Institutionen för Informatik

Handelshögskolan
vid Göteborgs Universitet

Arkivering av digital information

Magisteruppsats VT-01

Inlämning 010521
Handledare vid Institutionen för Informatik: Kjell Engberg
Handledare vid ADB-kontoret i Göteborg: Lars Sandin

Henrik Claesson 780530-5534

Karin Larsson 691203-5000

Abstract

The problems of recreating digital stored information is something that affects different organisations. The importance of studying how to structure data, when it comes to archiving, is to obtain a standard for further presentations of archived material. The purpose of this essay has been to create a general model, based on existing theories about information structuring, in a combination with an object oriented point of view, which resulted in our *information structuring process.* To evaluate our model we performed two case studies that gave us a foundation to draw the conclusion that it did have the general characteristics we wanted it to have. Based on the results of these case studies our belief is that the *information structuring process* can be used by either an organisation that wants to change their routine for archiving information or an organisation that is about to launch a whole new archiving project.

Abstrakt

Problematiken med att återskapa digitalt lagrad information är något som idag gör sig påmind inom olika organisationer. Vikten av att studera hur man kan strukturera upp data för arkivering ligger i att man eftersträvar en standard för att främja framtida presentationer av tidigare arkiverat material. Syftet med uppsatsen har varit att skapa en generell modell baserad på befintliga teorier om strukturerad information, i kombination med ett objektorienterat tänkande, vilket resulterade i vår informationsstruktureringsprocess. För en utvärdering av vår framtagna process gjordes två fallstudier som gav oss underlag till att dra slutsaten om att den är av den generella karaktär som vi eftersträvade. Baserat på resultaten av fallstudierna, anser vi, att vår informationsstruktureringsprocess kan användas av en organisation som vill förändra det sätt som information idag arkiveras på, eller en organisation som står inför ett helt nytt arkiveringsprojekt.

*Vi vill tacka vår handledare på ADB-kontoret, **Lars Sandin**, för den hjälp vi fått med att komma i kontakt med rätt personer på ADB-kontoret samt hans engagemang för vårt arbete. Vi vill även tacka för att vi fick deltaga vid seminariet för digital långtidslagring på Dokumenthuset i Stockholm den 29 mars 2001.*

*Vi vill även ge ett stort tack till vår handledare på Institutionen för Informatik, **Kjell Engberg**, som genom sitt kunnande och sina uppmuntrade ord inspirerat oss under uppsatsens gång.*

20 Maj, 2001

INNEHÅLLSFÖRTECKNING

1. Introduktion

När det handlar om arkivering av digitalt lagrad information, och speciellt återskapandet av denna, finns det idag svårigheter för Region- och Stadsarkivet i Göteborg (RSG) att hantera dessa aktiviteter. RSG är en arkivdepå för myndigheter, bolag och stiftelser tillhörande Västra Götalandsregionen samt Göteborgs stad[1]. De olika rutiner som finns för arkivering inom Göteborgs stad skiljer sig åt vilket ställer RSG inför problem då det krävs en mängd olika rutiner för att ta fram historisk data. Olika operativsystem, olika programvaror och till och med olika versioner av samma program kan ställa till problem då digitalt lagrade dokument utbyts. Vilken data som skall arkiveras bestäms i överenskommelse mellan RSG och den förvaltning eller det bolag som äger informationen. ADB-kontoret i Göteborg är en tjänsteleverantör som ofta utför arkiveringsarbetet.

En generell standard för arkivering inom Göteborgs Stad skulle kunna förenkla både ADB-kontorets och Regionarkivets framtida arbete. De problem som finns idag beror till största del på att Göteborgs Stad inte har några gemensamma regler över hur arkivering av digital information skall skötas.

Vikten av att studera hur man kan strukturera upp data för arkivering ligger i att man eftersträvar en standard som skall kunna främja framtida presentationer av tidigare arkiverat material. Idag finns en mängd olika dataformat vilket kräver olika program för att återskapa data. Detta försvårar och ibland omöjliggör ny framtagning av tidigare skapade dokument. Denna studie ämnar skapa en generell modell för en arkiveringsprocess samt ligga till grund för fortsatta studier inom arkivering av digital information.

När man talar om långtidslagring avser man bevarandet av information över mycket långa tidsperioder vilket kan sägas vara mer än tio år. Informationen måste bevaras oberoende av det program och den tekniska plattform som den skapats i.

Vad är det då som behöver lagras över 10 år och varför? Inom både den privata och den offentliga sektorn talar man om verksamhetens egna behov som till exempel organisationshistorik och kvalitetsdokument samt att systemen måste gallras[2] av prestandaskäl. Med prestandaskäl menar man att det blir ohållbart med för mycket information i systemen då de fungerar sämre och blir långsammare när databaserna blir för stora. Det finns även lagar som styr hur lång lagringstiden skall vara. Författningssamlingen som visas i bilaga 1 är ett exempel på regler och riktlinjer för arkivering inom Göteborgs stad. Enligt författningssamlingen så är det upp till varje myndighet, med vilket avses fullmäktige och dess revisorer, kommunstyrelse, övriga nämnder och styrelser med underställda förvaltningar samt organ med självständig ställning, att svara för vården av sitt arkiv till dess att den överlämnas till Arkivmyndigheten då det blir de som tar över ansvaret. Det är också Arkivmyndigheten som får utfärda de riktlinjer som behövs för att en god arkivvård efterföljs i staden. Såvida ej annat är föreskrivet i lag eller förordning, beslutar

[1] http://www.arkivnamnden.org/intro/gbginfo.html (2001-05-12, 17:33)
[2] Gallring - Selektering av data i arkiveringssyfte

myndighet i samråd med Arkivmyndigheten om gallring av handlingar i sitt arkiv vilket framgår under 8§ i Författningssamlingen.

Inom den offentliga sektorn måste man också lagra med tanke på rätten till insyn och framtidsforskning.

Långtidslagring är inget nytt, utan det är mediet som har förändrats genom åren. Redan för 5000 år sedan så arkiverades texter på lyckat sätt i de Sumeriska arkiven och här uppe i Norden har vi runstenar som kan ta oss tillbaka 1000 år i tiden.

Det finns dock de projekt som inte varit lika lyckade när det gäller långtidslagring av data. Amerikanska NASA lagrade data som de erhöll under Voyager-projektet (en rymdsond med destinationen Mars) på magnetband som sedan ställdes undan i arkiveringssyfte. När man ett antal år senare skulle analysera datan visade det sig att delar av den gått förlorad eftersom man inte kunde hantera och läsa det format som data var lagrat i. Vad man hade undgått att göra, vilket är en viktig del då det gäller ”underhållet” av arkiverad data, var att se till att någon ansvarade för migreringen[3] av data.[4]

Vi står dock inför problem då de gäller alla de olika sorters format som idag figurerar ute. Hur länge till kan vi läsa dessa dokument och kommer det finnas teknik som stödjer dessa tills den dagen vi vill återskapa dem?

I delavsnitten 1.1-1.8 kommer vi att belysa befintliga principer och teorier gällande hur strukturering av data och information kan göras. Tillsammans tror vi att dessa teorier kan vara en lösning på problemen med digital långtidslagring vilket leder oss till vårt undersökningsproblem som presenteras under delavsnitt 1.9.

[3] Migrera innebär att kontinuerligt konvertera information från gamla till aktuella format.
[4] Magnus Whålberg, Seminarie 010329, *Digital Långtidslagring*, Dokumenthuset

1.1. Metadata

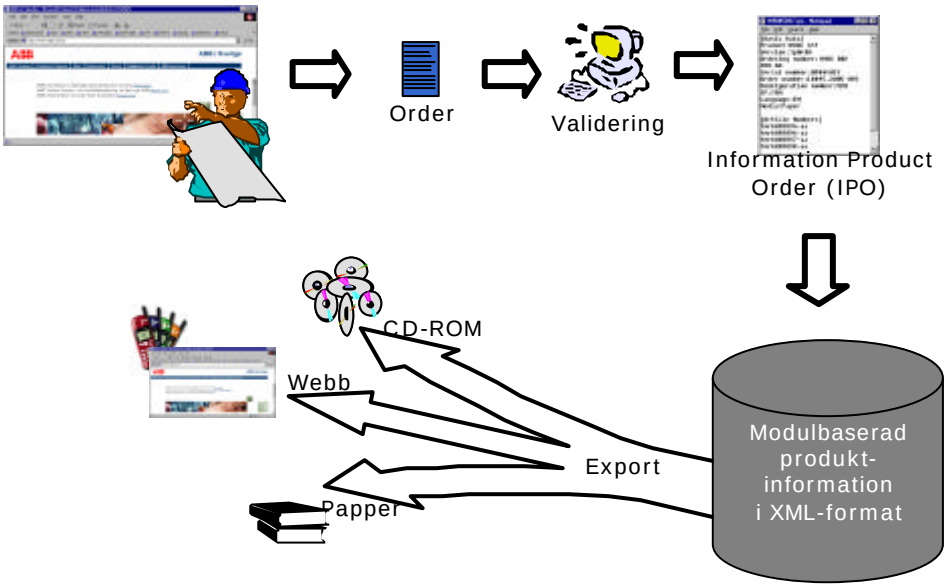
För att den ständigt växande informationsmängden skall kunna användas måste den vara strukturerad. I dagens informationssamhälle blir frågor om urval och struktur viktigare än någonsin. I konsten att organisera information ligger en bra organisationsstruktur vilken förutom återvinningsmöjligheter även bereder plats åt framtida dokument[5]. För att man skall kunna återvinna dokument måste de beskrivas. Beskrivningsmallen skall helst vara utformad med avseende på förutsägbarhet, logik och standardisering.

Metadata kan förstås som data om data, eller hellre information om information. All information som beskriver och därmed representerar ett objekt är en form av metadata. Ett exempel på metadata är bibliotekens klassiska katalogkort. Varje post består av en beskrivning av dokument, en beskrivning som ofta är högt strukturerad och innehåller olika beskrivningselement, exempelvis ”titel”, ”författare” och ”ämnesord”. Den deskriptiva presentation posten utgör, erbjuder oftast många olika sökmöjligheter. Som motpol till denna ställs ofta fulltextindexering. I teorin verkar fulltextindexering vara det optimala, eftersom hela dokumentet är sökbart, och informationssökaren alltså har möjlighet att finna alla de dokument som nämner det begrepp han eller hon är intresserad av. I praktiken blir nackdelarna snart uppenbara. Det är inte bara det att den mängd dokument sökmotorerna indexerat gör träfflistorna ohanterliga, det är också så att utan sökhjälp av ett strukturerat metadataformat är det omöjligt för en sökmotor att avgöra om ”Hemmingway” är ämne för, titeln på eller författaren till en text. Den strukturerade dokumentbeskrivningen, metadataformatet, är alltså ett sätt att förbättra sökningens träffsäkerhet och minska sökresultatets delmängd brus.

ABB, Automation Products i Västerås, är ett mycket lyckat exempel på hur strukturerad informationshantering har lett till minskad pappersåtgång och en effektiviserad dokumentprocess[6]. Innan de gjorde om sina rutiner så fick kunden inte den del av användardokumentationen som specifikt härrörde till hans vara utan fick all dokumentation, vilket kunde röra sig om sju pärmar fulla med papper. Denna bristande urvalsfunktion och onödiga pappersåtgång gjorde att de valde att se över sin dokumentationsprocess. Vad de gjorde var att de skapade en process för varje order som kom in enligt figur 1 på nästa sida.

[5] Björkhem Miriam, Lindhom Jessica, Examensarbete (20p), Bibliteks och informationsvetenskap, 2000, Lunds universitet

[6] Daniel Björkman, Seminarium 010329, *Digital Långtidslagring*, Dokumenthuset



Figur 1: ABBs dokumentprocess

Figuren beskriver flödet från beställning via nätet vilket resulterar i en order som skall valideras och efter det skickas en order med artikelnummer till dokumentdatabasen för att söka upp aktuell dokumentation som senare skickas ut till kunden i den presentationsform som han hade efterfrågat.

Metadata har kommit att spela en allt större roll då valet av katalogiseringsform kan få mycket stor betydelse för dokuments framtida åtkomlighet.

1.2. W3C

En av de stora frågorna idag när det gäller informationsmängden på World Wide Web är hur informationen skall struktureras. World Wide Web Consortium, (W3C), skapades i oktober 1994 och består av representanter för samtliga stora aktörer inom IT, bl a Netscape, Microsoft och IBM[7]. W3C:s syfte är att standardisera Webbens teknologi genom att producera specifikationer (kallade ”rekommendationer”) som beskriver webbens infrastruktur. De framtagna rekommendationerna förbinder sig W3C:s representanter senare att följa.

Några av W3C:s långsiktiga mål för webben är att göra webben tillgänglig för alla genom att lansera teknologier som tar hänsyn till användarnas olika bakgrund. Ett annat mål är att utveckla mjukvara som tillåter användarna att utnyttja webbens resurser på bästa sätt och att göra webben pålitlig med hänsyn till lagliga, kommersiella och sociala aspekter som kan tänkas uppstå.

[7] http://www.w3.org (2001-01-25, 14:23)

1.3. Digital lagring

Dagens informationshantering sker mer och mer datoriserat vilket gör att man frångår gårdagens pappersarkivering och strävar efter att arkivera i enlighet med dess skapande. Eftersom vissa delar av denna information skall bevaras för framtiden är det de ansvariga arkivens uppgift att lagra data på ett sådant sätt att informationen är läsbar under tiotals år, eller kanske ännu längre, vilket vi tidigare berört. Frågorna kring framtidens långtidslagring är många och mycket svåra att svara på då teknikutvecklingen går otroligt fort. Format som Ex 8”-floppy från 80-talet finns idag endast på tekniska museer.

CD anses hålla längre än andra digitala lagringsmedier vilket bland andra Marie-Louise Samuelsson beskriver i sin studie om lagring av information på CD-R[8]. Fördelen med CD-ROM enligt denna studie är bland annat att inspelning och avspelning sker beröringsfritt (vilket motverkar ”slitskador”) samt att CD-ROM ger en möjlighet att lagra text, ljud, bild och rörliga bilder i en sammanhållen enhet. Dessutom ger CD-ROM möjlighet till snabb återsökning i mycket stora informationsmängder.

