i mycket liten skala. Att språket i sig är nytt och instabilt hade man kanske kunnat ha överseende med, men när sedan web-läsarna inte klarar av det de ska, tappar man förtroendet för språket. Mycket av denna problematik beror på politiska motsättningar mellan de stora mjukvaruföretagen. Främst är det Microsoft som vägrar att ansluta sig till Javastandarden och i stället skapar en egen variant av språket som inte är 100% kompatibel med andra tillverkares produkter. På samma sätt klarar inte Microsofts web-läsare Internet Explorer av

alla de funktioner som finns i Java-standarden, vilket för vår del ledde till att vi blev tvungna att skriva en specialversion för den läsaren. Så länge man inte kan enas om en standard så kommer språket inte att finna någon större publik hos kunderna.

9 FRAMTIDA FORSKNINGSPROJEKT

Efter att ha skrivit denna uppsats har vi upptäckt att vi hittat fler nya frågor än dem vi svarat på. Nedan följer några tänkbara fortsättningar på vårt arbete:

Hur ska man bära sig åt för att kunna anpassa applikationer för ett intranät?

Om de applikationer man vill överföra till sitt intranät inte uppfyller de krav vi ställt upp i denna uppsats, hur ska man då bära sig åt för att anpassa dem så att de går att flytta över? Man skulle kunna tänka sig att mer i detalj gå in på fördelar med modularisering av system, eller undersöka om det finns några helt andra vägar att lösa problemet.

Hur kommer de nya tekniker som är på gång att påverka detta område?

Redan idag finns det en del andra tekniker för att bygga denna typ av applikationer. Vilka fördelar / nackdelar har de jämfört med Java? Exempel på nya tekniker att undersöka kan vara Microsofts ASP, dynamisk HTML och andra liknande produkter.

En mer grundlig jämförelse mellan de olika tekniska lösningsalternativen

Vad hade vi kunnat göra bättre om vi hade valt någon av de andra tänkbara lösningsalternativ vi målat upp i denna uppsats? Vad hade vi inte kunnat göra, osv.

Vad kommer denna utveckling att ställa för krav på framtidens intranät-hårdvara?

När intranäten blir hårdare belastade så kommer kraven på hårdvaran att öka. Vad får detta för följder för företagen? Kommer de att satsa pengar på intranät-utveckling eller kommer de helt enkelt att avstå från denna utveckling därför att det blir för dyrt att köpa nya maskiner?

Kommer denna utveckling att leda till att vi kommer att få se mer RMI och Corba lösningar i framtiden?

RMI och Corba är redan idag ganska etablerade på marknaden. Om den utveckling som vi påvisat i denna uppsats får fortsätta, kommer det då att innebära att andra aktörer med egna produkter kommer att försöka slå sig in på marknaden, och vad händer i så fall med dagens jättar? Kan de anpassa sig eller kommer de att gå under?

Hur ser tillvägagångssättet ut när man nyutvecklar applikationer för intranät?

Vi har koncentrerat oss på att studera hur man kan flytta befintliga applikationer till sitt intranät, men hur ska man bära sig åt för att utveckla helt nya applikationer direkt för sitt intranät? Hur skiljer sig denna process från den vi beskrivit, och vilka likheter / skillnader har den jämfört med "klassisk" systemdesign? Det här kommer antagligen att bli nästa stora förändring på intranät-marknaden. Ett nytt paradigmskifte kanske?

Om man istället hade satsat på feta klienter, hade man då kunnat bygga andra typer av applikationer?

Hade kriterierna för en applikations lämplighet att flytta till ett intranät varit annorlunda om vi istället hade satsat på att ha feta klienter istället för tunna?

Finns det fler faktorer som är avgörande för en applikations lämplighet att flyttas till ett intranät?

Antagligen. Vi har i denna uppsats valt att fokusera på dem vi såg som viktigast, men det finns antagligen en uppsjö med andra egenskaper hos program som också spelar in när man ska välja vilka applikationer man ska flytta till sitt intranät.

Är intranät något som kommer användas i framtiden?

Har intranät-fenomenet kommit för att stanna? Idag tror de flesta forskare på området att så är fallet, men kan det inte finnas andra sätt att göra samma sak, eller kan det vara så att den väg vi har sett för intranätens utveckling är helt felaktig? Det kanske finns något annat område där man ännu mer framgångsrikt kan applicera denna teknik?

Varför har ingen skrivit en rapport om det här området tidigare?

Om nu företagen är så intresserade som de säger, varför har då ingen redan gjort en undersökning som vår? Kan det vara så att företagen inte vill dela med sig av sina erfarenheter därför att de hoppas på konkurensfördelar om de sitter och håller på dem? Hur många företag håller idag på med utveckling av interaktiva intranät-applikationer?

Vad kan man mer göra för att få företagsledningarna att inse fördelarna med intranät-tekniken?