Enligt 10:e§ i författningssamlingen (bilaga 1), rörande informationens beständighet[9], anges att handlingar som skall bevaras skall framställas med materiel och metoder som garanterar informationens beständighet. Enligt Marie-Louise Samuelssons nämnda studie är det delvis teknologin som avgör informationens beständighet. Vad som är viktigt att man inser är att informationens beständighet inte är liktydigt med mediets beständighet. Den snabba teknologiska utvecklingen gör att man är i starkt behov av att använda sig av någon form av dokumentstandard som möjliggör för framtidens teknologi att återskapa det man en gång arkiverat. Beständigheten hänger intimt samman med problemet med utgående mjuk- och hårdvara vilket orsakar problem för de som ska tillhandahålla oss med dagens information i framtiden. Om data lagras i enklast möjliga form minskar behovet av sofistikerad mjukvara.

[8] Marie-Louise Samuelsson, 1999, ”Lagra information på CD-R för framtiden”, Borås, SP Sveriges Provnings- och forskningsinstitut

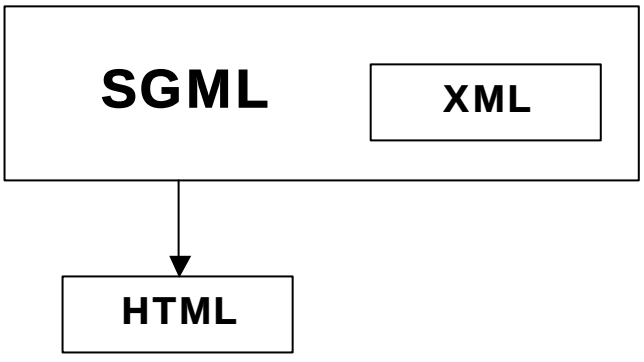
[9] *Beständighet* - möjlighet att återskapa en exakt kopia av originalet innan det har försämrats eller teknologin för att läsa det har utgått.

1.4. SGML/HTML

1986 publicerades en internationell standard för att strukturera dokument med hjälp av så kallade märkord (eng. markup) av International Standards Organization (ISO) med numret 8879 kallad Standard Generalized Markup Language (SGML)[10]. SGML specificerar en standard för att beskriva strukturen i ett dokument, med vilken man sätter upp hierarkiska modeller för dokumenten. SGML är i sig väldigt kraftfullt men samtidigt ett ganska komplicerat språk.

I början av 1990-talet föddes behovet av ett universellt format för att överföra vetenskapliga dokument i ett nätverk av i huvudsak universitet och högskolor. Det var önskvärt att det rörde sig om ett filformat som inte var beroende av operativsystem eller program och dessutom skulle det vara läs- och skrivbart för människor samt enkelt att lära. En man vid namn Tim Berners-Lee utvecklade då ett språk utifrån SGMLs syntax som kallades HyperText Markup Language (HTML) och som idag är det mest använda formatet på Internet[11]. HTML har dock den begränsningen i sin utformning att det finns en fast uppsättning märkord (även kallade ”tags”) för att beskriva dokumentet exempelvis `<BODY>`, `<TITLE>`, `<FONT>` etc. till skillnad mot SGML där man kan skapa sina egna märkord. HTML har dock gått mer och mer från att representera och strukturera själva innehållet (*descriptive markup*) till att istället säga hur innehållet skall presenteras (*formatting markup*).

Man brukar ibland säga att HTML är en applikation av SGML[9]. En mer dynamisk variant på ett språk som också härrör från SGML är eXtensible Markup Language (XML), som beskrivs betydligt mer utförligt i nästa avsnitt samt i bilaga **2**. Detta språk ses snarare som en delmängd av SGML då det i princip inte har några bestämda märkord vilket HTML har. Bild 2 illustrerar hur de tre språken hänger samman.



Figur 2: *Struktur över språkens relation till varandra*

[10] Maria Lundgren, VT 1994, ”Dokumenthantering med SGML”, Examensarbete vid systemvetenskapliga linjen, Handelshögskolan i Göteborg
[11] http://www.acc.umu.se/~alpha/xml/grund/xml2.html (2001-02-18, 11:13)

1.5. XML

”Microsoft och hela datorindustrin borde verkligen bygga sin framtid på xml, förklarade Bill Gates.”

Figur 3: Citat ur Computer Sweden[12]

XML är en förkortning av ”eXtensible Markup Language”, (sv Utbyggbar märkkod), och är kortfattat en metod för att beskriva information så att varje dator och de flesta människor förstår den. Författarna till boken ”Applied XML – a toolkit för programmers”[13] beskriver XML som datorvärldens Esperanto men med skillnaden att *alla* kan förstå det.

Ordet ”extensible” kommer från att man till skillnad från HTML kan använda egna elementnamn (”taggar”) vilket beskriver informationen i dokumentet bättre. I och med att det är den som skapar XML-dokumentet som definierar taggarna, förutom några reserverade namn, blir flexibiliteten i ett XML-dokument större än i ett HTML-dokument. Med XML struktureras information på ett sätt som gör att den är lättare att finna[14]. I figur 4 exemplifieras skillnaden mellan att skriva en adress med XML och HTML. XML ger datorn således nya möjligheter att sortera ut och hantera mer exakt den information som önskas.

Termen ”Markup” beskrivs i tidigare nämnda ”Applied XML – a toolkit för programmers”[11] som ett sätt att beskriva *hur* en text skall se ut. I ett ”Markup” -språk beskrivs dessutom informationen som innefattas i samma dokument. För de som är insatta i HTML kanske denna beskrivning känns lite klarare då man där exempelvis beskriver i ett dokument att ett eller flera ord skall vara skrivna med **fetstil** genom att explicit ange detta i dokumentet (vilket görs med `<b>` -taggen) eller som i exemplet i figur 4 ange att en tomrad skall skrivas efter varje adress (vilket görs med `<p>` -taggen).

HTML

```
<p>Prinsgatan 7<br>413 05<br>Göteborg
```

XML

```
<adress>
<gatunamn>Prinsgatan 7</gatunamn>
<postnummer>413 05</postnummer>
<postort>Göteborg</postort>
</adress>
```

Figur 4: Exempel på skillnaden mellan XML och HTML

[12] Computer Sweden 2000-11-13
[13] Alex Ceponkus, Faraz Hoodbhoy, 1999, *”Applied XML: A Toolkit for programmers”*, USA, John Wiley & Sons
[14] http://www.xml.se/xml/varfor.html (2001-02-18, 11:05)

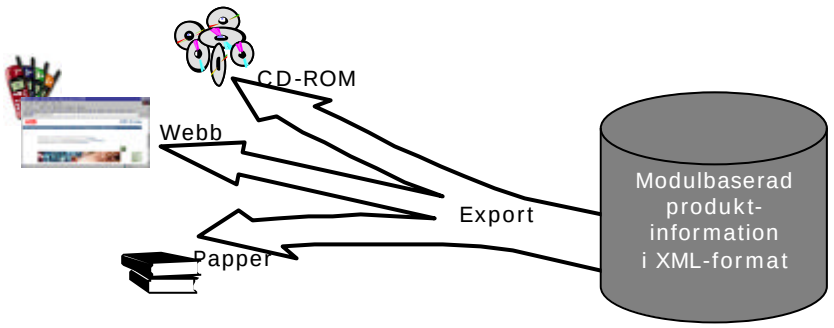
Standarden för XML är så kallad *"öpen standard"* vilket innebär att ett antal företag, individer och organisationer kommit överens om vad som skall inkluderas respektive inte inkluderas i standarden. XML är, liksom HTML, en officiell standard rekommenderad av W3C.

Lagring i XML ger oss på sikt ett nytt sätt att se på datorlagrade dokument. Det är inte längre dokumenten som är den minsta instansen av informationen utan med XML-dokumentet blir den minsta instansen varje liten datamängd inuti dokumenten, som t ex `<postort>` i figur 4.

Som tidigare nämnts så har XML utvecklats till stor del för att överbygga de begränsningar som SGML och HTML besitter. XML är tänkt som standard för strukturerad data som är anpassad för överföring via Internet[15].

Varför skulle XML då lämpa sig bra då det gäller digital långtidsarkivering?

XML-dokument är hårdvaru- och plattformsoberoende och med fullständigt stöd för Unicode, som ger varje tecken ett unikt nummer, oavsett plattform, program och språk[16], så kan olika användare utbyta dokument - vilket ger oss en ökad portabilitet. När det gäller informations beständighet kan XML-dokument underlätta då det lagras i rena text-filer samtidigt som lagringssättet av informationen är väl dokumenterat då detta är i stort sätt självbeskrivande. När det gäller återanvändning kan informationen i ett XML-dokument användas till en mängd olika ändamål. Formatet medger en hög grad av strukturering av informationen och samma källa kan användas till att producera olika saker enligt figur 5 nedan.



Figur 5: Exemplifiering av återanvändning av XML-format till olika ändamål

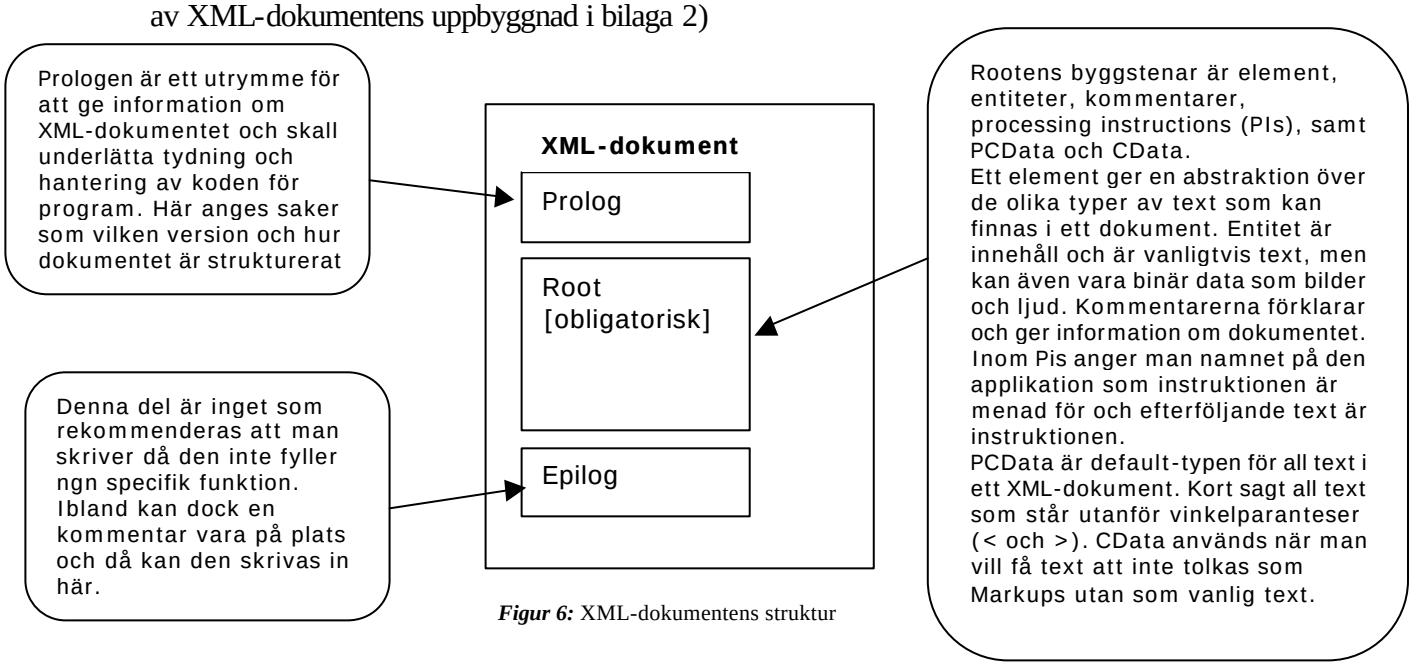
[15] http://www.acc.umu.se/ãlpha/xml/grund/xml3.html (2001-02-18, 11:13)
[16] http://www.unicode.org/unicode/standard/translations/swedish.html (2001-04-16, 15:38)

1.5.1. Uppbyggnad

Ett XML-dokument är strukturerat likt ett HTML-dokument, eller kanske snarare som ett SGML-dokument. Detta innebär att det består av ”markeringar” (eng. markup) som beskriver dokumentet och data (så kallad metadata), samt själva textinnehållet (data).

En XML-fil är egentligen inget annat än en ren textfil (eller datafil) som skrivits i antingen ASCII- eller Unicode-format, och som har ändelsen .xml.
Ett XML-dokument kan bestå av tre delar: ”prolog”, ”root” samt ”epilog” vilket illustreras med hjälp av figurerna 6 och 7 nedan. Den enda delen som *måste* vara med är root-delen även om man som grundregel bör innefatta även en prolog.

(För den mer tekniskt intresserade har vi satt samman en mer detaljerad beskrivning av XML-dokumentens uppbyggnad i bilaga 2)



Figur 6: XML-dokumentens struktur

```
Prolog:  <?xml version = ” 1.0” encoding = ”ISO- 8859-1” standalone = ”yes”?>

Root:    <dokument>
             <person>
                <namn> Henrik Claesson</namn>
                <epost>s97henk@student.informatik.gu.se</epost>
             </person>
             <person>
                <namn>Karin Larsson</namn>
                <epost>s97carin@student.informatik.gu.se</epost>
             </person>
         </dokument>

Epilog:  <!—XML exempel av: M. Eriksson -->
```

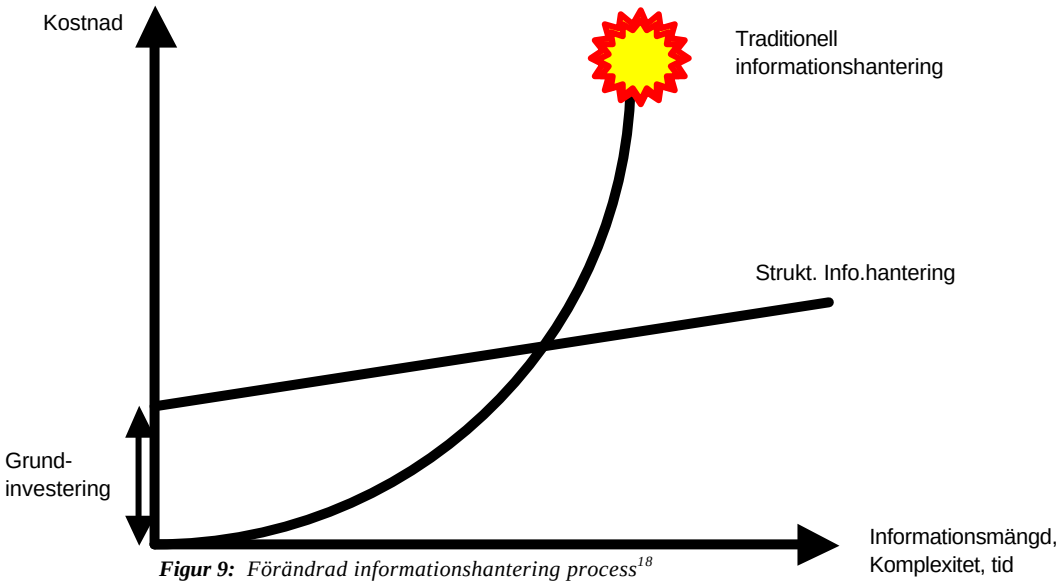
Figur 7: Exempel på kodstruktur

1.6. Framgångsfaktorer

> *"The Web, moreover, still lacks standard that would facilitate automated indexing. As a result, documents on the Web are not structured so that programs can reliably extract the routine information that a human indexer might find through a cursory inspection: author, date of publication, length of text and subjekt matter."*

Figur 8: *Citat ur Scientific American*[17]

Nu, knappt fyra år efter det att citatet ovan skrevs, har vi kommit en bra bit på väg mot ett strukturerat www.



Figur 9: *Förändrad informationshantering process*[18]

Traditionell informationshantering krävde ingen resursåtgång vilket ledde till låga kostnader i början. Problematiken som vi tidigare tagit upp med att återvinna lagrad information skapade dessvärre större kostnader. Idag läggs mer resurser på att utforma strukturer och standarder som kräver viss resursåtgång i begynnelsestadiet men som jämnar ut sig då informationstillgängligheten förenklas väsentligt.

[17] Clifford Lynch, "Searching the Internet", *Scientific American*, mars 1997
[18] Daniel Björkman, Seminarium 010329, *Digital Långtidslagring*, Dokumenthuset Stockholm

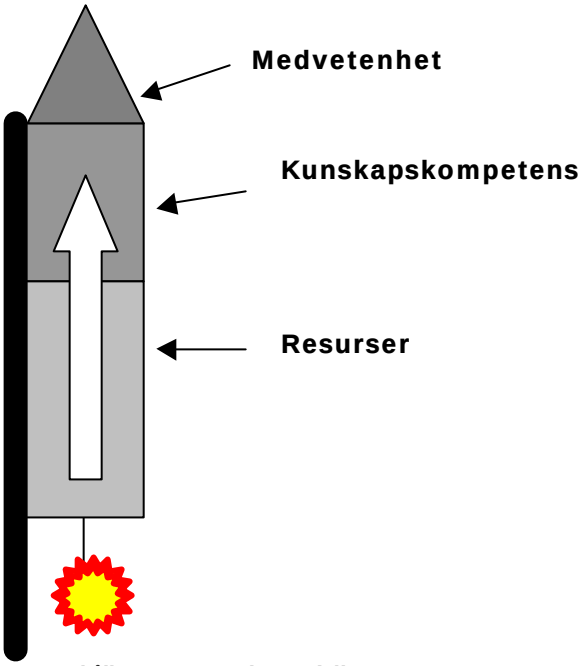
1.7. Informationsanalys

Den process för att skapa strukturerad information, i syfte att arkivera, som presenterades av Dimitris Dimitriadis, en av talarna, vid ett seminarium om Digital långtidslagring[19], innefattas av följande punkter:

?? Förstudie
?? Informationsanvändning
?? Informationsmodellering & Metadata
?? Design av XML-Schema och regler

Dessa punkter har i sin tur ett antal underrubricerade frågor för att klargöra de olika stegens (punkternas) syften där svaren tas vidare till nästa del av processen.

Magnus Whålberg[20] talade på samma seminarie om vikten av att göra en bra förstudie och presenterade en modell för att se till att man inkluderade alla de delar han ansåg nödvändiga. Han kallar sin modell, där receptet för en lyckad förstudie ingår som en del, för 3-stegs-raketen (se figur 10). Modellen i sin helhet innefattar alltså tre delar som enligt Whålberg alla är lika viktiga för att arkiveringsprojektet skall bli framgångsrikt och där den översta delen avser förstudien. Det är endast om alla delar finns med som raketen blir komplett och således även brukbar.



Figur 10: *Magnus Whålbergs 3-stegsraket-modell*

[19] Dimitris Dimitriadis, Seminarium 010329, *Digital Långtidslagring*, Dokumenthuset Stockholm
[20] Magnus Whålberg, Seminarium 010329, *Digital Långtidslagring*, Dokumenthuset Stockholm

1.7.1. Förstudie

I Dimitriadis modell, för informationsstrukturering, skall följande frågor vara besvarade då förstudien är gjord och svaren tar man sedan med sig till steg två i informationsanalysen för vidare bearbetning:

?? *Vad* är det för data som skall lagras?
?? *Varför* skall man lagra/arkivera?
?? Vilka *resurser* för lagring/arkivering har man att tillgå?

Den första av dessa frågor ämnar klargöra exakt *vilken* data som skall lagras. Det är eventuellt inte nödvändigt att *all* data i systemet arkiveras då man kanske tillämpar någon form av gallring. Detta innebär att man endast väljer att arkivera delar som det av olika anledningar finns intresse av att spara.

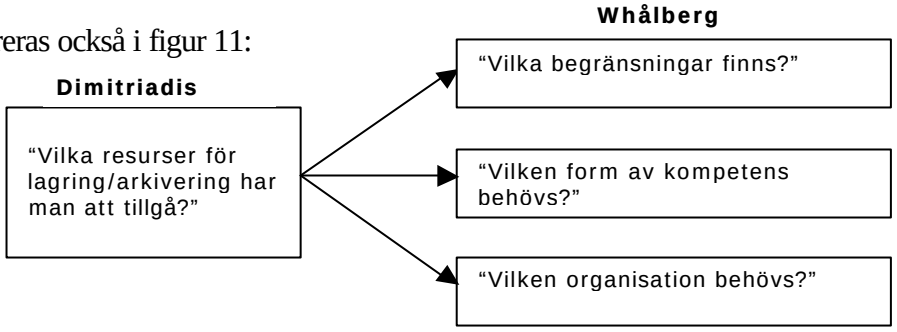
Hur vet man då vilken data som skall lagras? Frågorna ett och två hänger intimt samman på detta plan eftersom man måste veta syftet med den kommande arkiveringen innan man med säkerhet kan säga vilken data som skall arkiveras. Man ställer sig frågan: ”*Varför* skall arkiveringen göras?”

Ett argument för varför man skall arkivera data är naturligtvis av utrymmesskäl, där det dels kan handla om rent fysiskt utrymme men också att man av prestandaskäl väljer att arkivera data i systemet. Men, det gäller ändå att veta vad man kan tänkas vilja ha arkiverad data till om den någon gång måste återskapas. Kanske arkiverar man data i syfte att ha för framtida forskning eller så kanske det finns lagar och förordningar som säger att man måste kunna återskapa informationen i systemet inom ett visst antal år. Det handlar alltså om att klargöra i vilket syfte man arkiverar sin data och tillika vilken data som faktiskt skall arkiveras (och vilken som eventuellt skall gallras bort).

Den sista av dessa tre frågorna, ”Vilka *resurser* för lagring/arkivering har man att tillgå?”, kan anses vara mer utvecklad i de frågeställningar som Magnus Whålberg innefattar i förstudien för sin modell. Resursfrågan väljer nämligen Whålberg att dela upp i följande *tre* frågor:

?? *Vilka* begränsningar finns?
?? *Vilken* form av kompetens behövs?
?? *Vilken* organisation behövs?

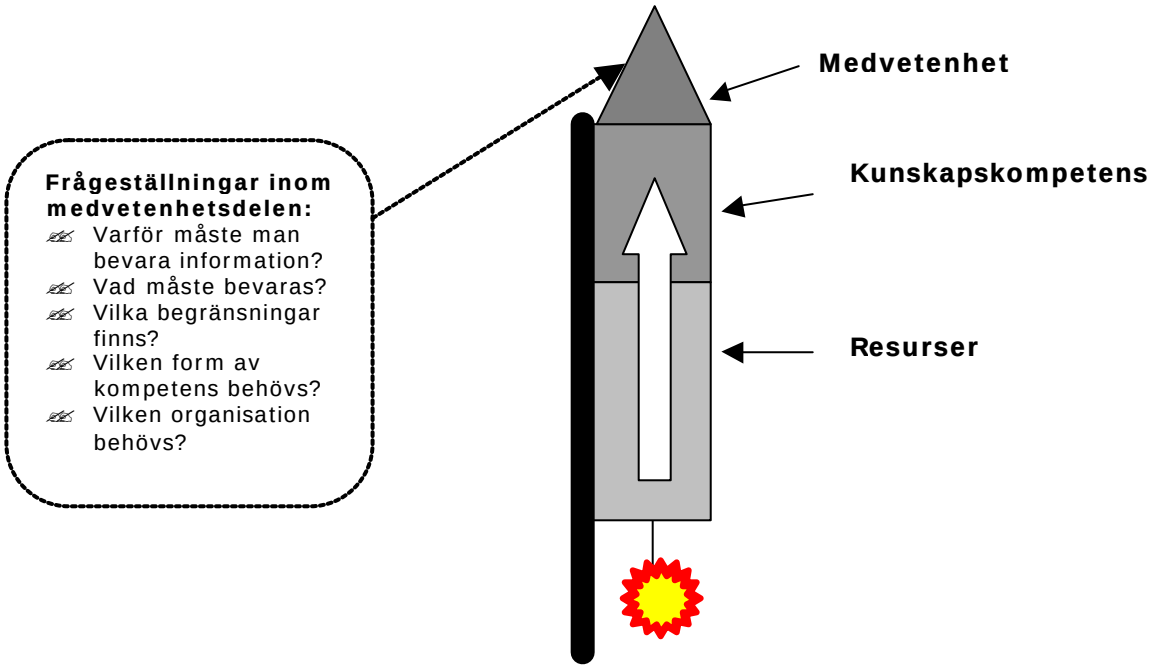
Detta illustreras också i figur 11:



Figur 11: *Resursfråga ur ett mer detaljerat perspektiv*

1.7.1.1. 3-stegsraketen

I Magnus Whålbergs 3-stegsraket-modell ingår det i medvetenhetsdelen **fem** olika frågeställningar enligt figur 12 nedan – där de **två** första kan anses vara ekvivalenta med de som Dimitriadis använder sig av under förstudien i sin modell (dvs. fokusering på *vad* för data som skall lagras samt *varför* arkivering görs). Som det beskrevs i föregående stycke kan man dessutom se de tre sista frågorna i Whålbergs förstudie som en mer detaljerad utveckling av Dimitriadis sista frågeställning, *”Vilka resurser för lagring/arkivering har man att tillgå”*.



Figur 12: *3-stegsraket-modell*

Whålberg gör alltså en uppdelning av denna fråga, gällande de resurser som finns att tillgå, till att specificera *”vilka begränsningar som finns?”*, *”vilken form av kompetens som krävs?”* samt *”vilken organisation behövs?”*.

I fallet då det handlar om att se till vilka begränsningar som kan finnas i arkiveringsprojektet är det exempelvis tekniska, formatmässiga och resursmässiga begränsningar. Man kanske är låst inom organisationen till arkivering på ett specifikt media i ett specifikt format vilket eventuellt inte kan frångås i det påbörjade arkiveringsprojektet. Begränsningarna måste noteras och beaktas under resten av arkiveringsprocessen.

Kompetensen som avses under näst sista frågan gäller formen av kompetens som behövs under projektet. Det handlar främst om vilken form av datateknisk och arkivmässig kunskap som kan tänkas behövas.

Den sista frågan vill lyfta fram behovet av en *stabil organisation* som sköter migrering av data mellan olika format dvs. att konvertera format över tiden i den mån det kommer att behövas.

Svaren på dessa frågor tillsammans med de som Dimitriadis ställer sig i sin modell utgör således ”toppen” av raketen som Whålberg kallat *medvetenhet*.

Övriga delar av 3-stegsraketen ingår inte i förstudien utan avser att realisera de delar som påvisades i medvetenhetsdelen. I vilket fall bör man se till att den *kompetens och kunskap* som krävs finns tillgänglig till projektet antingen inom organisationen eller utifrån. Det rör sig om datateknisk kunskap, arkivkunskap, viss juridisk kunskap samt kunskap om standarder för format.

Då det gäller *resurser* gäller detta både finansiella (t.ex. personal, utbildning, hårdvara, mjukvara) samt personalmässiga (datatekniker, informations- och systemarkitekter samt arkivarier och dokumentspecialister).

1.7.2. Informationsanvändning

När man gjort en förstudie och fått svar på de frågor som ställs där kommer nästa steg i Dimitriadis modell som är fokusering på *informationsanvändningen*.

Här är syftet att se vad man skall använda sin information till. Detta har man i princip påbörjat i förstudien då man försökt klargöra varför och vad man skall arkivera för data, men nu handlar det om att utöka och ytterligare definiera dessa svar. Följande tre frågor är detta steg tänkt att besvara:

?? För vilka är informationen tänkt?
?? Vilken information önskar de?
?? Hur kan informationen användas?

Detta steg flyttar alltså fokus från den data som skall arkiveras till de personer som eventuellt skall använda data som arkiverats i någon form. Handlar det exempelvis om arkivering av data i ett ekonomisystem kan man tänka sig att det är en revisor som vill kunna få tillgång till den arkiverade informationen någon gång i framtiden. Med utgång i målgruppen kan man således se till att komplettera den data man i förstudien bestämt skall arkiveras.

1.7.3. Informationsmodellering och metadata

De resultat och den information man tagit fram i de två tidigare stegen i informationsanalysen skall i nästa steg behandlas genom *Informationsmodellering*. Under denna del skapas en hierarkisk datastruktur samt att en beskrivning av data görs.

I beskrivningen av data, dvs. skapandet av metadata, är det enligt Dimitriadis viktigt med begrepp som *sökbarhet* och *administration*. Med tanke på hur data skall användas, och vilka som skall använda den, bör man tänka på hur man vill kunna göra utsökningar dvs. hur man skall skilja viss data från annan.