Hur ska man få företagsledningen att bli mer intresserade av denna teknik? Man skulle kunna tänka sig en större intervju med företagsledare för att se vad de vill höra för att bli intresserade av ett projekt och sedan sätta ihop någon sorts handbok för hur andra bör gå tillväga.

Vilka organisatoriska förändringar kan man vänta sig som en följd av detta paradigmskifte?

Att sköta allt mer av sitt dagliga arbete via företagets intranät kommer naturligtvis att medföra en massa förändringar i företagets vardagskultur. Att studera detta fenomen utifrån ett organisatoriskt perspektiv skulle kunna vara mycket intressant.

Genomföra den enkätundersökning bland stora företag som vi inte hann med

Det här är en av delar vi inte hann med under vårt arbetes gång och det skulle vara mycket intressant att få veta vad andra företag anser om intranät-tekniken och vad man kan använda den till. Vad har de stora företagen för planer för sina intranät? Är projekt av samma typ som det vi genomförde något som andra har börjat dra igång, eller var det en engångsföreteelse?

Vilka andra effekter kan man uppnå genom att ge de anställda tillgång till on-line information?

Vi har tagit upp ett par tänkbara effekter av vad det kan få för följder för de anställda att de får tillgång till den absolut senaste informationen om hur det går för deras avdelning eller företaget i stort. Vilka andra fördelar skulle man kunna uppnå? Finns det några nackdelar och i så fall; vad kan man göra åt dem?

10 Referenser

Andrews Torrey. (1997). *Intranet Paper.* (Magisteruppsats, University of Houston) (http://disc.cba.uh.edu/~rhirsch/spring97/andrews1/andrew~1.html) Besökt:980305

Professor Winston A.B, Chellapa, R., & Associate Professor Baura, A.,. (1998). *Intranets: Looking Beyond Internal Corporate Web Servers.* MA: Addison-Wesley.

Curtis David. (1997). *Java, RMI and Corba - A White Paper.* *(*http://ww.omg.org/news/wpjava.html*).* Besökt:980402

Date C J. (1994). *An Introduction to Database Systems 6th Edition. S D:* Addison-Wesley Pub Co.

Duplancic Anita & Lindberg Katarina. (1998). *Technologies in Self Provisioning applications.* Institutionen för informatik vid Göteborgs Universitet.

Eckel Bruce. (1998). *Thinking in Java.* Prentice Hall Computer Books

Edquist Kent. (1998). *Lägg allt i webbläsaren.* (Computer Sweden Nr 44, fredagen 15 maj)

Ericsson Mikael. (1997). *Svenska Skoldatanätet Presentera information på WWW.* (http://www.skolverket.se/skolnet/htmlkurs/part6.html) Besökt: 980327

Eriksson Mikael. (1998). *Storföretag rädda koppla stordatorn till webben*. (Computer Sweden, 17 April, s.15)

Ewert Magnus. (1997). *Data & Telekommunikation: Från kablar till Corba*. Göteborg: Prevas

Flanagan David. (1997). *Java in a Nutshell : A Desktop Quick Reference.* O'Reilly & Associates

Flanagan David & Shafer Dan. (1998). *Javascript : The Definitive Guide 3rd Edition*. M A: O'Reilly & Associates.

Flood Wayne. (1997). *Software Engineering for Network Applications, Sockets* (http://www.cs.uwf.edu/~wilde/CEN6990/papers/flood/Flood.htm) Besökt:980306

Gaines Brian R.. (1996). *Porting Interactive Applications to the Web.* University of Calgary, Canada. (http://ksi.cpsc.ucalgary.ca/ksi) Besökt: 980222

Holtz Shel. (1996). *The Intranet Advantage*. Macmillan Computer Publishing USA.

Kalakota Ravi. (1997). *Client-Server Frequently asked questions.* University of Texas at Austin.(http://cism.bus.utexas.edu/resources/csfaq.html) Besökt:980303

Kyle Keith, Steward Eric & Martines Ian. (1997). *Communication Aspects of Intranet Usage in Business.* (University of Texas, School of Business) (http://www.utpb.edu/business/commun.htm) Besökt: 980312

Mayrs David. (1997) *History of the Internet and WWW.* (http://ourworld.compuserve.com/homepages/dmayr/history.htm) Besökt: 980212

McCullough. Dick (1998). *Qualitative versus Quantitative*. (http://edweb.sdsu.edu/Courses/Ed690DR/Class01/QvsQ.htm) Besökt: 980307

Moeller Michael. (1996). *Boeing network takes flight with pioneering intranet project*. (Pcweek Nr 2)

Nattey Joseph O. (1996). *Client/Server Technology*. Kent State University Ohio. (http://www.personal.kent.edu/~jnattey/spage7.htm) Besökt: 980512