Informationsmodelleringen i sig är en process där man skapar en hierarkisk struktur för den data som skall arkiveras och Dimitriadis använder här termer som *moduler* och *dokument*. Det är sedan utifrån dessa strukturer, eller dessa moduler och dokument, som man enligt Dimitriadis designar XML-scheman och regler. Hur denna transformering sker i form av modulskapandet och XML-scheman återkommer i nästa stycke.

Man skiljer enligt Dimitriadis på två olika beskrivningssätt under Informationsmodellering - *deskriptiv* samt *normativ* modellering. Den deskriptiva Informationsmodelleringen utgår från den egna verksamheten och beskriver informationen på ett sätt som är lämpligt inom den egna organisationen. Den normativa modelleringen beskriver verksamheten generiskt och görs av ett branschorgan. Den beskriver informationen på ett sätt som är lämplig för alla aktörer och möjliggör utbyte och hantering av information mellan olika system.

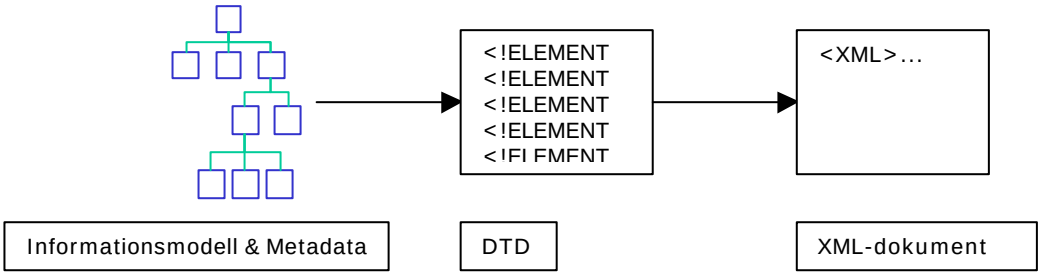
Dimitriadis belyser att informationsmodelleringen i sig ger upphov till gemensamma gränssnitt mellan olika system för datautbyte och kommunikation samt skapar verksamhets- och företagsspecifika vokabulärer i analogi med naturliga språk.

När det gäller informationsutbyte så beskriver den deskriptiva metoden den interna strukturen i olika system inom verksamheten och den normativa metoden datautbyte. En kombination av dessa kan resultera i en XML-struktur vilket kan användas internt för databaser i system samt att det används för datautbyte mellan system. XML-syntaxen är det som möjliggör kommunikation vilket kan jämföras med ett alfabet.

1.7.4. Design av XML-Schema och regler

Med utgång i de hierarkiska datamodulerna, samt tillhörande metadata, kan man nu designa XML-scheman och regler enligt Dimitriadis.

Magnus Whålberg beskriver denna process lite mer ingående då han mer konkret visar hur en framtagen informationsmodell översätts till en Document Type Definition (DTD, se bilaga 2 för vidare information) och sedan kodas i ett XML-dokument (se figur 13).



Figur 13: Design av DTD och XML-dokument utifrån Informationsmodell och Metadata

1.8. Objektsystem

Ett sätt att få en överskådlig syn på strukturen över datamodulerna är att skapa ett objektsystem där såväl de interna som de externa hierarkiska förhållandena kan visualiseras. Steget att gå från att tala om datamoduler till objekt är heller inte speciellt långt.

Tanken med ett objektsystem är att uttrycka en förenklad och gemensam bild av ett datasystems problemområde (med problemområde avses den del av omgivningen som administreras, övervakas eller styrs med hjälp av ett datasystem). När detta görs inom objektorienterad systemutveckling är syftet att bestämma de klasser[21] (som senare ligger till grund för de objekt som figurerar i systemet), med tillhörande attribut[22], som ingår i systemet som byggs[23].

Ett objekt är enligt definition i detta sammanhang *en helhet med identitet, tillstånd och beteende*. Identiteten avser att kunna identifiera ett specifikt objekt och tillståndet beskriver värdena på objektets attribut. Beteendet avser att beskriva objektets funktionalitet (detta är dock inte relevant i arkiveringssammanhang vilket gör att vi inte går vidare in på det). Ett exempel på ett objekt skulle kunna vara en bil. Denna bil har ett registreringsnummer som gör den unik så att vi kan särskilja den från alla andra bilar. Dessutom har bilen ett antal andra egenskaper eller attribut som vidare beskriver den (se exempel figur 14 nedan):

Objekt: Bil

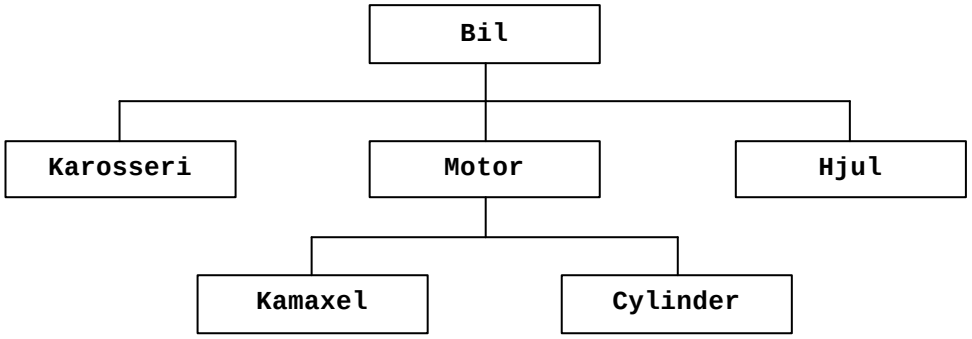
[Identitet]	
Reg.nr:	abc123
[attribut]	
Märke:	Volvo
Modell:	S40 T4
Färg:	svart
Längd(mm):	4480
Bredd(mm):	1720
Höjd(mm):	1390
Vikt(kg):	1385
Cyl.vol (cc):	1855
Hästkrafter:	200
Toppfart(km/h):	235
Nm/varv:	300/2400

Figur 14: Exempel på attributlista för ett objekt

[21] Klass - *en beskrivning av en mängd objekt med samma struktur, beteendemönster och attribut*
[22] Attribut – *namnet på en beskrivande egenskap hos en klass eller händelse*
[23] Lars Mathiassen et al., 1998, *Objektorienterad systemutveckling*, LUND, Studentlitteratur

En viktig del i skapandet av ett objektsystem är, traditionellt sett, att beskriva de *strukturer* som råder mellan objekten. Då vi talar om att skapa objektsystem ur ett arkiveringssyfte är det i princip endast en typ av dessa strukturer som vi behöver koncentrera oss på och det är *aggregat*.

En *Aggregatstruktur* beskriver en relation mellan ett objekt och de andra objekt som utgör dess beståndsdelar. Vi kan exemplifiera detta ännu en gång med vårt bilobjekt vilket illustreras i figur 15. Denna gången bortser vi dock från attributen och ser istället på hur uppbyggnaden av bilen ser ut.



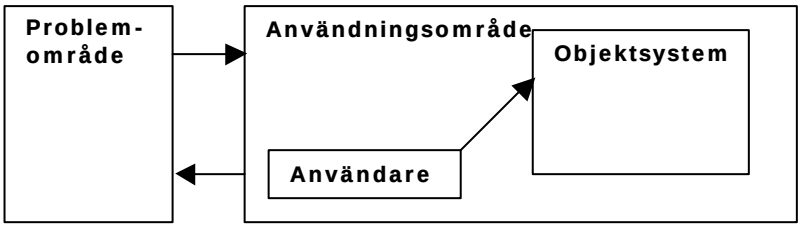
Figur 15: Aggregatstruktur för en bil

Lars Mathiassen beskriver tre olika typer av aggregatstrukturer, kallade *helhet – del*, *behållare – innehåll* och *förening – medlem*, men det finns egentligen inte någon direkt anledning att skilja dem åt i fallet då det endast handlar om att skapa en struktur i ett arkiveringssyfte.

Man talar också ibland i objektorienterade sammanhang om publika och privata attribut och funktioner för ett objekt för att åstadkomma inkapsling[24]. Detta är dock termer som snarare gör sig påminda då det handlar om implementation i ett objektorienterat programspråk.[25] Begreppen avser snarare att klargöra de interna förhållanden som råder i ett objekt och är kanske inte något man direkt talar om under objektorienterad analys och design. Detta är enda gången man ser till uppbyggnaden av ett objekts struktur.

[24] Att gömma ett objekts attribut i objektet så att de blir oåtkomliga utifrån och bara kan förändras av objektet självt.
[25] Jan Skansholm, C++ direkt, Studentlitteratur Lund 1996

Mathiassen hanterar under sin objektorienterade analys begrepp som problemområde (som beskrivs ovan) men också användningsområde. Användningsområdet är den organisation som administrerar, övervakar eller styr ett problemområde. Således hänger de tre begreppen problemområde, användningsområde och objektsystem samman på det sätt som figur 16 visar:



Figur 16: Kopplingen mellan problemområde, användningsområde, användare och objektsystem.

I *användningsområdet* finns en *användare* vars bild av *problemområdet* utgörs av *objektsystemet*.

1.9. Undersökningsproblem

Vårt arbete kommer att vara inriktat mot strukturering av data för långtidslagring på CD-ROM skivor.

Vårt undersökningsproblem kommer att utgöras av följande frågor:

”Hur skulle en informationsstruktureringsprocess kunna se ut som bygger på beskrivna teorier och tankar om strukturering av data och ett objektorienterat synsätt?”

Denna fråga ger oss underlaget till nästa fråga som lyder:

”Kan vår framtagna informationsstruktureringsprocess tillämpas som en generell modell för en arkiveringsrutin, vid strukturering av data med hjälp av XML?”

Skälet till att XML kommer att användas är att önskemål om en fördjupning inom detta område kom från ADB-kontoret samt att XML- formatet spås av många, bland andra organisationen World Wide Web Consortium (W3C), som det nya framtidsformatet för strukturering av data.

1.10. Avgränsning

Vårt arbete är avgränsat till att utnyttja CD-ROM skivor som lagringsmedie och XML som struktureringsformat. Då vårt examensarbete utförs hos ADB-kontoret är det just deras system som kommer att ligga till grund för våra fallstudier.

Vi har avgränsat vårt arbete till att studera hur man kan strukturera upp data för en framtida arkivering och valt att inte fördjupa oss i redan arkiverat material.

Vår modell bygger på att en abstrahering av data görs, vilket innebär att man under modelleringen bortser från hur data är strukturerad och lagrad och istället fokuserar på vad och hur datan skall arkiveras. Detta innebär att vi inte studerar hur man kan göra en ”översättning” av dagens datastrukturer eftersom den skiljer sig mycket mellan de olika systemen på ADB-kontoret utan fokuserar på att skapa en generell modell för att arkivera data ur ett presentationsperspektiv.

2. Metod

När det gäller valet om en kvalitativ eller kvantitativ inriktning, vilka båda syftar på hur man väljer att bearbeta och analysera den information som man samlat in, faller vår uppsats mer inom den kvalitativa analysmetoden. Den kvantitativt inriktade forskningen använder sig mer av statistiska bearbetnings- och analysmetoder i motsats till den kvalitativas verbala analysmetoder, vilken vi ansåg vara bäst lämpad inom vårt område.

Etnografi är i huvudsak en kvalitativ insamlings- och analysmetod med sitt ”inifrånperspektiv”[26]. Etnografins fokusering ligger i att försöka förstå människorna som de förstår sig själva genom att man sätter sig in i deras värld. Detta arbete sker genom olika insamlingsmetoder vilka i huvudsak oftast är: deltagande observationer, intervjuer, dokumentanalys samt alla andra metoder som strävar efter att förstå vad människor faktiskt gör i bestämda situationer. En jämförelse med etnografins ”inifrånperspektiv” kan man återfinna i litteratur om objektorienterad systemutveckling[27,28] när det gäller processen att identifiera objekt och attribut som skall användas i systemet. I objektorienterad analys och design poängteras nyttan av ett nära samarbete med användarna för att ett bra och användbart system, för just dessa användare, skall kunna tas fram. När det gäller vår uppsats och dess inriktning på strukturering av information behövs inte lika stort fokus läggas på de dagliga användarna av systemen utan här ligger mer tyngdpunkt på de som vet hur data är strukturerad och hur man kan komma åt den.

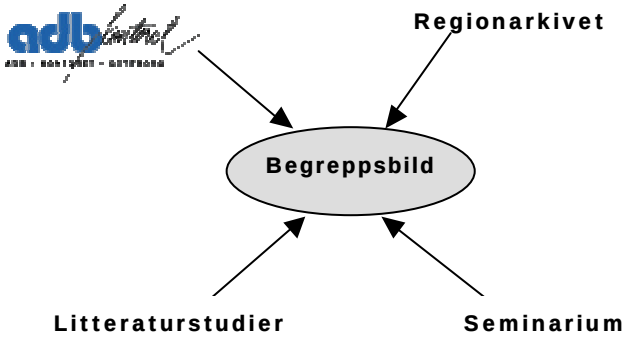
2.1. Fallstudier

Våra fallstudier utförs, som tidigare nämnts, vid ADB-kontoret i Göteborg på deras system. Fallstudier anses vara särskilt tillämpliga i utvärderingar, där studieobjekten är mycket komplexa. Man försöker exempelvis förklara, förstå eller beskriva stora företeelser, organisationer eller system, som inte enkelt låter sig undersökas med en annan metodik[29]. Vår fallstudie har olika avsikter i olika sammanhang. Den är beskrivande då det gäller vår granskning av hur det ser ut idag. Den är också förklarande när det gäller hur ny strukturering av dess data kan bli och den är undersökande då det gäller att koppla samman de två tidigare dvs. dagsläget och den nya struktureringstekniken.

[26] Magnus Bergkvist, Föreläsning Handelshögskolan, 001012
[27] David Brown, 1997, *Object-Oriented Analysis*, USA, R.R Donelly/Crawfordsville
[28] Lars Mathiassen, 1998, *Objektorienterad systemutveckling*, LUND, Studentlitteratur
[29] Jarl Backman, 1998, *Rapporter och uppsatser*, Lund, Studentlitteratur

2.2. Empiriskt material

Skälet till att vi valt det kvalitativa perspektivet ligger i att insamlingsmetoderna i huvudsak har varit intervjuer och dokumentanalys hos ADB-kontoret i Göteborg och Regionarkivet. Vi har även genomfört en egen granskning av systemen hos ADB-kontoret för att detta skulle ge oss en rik bild av de respektive systemen, av vilka ett urval kommer att presenteras senare under resultatdelen. Vår begreppsbild över problemområdet illustreras i figur 17 nedan.



Figur 17: Referenser för begreppsbild

2.2.1. Litteraturstudier

Vårt teoretiska ramverk inför uppsatsen har bestått av det som vi återspeglar i introduktionen under rubrikerna 1.1 – 1.8 samt ytterligare litteratur om systemutveckling och djupare studier inom XML-området. Litteraturstudierna har gett oss en översikt över vårt problemområde och de tidigare studier som gjorts inom ämnet digital lagring och arkivering.

2.2.2. Intervjuer

Som det beskrivs i boken, Management research (Easterby-Smith)[30], när det gäller kvalitativa metoder så är det mest fundamentala att genomföra en intervju. Intervjuerna hos ADB-kontoret har inte baserats på färdiga frågor utan har grundats på samtal med respektive ansvarig för de olika delsystemen. Systemen har beskrivits för oss utifrån dess uppgift, arkitektur samt nuvarande arkiveringsrutiner. Intervjun hos Regionarkivet baserades inte heller på färdiga frågor utan där erhöll vi dokument för att studera olika rutiner och regler när det gäller arkivering. Vi fick även en muntlig presentation över behovet efter nya rutiner i arkiveringssyfte då det idag finns en mängd oläsbart arkiverat material vilket demonstrerades för oss i samband med vårt besök.

2.2.3. Seminarium

Fokuseringen under seminariet, Digital långtidslagring hos Dokumenthuset 010329, lades på ”Dagens dokument i framtiden- kommer vi att kunna läsa dem?”.
Utifrån seminariet fick vi många infallsvinklar då det gäller själva processen vid framtagning av mer strukturerad information.

[30] Easterby-Smith et al. (1991) *Management research – An Introduction*, Sage publications

2.3. Framtagning av egen modell för informationsstrukturering

Vår modell, vilken presenteras under rubrik 3.1, är framtagen genom en sammanslagning av delar ur informationsanalysen i avsnitt 1.8 dvs Magnus Whålbergs modell tillsammans med Dimitris Dimitriadis. Vi har även infört ett objektorienterat tänkande baserat på Lars Mathiassens objektorienterade systemutveckling. Modellen är således vår tolkning och sammansättning av de modeller (och teorier) som presenterats tidigare under rubrikerna 1.8 och 1.9. Tillsammans med egna idéer och tankar samt XML-formatet, som presenterades under rubrik 1.6, utgör detta en komplett modell för att skapa strukturerad information avsedd för arkivering.

2.4. Fallstudiernas utförande

Utifrån de olika systemen som finns hos ADB-kontoret i Göteborg har vi valt ut två som skall ingå i vår fallstudie vilka vi kommer att beskriva i resultatet. Skälet till vårt val av dessa är att de idag ser helt olika ut i deras arkiveringsrutiner och struktur samt att de är uppbyggda ur olika miljöer. Då vi vill pröva och påvisa hur generell vår modell är har vi valt att tillämpa den på system som skiljer sig väsentligt i karaktär.

Vi beskriver först systemen lite översiktligt för att sedan strukturera upp aktuell data enligt vår informationsstruktureringsprocess (avsnitt 3.1).

3. Resultat

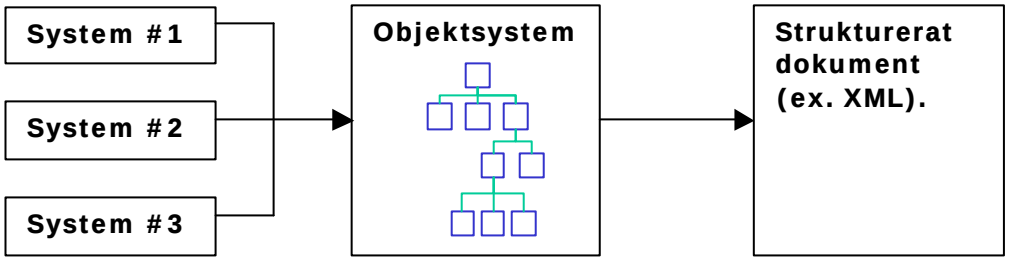
Resultatdelen byggs upp genom en presentation av vår modell vilken följs av fallstudierna som modellen provats på.

3.1. Informationsstruktureringsprocess

Syftet och förutsättningen med vår framtagna process är att skapa en generell modell för att kunna återskapa arkiverad data ur ett presentationssyfte.

Genom att arbeta med objektsystem i arkiveringsprocessen skapar man ett generellt tillvägagångssätt eftersom det i detta skede inte blir lika intressant *hur* data är strukturerad idag (i systemet där den används) utan fokus ligger istället på strukturering av data ur ett arkiveringsperspektiv. Detta görs genom en abstrahering av data och genom att skapa ett objektsystem för den data som skall arkiveras. Det finns en koppling mellan strukturen data haft innan arkiveringen gjorts och den struktur den får efter informationsmodelleringen – men kopplingen gör sig snarare påmind då det handlar om att skapa någon form av arkiveringsapplikation vilket vi bortser från i detta skede. För att förtydliga dessa tankegångar kan följande illustrerande exempel ges:

Ponera att man har tre olika datasystem som alla har olika sätt att lagra data (ett har en hierarkisk databas, ett en relationsdatabas och det sista är en enkel dokumentdatabas). Data ifrån dessa tre olika system skall arkiveras och samma metod kommer att användas för att ta fram den data som skall arkiveras oberoende vilket av systemen det gäller. För att kunna göra detta görs en form av abstrahering där man istället för att se till lagringssättet för data idag ser till hur datan *skall struktureras* vilket illustreras i figur 18.



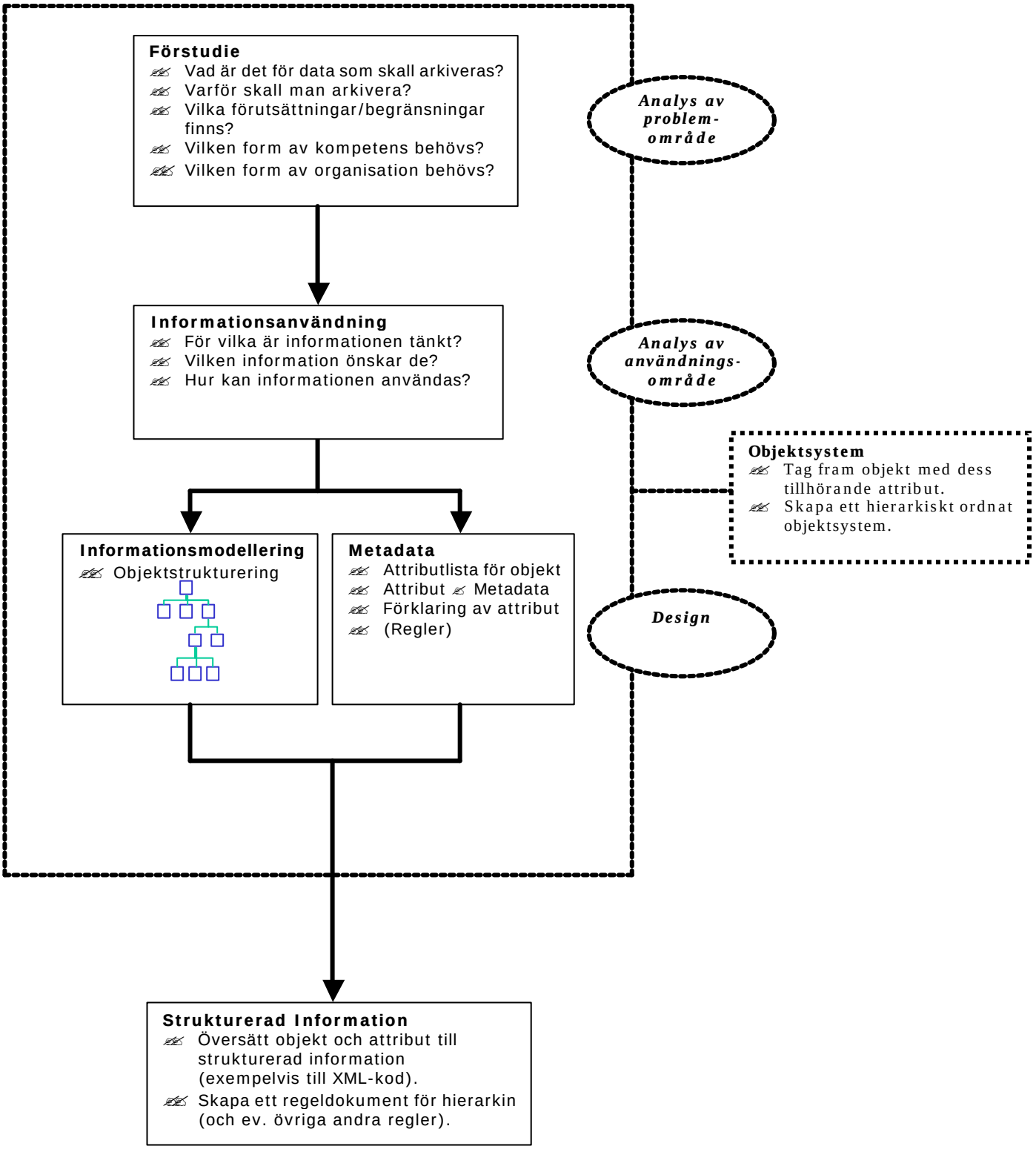
Figur 18: *Generell informationsstruktureringsprocess med hjälp av objektsystem*

Vi tror att en objektorienterad filosofi ett bra sätt att hantera data under förstudie- samt informationsanvändningssteget, samtidigt som det skapar en bra förutsättning vid informationsmodelleringen. Tänkandet och terminologin bidrar till att låta Informationsstruktureringsprocessen vara generell. Det kan även komma att bli ett naturligt inslag i en systemutvecklingsprocess om man avser skapa underlag för arkiveringen redan under själva systemutvecklingsprocessen.

Vår sammansatta modell har sin stomme i de steg som Dimitriadis presenterar men vi vill inte begränsa det sista steget i processen till att säga att realiseringen av informationsmodellen och aktuell metadata måste resultera i just ett eller flera XML-dokument utan skulle generellt vilja kalla det för ett *strukturerat dokument med information*. Vi vill också dra paralleller till den objektorienterade systemutvecklingsprocessen Lars Mathiassen beskriver för de olika stegen. Objekttänkandet är något som vi för med oss och nyttjar under stegen *förstudie*, *informationsanvändning*, *informationsstrukturering och metadata* under utvecklingsprocessen och släpper det först i princip då vi kommer till det sista steget, *strukturerad information*, då själva realiseringen av modellen man arbetat fram sker. Modellen kan punktvis beskrivas som följer i figur 19(och presenteras i sin helhet i figur 20):

- **Förstudie** - [analys av problemområdet]
- **Informationsanvändning** - [analys av användningsområdet]
- **Informationsmodell & Metadata** - [design av objektsystem]
- **Strukturerad information** - [realisering]

Figur 19: Informationsstruktureringsprocessen i punktform



Figur 20: *Informationsstruktureringsprocess modellen*

3.1.1. Förstudie

Vid en sammanslagning av Dimitriadis och Whålbergs frågeställningar vid förstudien kan man slutligen säga att den utgörs av följande fem frågor där vi dock valt att i vissa fall omformulera frågorna något (se fig. 21):

1. *Vad* är det för data som skall arkiveras?
2. *Varför* skall man arkivera?
3. *Vilka* begränsningar finns?
4. *Vilken* form av kompetens behövs?
5. *Vilken* organisation behövs?

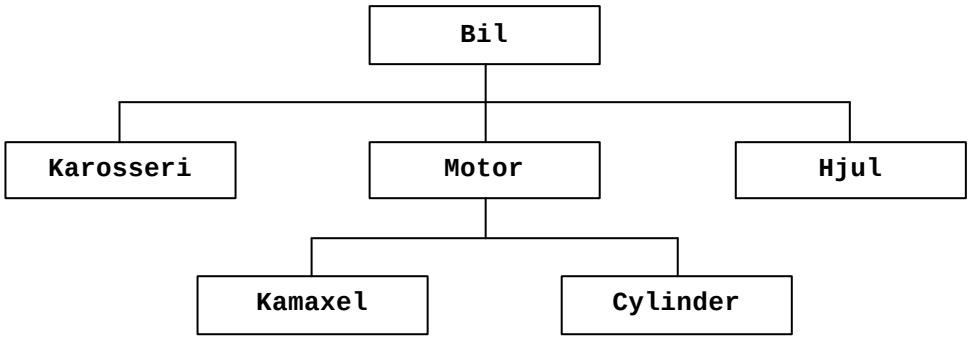
Figur 21: Förstudiens fem frågor

En utveckling av de tilltänkta svaren på frågorna redovisas nedan:

1. Klargör vilken data i systemet som skall lagras. Förekommer gallring?
2. Varför utföra arkiveringen? Vad är det för faktorer som påverkar som man måste ta hänsyn till och ha i åtanke (ex. lagar och förordningar)?
3. Vilka tekniska, formatmässiga, resursmässiga förutsättningar/begränsningar finns?
4. Redogör för kompetensbehoven: ex. datateknisk kunskap, arkivkunskap, juridisk kunskap samt kunskap om standarder.
5. Finns en stabil organisation som kan sköta eventuell migrering av data (om detta kommer att behövas)?

Redan på den här nivån inleder vi det objektorienterande tänkandet genom att se, och identifiera, den data som skall arkiveras som objekt med en identitet och tillstånd (attribut). När vi endast talar om objekt som ett sätt att abstrahera data som skall arkiveras behöver vårt fokus i princip bara vara riktat mot dessa delar av objektets egenskaper (dvs. *beteendet* ignoreras).

Resultat: Resultatet av förstudien blir en beskrivning av den data som skall arkiveras genom en beskrivning av de tilltänkta objekten med namn och struktur (enligt figur 22 nedan). Dessutom har vi beskrivit de eventuella begränsningar, eller förutsättningar, som kan tänkas finnas för arkiveringsprojektet samt vilken kompetens som behövs. Notera att vi redan på det här stadiet måste involvera människor som har kunskap om systemet samt människor som besitter arkivkunskap då en diskussion kring de identifierade objektens attribut och identitet förs.