Phanouriou Constantionos & Abrams Marc.(1997). *Transforming Command-Line Driven systems to Web Applications.* (http://www6.ntlabs.com/HyperNews/get/PAPER41.html) Besök:980223

Raggett Dave, Le Hors Arnaud & Jacobs Ian. (1997). *W3C Recommendation: HTML 4.0 Specification. (*http://www.w3.org/TR/REC-html40-971218) Besökt: 980402

Smith Patrick. (1994). *Client/Server Computing.* by Prentice Hall Computer Publishing Printed in the United States of America

Skansholm Jan. (1996). *C++ direkt.* STUDENTLITTERATUR:Sverige

Tall Eric & Ginsburg Mark. (1996). *Late Night ActiveX*. Macmillan Computer Publishing USA.

Telleen PhD Steven L. (1996). *The IntraNet Architecture: Managing information in the new paradigm*. (Utbildningsrapport) (http://amdahl.com/doc/products/bsg/intra/infra.html) Besökt: 980412

Telleen, Ph.D. Steven L. (1996). *Intranets and Adaptive Innovation.* (Utbildningsrapport) (http://amdahl.com/doc/products/bsg/intra/adapt.html) Besökt: 980512

Telleen, Ph.D Steven L.. (1996). *Intranet Methodology*. (Utbildningsrapport) (http://amdahl.com/doc/products/bsg/intra/offering.html) Besökt: 980512

Wallström Martin (1998) *Nordiska storföretag satsar på Java*. (Computer Sweden nr 37 27 april, s.4)

Wallström Martin. (1998). *SJ utvecklar plattformsoberoende*. (Computer Sweden nr. 37, 27 april 1998, s.4)

Wesley, D & Wesley, J. (1996). *Developing Real-World Intranets* Scottsdale, AZ: The Coriolis Group, Inc

Wilkerson Robert. (1996). *Intranet experiment pays off in the lab.*. (Pcweek Nr 4)

Intranets: Internet Technologies Deployed Behind the Firewall for Corporate Productivity.(1996). *Internet Society Annual Meeting*, (http://www.process.com/intranets/wp2.htp) Besökt: 980425

Tidrapportering och lönebesked via Intranät. (1998). (Computer Sweden, 4 maj)

Ebusiness Magazine. (1996). *A Close-up Look at HP's Corporate Intranet.*

Bilaga 1.

Microsofts personal får inte längre sina lönebesked hem i brevlådan. Istället har företaget satsat på ett nytt lönesystem där de anställda själva knappar in sin frånvaro och sedan får sitt lönebesked via en personlig webbsida i företagets intranät. Denna nyhet går att läsa i Computer Sweden under rubriken "Tidrapportering och lönebesked via Intranät". Microsoft har inför ett nytt lönesystem vilket skall spara på en onödigt stor administration med olika blanketter. Den anställda får själva knappa in sin närvaro och det skickas sedan till respektive chef för godkännande. Projektet har enligt Camilla Gunnarsson på Microsoft varit lyckat och systemet skall byggas ut. Kostnaden för lönesystemet och intranätlösningen beräknas ligga på mellan 100000 och 125000 kronor, dvs. En realtivt liten summa pengar i sammanhanget.

Hewlett-Packard kunde i November 1996 visa upp följande siffror kring sitt intranät skriver Ebusiness HPs egna tidning på nätet i november 1996:

- 100 000 PCs
- 23 000 Unix datorer
- 6000 Servrar
- 2500 Webservrar

HPs intranät är världens största med mer än 140 000 "värdar" som transporterade mer än 7 terabyte av data varje månad. Bara vid beräkning av underhållningsbesparingar har HP sparat $25 000 000 med mer än 100 000 PCs som man måste ge support.

Införandet av ett intranät visade sig vara ett smart experiment vid **Los Alamos National Laboratory** (LANL), då det sparade ungefär $500 000 i utskrifts och distributions kostnader under det första året skriver Robert Wilkerson i sin artikel "Intranät experiment pays off in the lab" i Pc week April 1996. Forskare har nu tillgång till 10 miljoner dokument som "tar fram biblioteket till datorn" menar Jim McDonald projekt ledare hos LANL. Fram till april 1996 har företaget spenderat $250 000 vid införandet kring intranätet i jämförelse med att ha sparat $500 000 enbart första året.

Boeings intranät bestod 1995 av 300 servrar och man räknade med att vid årets (1996) slut skulle större delen av de 96 000 anställda ha tillgång till intranätet säger Joe Meadows, en av grundarna av Boeings Web (Michael Moeller ,"Boeing network takes flight with pioneering intranät project", Pc week Februari 1996). Då webmastern inte kunde fastställa kvantiteten på besparingen för en investering i ett intranät, kunde dom sälja in förslaget med hjälp av dess låga kostnad och höga flexibilitet.