Figur 22: Aggregatstruktur för ett objekt med underordnade objekt

3.1.2. Informationsanvändning

Frågorna som skall besvaras när man riktar blicken mot användningen av den arkiverade informationen redovisas nedan i figur 24:

1. *För vilka* är informationen tänkt?
2. *Vilken* information önskar de?
3. *Hur* kan informationen användas?

Figur 24: Informationsanvändningens tre frågor

Diskussionen kring de tre ovanstående frågorna innefattar steget för att definiera informationsanvändningen och ger underlag för att kunna gå vidare till nästa steg i processen som är *Informationsmodellering och metadata.* Frågorna är inriktade mot att få fram vilken information som skall sparas om objekten som vi tidigare identifierat under förstudien.

Parallellen till objektorienterad systemutveckling, då det handlar om analysen av användningsområdet, ligger helt enkelt i att fokusera på vad *användaren* önskar sig av systemet. Ur ett arkiveringsperspektiv försöker man tänka sig vad den användare som skall hantera arkiverad data i framtiden vill kunna göra med den. Detta bör resultera i en fullständig lista på de övergripande krav användaren har. Man bör här se till att man har tillgång till den kompetens som exempelvis kan avgöra vilka attribut som krävs för att kunna uppfylla de krav som användaren ställer (om det exempelvis handlar om ett krav på att kunna göra utsökningar på speciella kriterier).

Resultat: Resultatet av informationsanvändningsdelen blir således att man kompletterar den ursprungliga definitionen av den data som skall arkiveras (den som togs fram under förstudien) genom att eventuellt addera ett eller flera attribut till sitt/sina objekt. I figur 25 har vi exemplifierat detta med objektet Bil.

Objekt: Bil

Attribut
Registreringsnummer
Modell
Vikt
Nm/varv

Figur 25: *Fullständig attributlista för objekten som ingår i modellen.*

Vi bör även nu, efter att ha avslutat informationsanvändningssteget, ha en klar bild över vilket/vilka objekt som ingår i modellen samt eventuellt vilken struktur som råder objekten emellan – dvs. den hierarki som visas genom en aggregatstruktur.

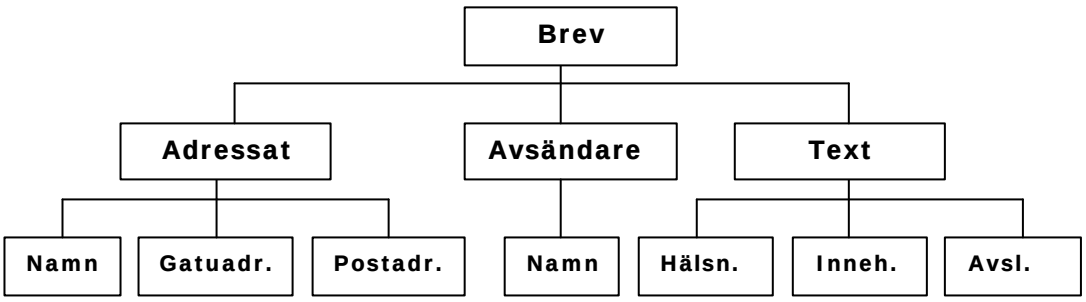
3.1.3. Informationsmodellering och metadata

Informationsmodelleringen innebär att man skapar en mer konkret och detaljerad modell av den data man i de två föregående stegen tagit fram. Denna process kan delvis liknas vid skapandet av en objektmodell i analogi med objektorienterad analys och design.

I objektorienterad analys och design ser man inte till den *inre* strukturen för ett objekt annat än att klargöra identitet och då det handlar om implementering, huruvida ett attribut eller en funktion exempelvis är privat eller publik vilket vi beskrev i stycke 1.8 Objektsystem. Vi vill dock i detta läge tala om ett objekts *interna* struktur. Man skulle också kunna uttrycka det som att från att ha haft ett *makroperspektiv* på våra objekt, då vi sett dem i relation till varandra, nu antar ett *mikroperspektiv* istället. Vi vill alltså se till den interna strukturen, eller den interna hierarkin, för vårt/våra objekt.

Vi tittar alltså nu på ett objekts delar och sätter dessa i relation, hierarkiskt sett, till varandra. Detta gör vi på samma sätt som vi illustrerade hierarkin i ett objektsystem nämligen med en aggregatstruktur. Anledningen till att vi väljer att strukturera de attribut som innefattas i ett objekt är att vi skall kunna återskapa objektet då det blivit arkiverat och då blir även den interna strukturen relevant.

Ett exempel på ovanstående resonemang skulle kunna visas med objektet Brev. Ett brev, sett ur ett makroperspektiv, är endast ett ensamt objekt till skillnad från det tidigare bilexemplet som består av flera objekt. Sett ur ett mikroperspektiv (objektets interna struktur) innefattas brevet exempelvis av en *Adressat*-del en *Avsändare*-del samt en *Text*-del. *Adressat*-delen består av ett *Namn*, en *Gatuadress* samt en *Postadress*. *Avsändar*-delen består också av ett *Namn* och *Text*-delen består av en *Hälsning*, ett *Innehåll* och ett *Avslut*. Detta är ett sätt att beskriva brev-objektets interna struktur. Se figur 25 för att se hur denna struktur illustreras.



Figur 25: Aggregatstruktur för ett brev på mikronivå

Informationsmodelleringen kommer att ge en struktur för hur data skall struktureras hierarkiskt, internt i ett objekt samt objekten emellan (om fler än ett objekt finns). Denna modell kan sedan direkt översättas på både makro- och mikronivå till strukturerad information.

I denna fas av processen bestämmer man dessutom de slutgiltiga namnen på de attribut som innefattas i objekten samt klargör objektens identitet(er). Detta är det steg då man hanterar metadata – dvs. beskriver den data som utgör objektens attribut. Man skall också försäkra sig om att man har med de attribut som behövs för att kunna realisera de krav som togs fram under informationsanvändningssteget.

Resultatet av detta steg i Informationsstruktureringsprocessen blir att man lägger till två kolumner till objektets attributlista. Den ena kolumnen innehåller det attributnamn man vill använda sig av vid arkiveringen, Metadatanamn, och den andra en kort förklaring av attributet. Som en regel kan man försöka att ha namn på sin metadata som beskriver själva attributet så bra som möjligt så att en person som läser det intuitivt förstår vad som menas.

I listan beskriver vi *i vilken strukturdel av objektet* som attributet återfinns genom att skriva in namnet på delen som innehåller attributen ovanför på en ensam rad. I de fall vi får en hierarki som är djupare än *en* del skriver vi namnet på den hierarkiskt överordnade delen inom hakparenteser före strukturdelens namn. Även strukturdelarna måste namnges i listan enligt samma princip som attributen, dvs. man skall ange metadatanamnet för dem. För att klargöra att det rör sig om en *del* skriver vi dock deras namn med versaler men beskriver dem sedan på samma sätt som vi gör med attributen. Även objekten måste namnges och deras metadata-namn skriver vi inom parenteser ovanför attributlistan för respektive objekt. Vi markerar också vilket attribut som identifierar objektet genom att skriva [ID] på attributraden. Listan bör få det utseende liknande det som exemplifieras i figur 26 nedan.

Objekt: Brev (`brev`)

Attribut	Metadata	Förklaring
BrevID	`brev_id`	brevobjektets identitet [ID]
ADRESSAT		
ADRESSAT	`adressat`	Strukturdel i brev
[Adressat] NAMN		
NAMN	`namn`	Strukturdel som innefattar underordnade attribut
Förnamn	`fornamn`	Adressatens förnamn
Efternamn	`efternamn`	Adressatens efternamn

Figur 26: Exempel på utökad attributlista med metadata och beskrivning

Vi kommer också, som ett resultat av informationsmodelleringen, få en hierarkisk beskrivning av vårt objekt som visualiseras genom en aggregatstruktur. Det är denna aggregatstruktur som vi sedan kommer att ”avbilda” genom det sätt vi beskriver den strukturerade informationen på.

I vissa fall kan det hända att man endast identifierar ett enda objekt och således inte kan skapa någon direkt aggregatstruktur eftersom man inte relaterar objektet till andra objekt. Detta behöver i sig inte innebära att man gjort fel, eller gjort en dålig modellering. Vi kan dock fortfarande tala i objekttermer och inriktar oss istället mer på objektets interna struktur för att se hur den skall byggas upp. Men, om man inte heller här inte får någon form av aggregering bör man nog se över sitt resultat. Det *kan* vara så att man har en extremt enkel struktur, men troligare är att man inte gjort den bästa av modelleringar och således bör se över sitt objekt på mikronivå ännu en gång.

Om man vill innefatta regler för objekten samt deras attribut är det även i detta stadiet man bör klargöra dessa. När vi här talar om regler så kan det exempelvis vara så att det finns regler för hur datastrukturen *måste* se ut (detta är en regel man definitivt bör ha med). Om vi ännu en gång tar vårt bilobjekt så måste exempelvis *cylinder* och *kamaxel* vara underordnade *motor* (som i sin tur är underordnad bil) när vi översätter vår objektmodell av bilen till någon form av (strukturerad) kod eftersom det är så vi valt att bygga upp vårt bilobjekt. Visar det sig att detta inte alltid passar bra (av någon anledning) så bör vi helt enkelt se över vår rådande objektmodell eller skapa en ny typ av objekt.

Den hierarkiska stukturen är alltså exempel på regler. Andra regler kan vara att ett attribut *måste* ha ett värde (och har det inget värde så får man specificera en regel som säger att det då skall tilldelas ett ”standardvärde”). Har vi den här typen av regler vi vill uttrycka kan vi lägga till en ”regelkolumn” (se figur 27) i vår lista över objekt och attribut och där fylla i de regler som eventuellt gäller för attributet.

Objekt: Brev (brev)

Attribut	Metadata	Förklaring	Regel

Figur 27: *Lista med en regelkolumn*

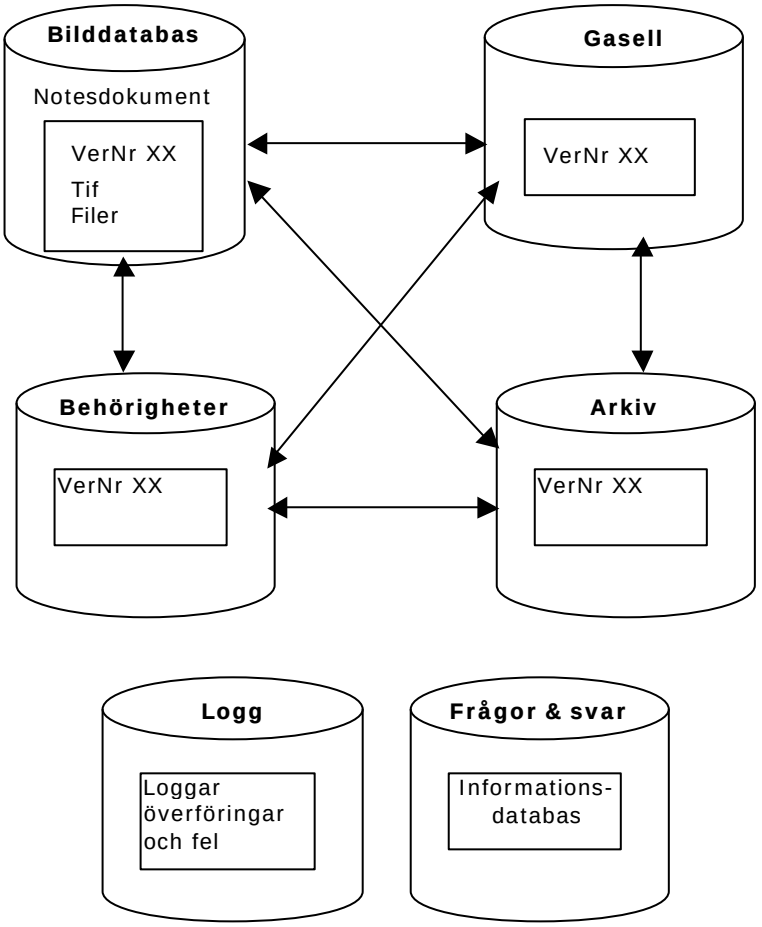
3.1.4. Strukturerad information

Det sista steget i processen är att översätta informationsmodellen med tillhörande data till någon form av strukturerat dokument – vilket företrädesvis är XML. Vid översättning till XML avbildar vi helt enkelt de objekt vi modellerat fram under processens övriga steg i XML-kod. Vi kodar de *regler* gällande attributen samt strukturen för objekten i en Document Type Definition (DTD), se bilaga 2, och datamängden som skall arkiveras i ett eller flera XML-dokument.

Då det gäller själva XML-kodningen kan det dock forfarande finnas några olika sätt att realisera den information vi har med oss från de tidigare stegen i modellen. Ett exempel på en frågeställning som kan uppstå skulle kunna vara huruvida man skall låta ett attribut i ett objekts attributlista vara *innehåll till ett element* eller *värdet på ett attribut* för elementet (se bilaga 2 för ytterligare information). Detta är inte alltid helt självklart men en liten regel skulle kunna vara att låta det attribut som identifierar objektet representeras som attribut till det element som motsvarar objektet.

3.2. Gasell

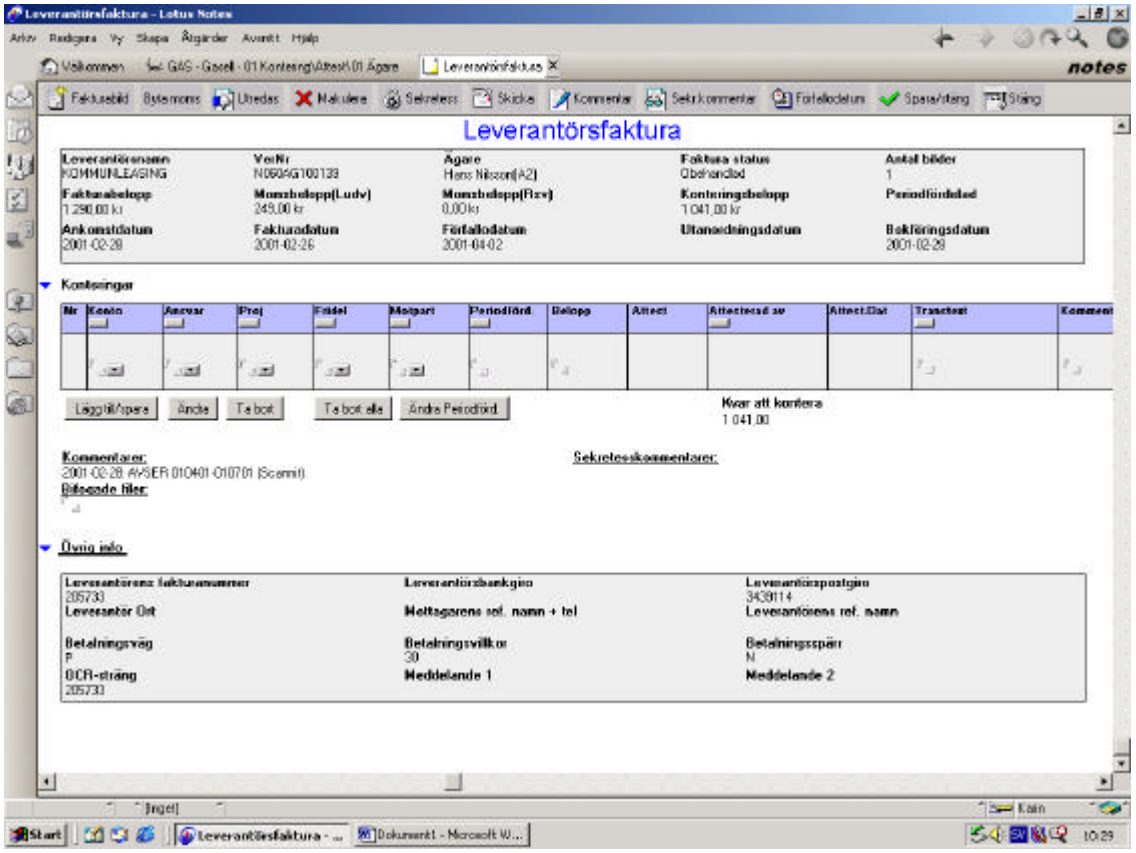
Gasell används för en rationell hantering av leverantörsfakturor hos ADB-kontoret. Den består av sex stycken samverkande Lotus Notesdatabaser och är grovt uppbyggt enligt figur 28 nedan.



Figur 28: Gasells uppbyggnad

Gaselldatabasen hanterar fakturor som är under arbete vilket innebär att de inte är klara för betalning. Då en faktura har gått till betalning flyttas den över till arkivdatabasen som är en av de databaser som gallras i arkiveringssyfte. Gallring sker även från Bilddatabasen då det är där fakturornas bifogade filer finns. De bifogade filerna består av inscannade leverantörsfakturor i TIFF-format samt andra tillhörande dokument. Behörighetsdatabasen innehåller de attest och sekretessregler som kan härröras till en faktura. Fakturornas verifikationsnummer är kopplingen mellan de olika databaserna. Loggdatabasen som loggar överföringar och fel samt Frågor och svardatabasen, som är en informationsdatabas, har ingen direkt anknytning till arkiveringen.

För vår strukturering ligger figur 29 som underlag då det är just den data som presenteras där som skall arkiveras samt de inscannade fakturor och bifogade dokument vilka ligger lagrade i bilddatabasen.



Figur 29: Ett exempel på en leverantörsfaktura i Gasell

3.2.1. Informationsstruktureringsprocessen

3.2.1.1. Förstudie

I förstudien kommer vi nu att gå igenom de fem frågeställningarna som ingår i startskedet av vår modell.
Den första frågan i förstudien lyder:

1. Vad är det för data som skall arkiveras?
Svaret på detta är att informationen som skall arkiveras är den fakturainformation som innefattas i en leverantörsfaktura i systemet. I princip är det den ”fakturabild” som användaren möter i systemet som i framtiden skall kunna återskapas. Den övriga data som också finns i systemet, men som aldrig användaren möter, bortser vi ifrån i linje med ADB-kontorets anvisningar för den här fallstudien. Enligt dessa anvisningar är det också årsvis som gallring sker.
Vid arkivering skall vi dessutom se till att fakturans bifogade filer inkluderas samt de bilder som är kopior av den fysiska originalfakturan.

2. Varför skall man arkivera?
Bokföringslagen kräver att fakturor och information om dessa sparas i minst 10 vilket är ett av skälen till att man skall arkivera. I framtiden kan det hända att man måste kontrollera fakturor och då måste dessa kunna återskapas fullständigt med tillhörande information. Dessutom råder offentlighetsprincipen inom kommunala institutioner.

3. Vilka förutsättningar/begränsningar finns?
På ADB-kontoret har man önskemål om att XML skall användas för att skapa dokument med strukturerad information. Detta bland annat eftersom Regionarkivet vill ta emot datafiler i textformat (vilket XML-filerna kommer att vara). Dessutom är det bestämt att den fysiska lagringen skall ske på CD-skivor.

De bilder som inkluderas i en faktura är i TIFF-format. Övrigt material som eventuellt bifogas är Microsoft Word och Excel-filer. Man kan dock på det här stadiet fundera över lämpligheten att bifoga filer i dessa format eftersom de kanske inte kommer att vara läsbara inom den tidsrymd man önskar. Detta framtida problem kommer att frånses i denna fallstudie och vi kommer låta de bifogade filerna förbli i ursprungs formaten.

4. Vilken form av kompetens behövs?
För att skapa strukturerad information i XML-format, med eventuellt tillhörande regler för dessa dokument, krävs kunskap i XML-programmering. Detta inkluderar även att det finns folk som kan se till att XML-filernas innehåll kan presenteras på ett bra sätt. Man måste också se till att ha kunskap om Gasell-systemets uppbyggnad och struktur för att en beskrivning skall kunna göras i nästa steg.

5. Vilken organisation behövs?
Eftersom det är Regionarkivet som har som uppgift att se till att eventuell migrering av data sköts är det inte ADB-kontorets ansvar eller skyldighet att ha en organisation där detta tas om hand. I den mån det handlar om att se till att mediet som informationen är lagrad på är hållbart kan även detta anses vara Regionarkivets uppgift och därför kommer vi inte att gå in djupare på detta i denna fallstudie.

Resultat: I det här stadiet kan vi urskilja att leverantörsfaktura är ett potentiellt objekt. Detta objekt skulle i så fall bestå av några olika delar som i sin tur har ett antal attribut under sig vilka identifieras under informationsanvändningen. Leverantörsfakturan som visas i figur 29 har nu blivit abstraherad till ett objekt med identitet. (figur 30)

Figur 30: Identifierade objekt efter förstudie

3.2.1.2. Informationsanvänding

Då vi går vidare till informationsanvändningssteget kommer vi här att komplettera vårt fakturaobjekt med attribut vilket sker genom att vi besvarar de frågor som ingår i informationsanvändningssteget:

1.För vilka är informationen tänkt?
Arkiveringen sker med syftet att i framtiden kunna återskapa en leverantörsfaktura på det sätt som den visades när den fanns i systemet. De personer som kan vara intresserade av denna information kan exempelvis vara revisorer och forskare men också personer inom den egna förvaltningen.

2. Vilken information önskar de?
Enligt de förutsättningar som givits för denna fallstudie är det endast den information som visas på en leverantörsfaktura som dessa personer önskar se (figur 29), inklusive de eventuellt bifogade filerna. Den övriga data som också tillhör en faktura (men som inte visualiseras för användaren) bortser vi ifrån i denna studie.

3.Hur kan informationen användas?
Den kan användas som underlag till forskning, revidering och som kontroll av tidigare skapade händelser. Detta innebär i stort dock bara att man skall använda informationen till att kunna återskapa en faktura i presentationssyfte och inte i bearbetningssyfte för någon form av kontroll.

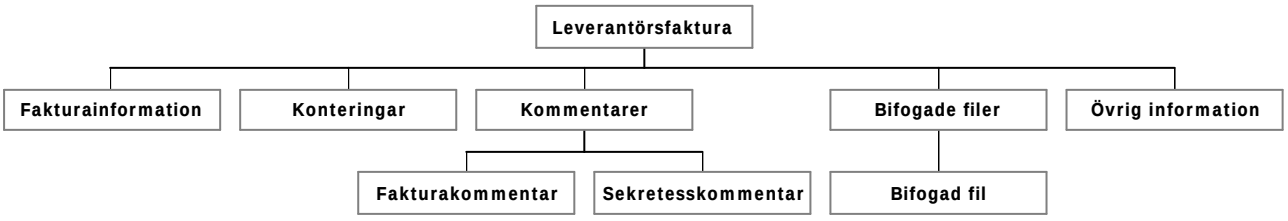
Resultat: Efter dessa två steg i Informationsstruktureringsprocessen har vi skapat en lista över de(t) objekt vi funnit samt de tillhörande attributen enligt figur 31 på nästa sida.

Attribut	
Leverantörsnamn	Belopp
VerNr	Attest
Ägare	Attesterad av
Faktura status	Attest.Dat
Antal bilder	Transtext
Fakturabelopp	Kommentar
Momsbelopp(Ludv)	Kommentar
Momsbelopp(Rsv)	Sekretesskommentarer
Konteringsbelopp	Bifogade filer
Periodfördelad	Leverantörens fakturanummer
Ankomstdatum	Leverantörsbankgiro
Fakturadatum	Leverantörspostgiro
Förfallodatum	Leverantör Ort
Utanordningsdatum	Mottagarens ref.namn
Bokföringsdatum	Mottagarens ref.tel
Nr	Leverantörens ref.namn
Konto	Betalningsväg
Ansvar	Betalningsvillkor
Proj	Betalningsspärr
Fridel	OCR-sträng
Motpart	Meddelande 1
Periodförd.	Meddelande 2

Figur 31: Attributlista för leverantörsfaktura-objektet

3.2.1.3. Informationsmodellering och Metadata

I detta steg skall vi försöka visualisera den hierarkiska strukturen som gäller för vårt ”leverantörsfaktura-objekt”. Med utgång i hur fakturan är uppdelad då den visas för användaren i systemet idag finner vi att en struktur för objektet kan vara som visas i figur 32 nedan. *Leverantörsfaktura* blir namnet på objektet i det här stadiet och underordnat finner vi delarna *Fakturainformation*, *Konteringar*, *Kommentarer*, *Bifogade filer* samt *Övrig information*. *Kommentarer* är dessutom uppdelad i *Fakturakommentar* och *Sekretessbelagda kommentarer*. *Bifogade filer* får också en underordnad del (*Bifogad fil*) där de bifogade filerna kommer att kunna inkluderas.



Figur 32: Aggregatstruktur över objektet leverantörsfaktura

Under dessa kategorier kommer i sin tur de olika attributen att återfinnas med de namn som vi väljer att beskriva dem med i form av metadata.

Då vi i listan vill beskriva inom vilken del av vårt objekt som attributet, eller aktuell metadata, ligger gör vi detta genom att skriva stukturdelens namn inom hakparenteser på raden ovanför de attribut som hör till just den delen.

Till vår lista med objektets attribut lägger vi nu till två stycken kolumner; en som vi kallar *metadata* och en som vi döper till *förklaring*. I metadata-kolumnen skriver vi in de namn som attributet får då det beskrivs som strukturerad information och i förklarings-kolumnen gör vi en kort beskrivning av vad attributet avser att beskriva. Listan för vårt leverantörsfaktura-objekt får utseendet som beskrivs i bild 33.

Objekt: Leverantörsfaktura (`lev_faktura`)

Attribut	Metadata	Förklaring
FAKTURAINFORMATION		
FAKTURAINFORMATION	`faktura_info`	Strukturdel i leverantörsfakturan
Leverantörsnamn	`leverantorsnamn`	Leverantörens namn
VerNr	`vernr`	Verifikationsnummer [ID]
Ägare	`agare`	Faktura-ägare. Den som skall handlägga fakturan
Faktura status	`faktura_status`	Status: obehandlad, konterad, attesterad, makulerad o. dyl.
Antal bilder	`antal_bilder`	Denna siffra kommer att visa hur många inscannade bilder som tillhör fakturan.
Fakturabelopp	`fakturabelopp`	Hur stort belopp som fakturan har
Momsbelopp(Ludv)	`momsbelopp_ludv`	Moms på ej affärsmässig verksamhet. Vård osv
Momsbelopp(Rsv)	`momsbelopp_rsv`	Moms på affärsmässig verksamhet. Ex VA
Konteringsbelopp	`konteringsbelopp`	Nettosumman på fakturan
Periodfördelad	`periodfordelad`	Data för att periodisera en faktura. Dela upp den på flera redovisningsmånader.
Ankomstdatum	`ankomstdatum`	När fakturan inkommit
Fakturadatum	`fakturadatum`	När fakturan är utskriven
Förfallodatum	`forfallodatum`	Sista betalningsdag
Utanordningsdatum	`utanordningsdatum`	Utanordning är den slutgiltiga godkännandet för betalning.
Bokföringsdatum	`bokforingsdatum`	När fakturan blivit bokförd

KONTERINGAR		
KONTERINGAR	konteringar	Strukturdel i leverantörsfakturan
Nr	nr	Konteringsnummer
Konto	konto	Vilket konto som berörs av fakturan
Ansvar	ansvar	En konteringskod. Som kostnadsställe.
Proj	projekt	Vilket projekt som fakturan är relaterat till.
Fridel	fridel	Konteringskod
Motpart	motpart	Vem har man handlat med
Periodförd.	periodford	Periodfördelning. Se ovan.
Belopp	belopp	Konteringsbeloppet
Attest	attest	Systembegrepp
Attesterad av	attest_av	Vem som attesterat fakturan
Attest.Dat	attest_datum	Datumet attesteringen skedde
Transtext	transtext	Transaktionstext
Kommentar	k_kommentar	Konteringskommentar
KOMMENTARER		
KOMMENTARER	kommentarer	Strukturdel i leverantörsfakturan
[Kommentarer] FAKTURAKOMMENTAR		
FAKTURAKOMMENTAR	f_kommentar	Strukturdel som innefattar underordnade attribut
Kommentar	f_komment	Fakturakommentar
[Kommentarer] SEKRETESSKOMMENTAR		
SEKRETESSKOMMENTAR	s_kommentar	Strukturdel som innefattar underordnade attribut
Sekretesskommentarer	s_komment	Sekretessbelagd kommentar
BIFOGADE FILER		
BIFOGADE FILER	bifogade_filer	Strukturdel i leverantörsfakturan
[Bifogade filer] BIFOGAD FIL		
BIFOGAD FIL	b_fil	Strukturdel som innefattar underordnade attribut.
Fil	fil	Sökväg till bifogad fil. Ex MS Word el. Excel filer samt bildfiler.

ÖVRIG INFORMATION		
ÖVRIG INFORMATION	`ovrig_info`	Strukturdel i leverantörsfakturan
Leverantörens fakturanummer	`lev_fakturanummer`	Fakturans nummer i leverantörens system.
Leverantörsbankgiro	`lev_bankgiro`	Leverantörens bankgironummer.
Leverantörspostgiro	`lev_postgiro`	Leveratörens postgironummer
Leverantör Ort	`lev_ort`	Orten leverantören befinner sig på
Mottagarens ref.namn	`mottagare_refnamn`	Namnet leverantörens kontaktperson hos oss.
Mottagarens ref.tel	`mottagare_reftel`	Telefonnumret på leverantörens kontaktperson hos oss.
Leverantörens ref.namn	`leverantor_refnamn`	Vår kontaktperson hos leverantören
Betalningsväg	`betalningsvag`	post- eller bankgiro (P el. B)
Betalningsvillkor	`betalningsvillkor`	Inom detta antal dagar skall fakturan betalas.
Betalningsspärr	`betalningssparr`	J eller blank. J spärrar fakturan
OCR-sträng	`ocr`	OCR-nummer
Meddelande 1	`meddelande_1`	Meddelande till leverantören. Pos 1-40.
Meddelande 2	`meddelande_2`	Meddelande till leverantören. Pos 41-80.

Figur 33: Utökad attributlista för leverantörsfaktura-objektet

Detta blir alltså den kompletta listan för leverantörsfaktura-objektet där dess attribut och metadata beskrivs och som sedan tillsammans med informationsmodellen, figur 32, översätts till strukturerad information i form av ett XML-dokument.

Som det beskrivs i modellen kan man också lägga till en kolumn där man klargör eventuella regler för attributen. I vårt fall finns det dock inte några direkta regler, annat än de rent hierarkiska och strukturella reglerna som beskrivs visuellt i informationsmodellen, och därför lämnar vi denna kolumn därhän.

3.2.1.4. Strukturerad information

Nu återstår endast steget att översätta informationsmodell och metadata till XML-kod och skapa ett XML-dokument samt en DTD där strukturen beskrivs regelmässigt. Ett exempel på hur XML-koden tillsammans med DTDn kan komma att se ut för ett leverantörsfaktura-objekt visas i kodavsnitten som följer i figur 34 och 35:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE root SYSTEM “gasell_struktur.dtd”>
  <root>
    <lev_faktura ver_nr="N060AG100137">
      <faktura_info>
        <leverantorsnamn>INTERN TELE AB</leverantorsnamn>
        <ver_nr>N060AG100137</ver_nr>
        <agare>Hans Nilsson(A2)</agare>
        <faktura_status>Obehandlad</faktura_status>
        <antal_bilder>1</antal_bilder>
        <faktura_belopp>544,00</faktura_belopp>
        <momsbelopp_ludv>108,75</momsbelopp_ludv>
        <momsbelopp_rsv>0,00</momsbelopp_rsv>
        <konteringsbelopp>435,25</konteringsbelopp>
        <periodfordelad> </periodfordelad>
        <ankomstdatum>2001-02-22</ankomstdatum>
        <fakturadatum>2001-02-19</fakturadatum>
        <forfallodatum>2001-03-26</forfallodatum>
        <utanordningsdatum> </utanordningsdatum>
        <bokforingsdatum>2001-02-22</bokforingsdatum>
      </faktura_info>
      <konteringar>
        <nr> </nr>
        <konto>  </konto>
        <ansvar>  </ansvar>
        <projekt>  </projekt>
        <fridel>  </fridel>
        <motpart> </motpart>
        <periodford>  </periodford>
        <belopp> </belopp>
        <attest> </attest>
        <attest_av> </attest_av>
        <attest_datum> </attest_datum>
        <transtext>  </transtext>
        <k_kommentar> </k_kommentar>
      </konteringar>
      <kommentarer>
        <f_kommentar> </f_kommentar>
        <s_kommentar> </s_kommentar>
      </kommentarer>
      <bifogade_filer>
        <b_fil>
          <fil></fil>
        </b_fil>
      </bifogade_filer>
```

Fortsättning av XML-filen

```
      <ovrig_info>
        <lev_fakturanummer>2010364</lev_fakturanummer>
        <lev_bankgiro> </lev_bankgiro>
        <lev_postgiro>2414563</lev_postgiro>
        <lev_ort> </lev_ort>
        <mottagare_refnamn> </mottagare_refnamn>
        <mottagare_reftel> </mottagare_reftel>
        <lev_refnamn> </lev_refnamn>
        <betalningsvag>P</betalningsvag>
        <betalningsvillkor>30</betalningsvillkor>
        <betalningssparr>N</betalningssparr>
        <ocr>2010364</ocr>
        <meddelande_1>  </meddelande_1>
        <meddelande_2>  </meddelande_2>
      </ovrig_info>
    </lev_faktura>
  </root>
```

Figur 34: Strukturerad information i XML-format

```
<!ELEMENT lev_faktura (faktura_info, konteringar, kommentarer,
bifogade_filer, ovrig_info)>

<!ELEMENT faktura_info (leverantorsnamn, ver_nr, agare,
faktura_status, antal_bilder, faktura_belopp, momsbelopp_ludv,
momsbelopp_rsv, konteringsbelopp, periodfordelad, ankomstdatum,
fakturadatum, forfallodatum, utanordningsdatum, bokforingsdatum)>

<!ELEMENT konteringar (nr, konto, ansvar, projekt, fridel, motpart,
periodford, belopp, attest, attest_av, attest_datum, transtext,
k_kommentar)>

<!ELEMENT kommentarer (f_kommentarer, s_kommentarer)>

<!ELEMENT bifogade_filer (b_fil)*>
<!ELEMENT b_fil (fil)*)>

<!ELEMENT ovrig_info (lev_fakturanummer, lev_bankgiro, lev_postgiro,
lev_ort, mottagare_refnamn, mottagare_reftel, betalningsvag,
betalningsvillkor, betalningssparr, ocr, meddelande_1, meddelande_2)>

<!ELEMENT f_kommentarer (f_komment)*>

<!ELEMENT s_kommentarer (s_komment)*>

<!ELEMENT fil (#PCDATA)>

<!ELEMENT ver_nr (#PCDATA)>
<!ELEMENT agare (#PCDATA)>
<!ELEMENT faktura_status (#PCDATA)>
<!ELEMENT antal_bilder (#PCDATA)>
<!ELEMENT faktura_belopp (#PCDATA)>
<!ELEMENT momsbelopp_ludv (#PCDATA)>
<!ELEMENT momsbelopp_rsv (#PCDATA)>
<!ELEMENT konteringsbelopp (#PCDATA)>
<!ELEMENT periodfordelad (#PCDATA)>
```

Fortsättning av DTD-filen

```
<!ELEMENT ankomstdatum (#PCDATA)>
<!ELEMENT fakturadatum (#PCDATA)>
<!ELEMENT utanordningsdatum (#PCDATA)>
<!ELEMENT bokforingsdatum (#PCDATA)>
<!ELEMENT forfallodatum (#PCDATA)>

<!ELEMENT nr (#PCDATA)>
<!ELEMENT konto (#PCDATA)>
<!ELEMENT ansvar (#PCDATA)>
<!ELEMENT projekt (#PCDATA)>
<!ELEMENT fridel (#PCDATA)>
<!ELEMENT motpart (#PCDATA)>
<!ELEMENT periodford (#PCDATA)>
<!ELEMENT belopp (#PCDATA)>
<!ELEMENT attest (#PCDATA)>
<!ELEMENT attest_av (#PCDATA)>
<!ELEMENT attest_datum (#PCDATA)>
<!ELEMENT transtext (#PCDATA)>
<!ELEMENT k_kommentar (#PCDATA)>

<!ELEMENT lev_fakturanummer (#PCDATA)>
<!ELEMENT lev_bankgori (#PCDATA)>
<!ELEMENT lev_postgiro (#PCDATA)>
<!ELEMENT lev_ort (#PCDATA)>
<!ELEMENT mottagare_refnamn (#PCDATA)>

<!ELEMENT mottagare_reftel (#PCDATA)>
<!ELEMENT betalningsvag (#PCDATA)>
<!ELEMENT betalningsvillkor (#PCDATA)>
<!ELEMENT batalningssparr (#PCDATA)>
<!ELEMENT ocr (#PCDATA)>
<!ELEMENT meddelande_1 (#PCDATA)>
<!ELEMENT meddelande_2 (#PCDATA)>
```

Figur 35: DTD över strukturen i XML-dokumentet

Vi har inte valt att inkludera några andra regler än de som i princip beskriver hierarkin för en leverantörsfaktura. I fallet med bifogade filer (gällande för såväl bildfiler som övriga filer) så låter vi regeln säga att det kan finnas *"noll eller flera element vars innehåll refererar till en extern fil"*. Det samma gäller för `f_kommentar` respektive `s_kommentar` som också innehåller noll eller flera underordnade element (`f_komment` respektive `s_komment`).

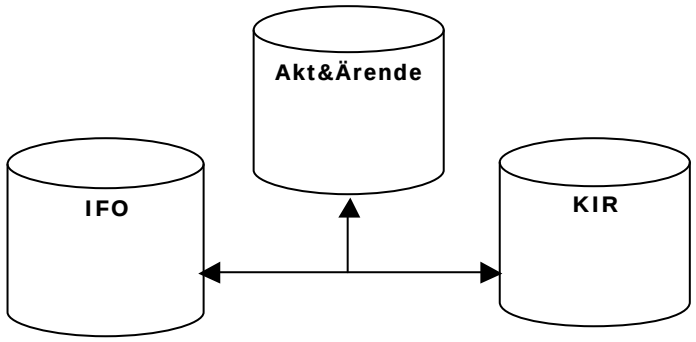
För övrigt har vi valt att låta innehållet i alla elementen vara av typen `#PCDATA` dvs. de kan innehålla ett obestämt antal tecken av något slag.

Givetvis kan reglerna för dokumentet utökas och specificeras mycket mer som att exempelvis fastställa värden för attribut etc. men vi har i detta fallet endast velat påvisa hur reglerna för strukturen kodas och således bortsett från dessa. XML-koden refererar till en DTD-fil kallad gasell_struktur.dtd vars innehåll återfinns i figur 35 ovan.

3.3. Sociala system

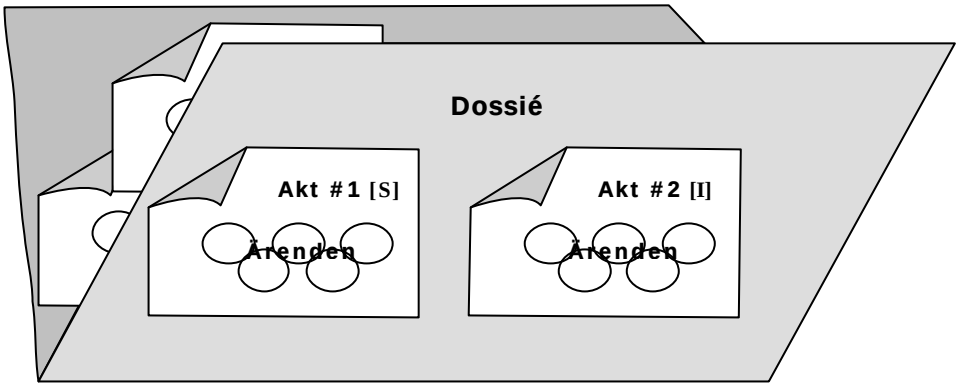
De sociala systemen, ”SOTIS”, används för att hantera socialt relaterade ärenden som exempelvis bidragssökande, familjerätt, och adoption. De sociala systemen är uppbyggda i hierarkiska databaser i stordatormiljö och är skrivna i COBOL.

Det finns två system som används och dessa är **Akt&Ärende** samt **IFO.** Dessa system är dessutom kopplade till folkbokföringssystemet **KIR** där alla personuppgifter hämtas (figur 36).



Figur 36: SOTIS struktur

Akt &Ärende systemet är uppbyggt genom att varje person som finns registrerad i systemet tillhör en dossié i vilken även hans/hennes familjemedlemmar finns registrerade. En dossié är uppdelad i olika akter i vilka man samlar de ärenden som är av samma typ. I figur 37 illustreras att en dossié innehåller olika akter som i sin tur innehåller olika ärenden.



Figur 37: Dossién uppbyggnad

När det gäller denna fallstudien har vi tvingats göra en del generaliseringar och avgränsningar i brist på information om systemet. Avsaknaden av fullständig systemdokumentation (såväl teknisk som mer överskådligt orienterad) samt det faktum att vi inte kunnat komma i kontakt med den personal som besitter den systemkunskap samt kunskap kring arkiveringsrutinerna som vi behövt har gjort att vi inte kunnat framställa en fullständigt lista över attribut.

3.3.1. Informationsstruktureringsprocessen

3.3.1.1. Förstudie

1. Vad är det för data som skall arkiveras?
Man vill arkivera data i SOTIS ”dossié-vis” med vilket menas att man vill arkivera den information som innefattas i en dossié. Detta innebär att man hämtar data från de tre olika systemen Akt&Ärende, IFO och KIR. Ur Akt&Ärende systemet hämtas data om vilka Ärenden respektive Akter som ingår i Dossién samt dess ägare och ID. IFO förser oss med data om uppgifterna som ingår i de specifika Ärendena/Akterna och KIR tillför personuppgifterna som hör ihop med dossiéerna.

2. Varför skall man arkivera?
Kommunala handlingar skall arkiveras enligt arkivlagen samt att ADB-kontoret måste tömma sina databaser på information som inte längre nyttjas. När fem år har gått och ingenting inom en dossié har blivit ändrat, uppdaterat eller tillagt skall denna arkiveras. När Regionarkivet erhållit det arkiverade materialet skall en enskild person kunna ta del av den information som finns registrerat i personens Dossié i vilken tidigare registrerade Akter och Ärenden finns. När det gäller framtida forskning är det också av väsentlig betydelse att material arkiveras för att studier om vissa ämnen skall kunna göras. Regler rörande arkivering av sociala system återfinns i socialtjänstlagen vilken bl a anger fem års kriteriet som togs upp ovan.

3. Vilka förutsättningar/begränsningar finns?
I enlighet med föregående fallstudie har man på ADB-kontoret önskemål om att XML skall användas för att skapa dokument med strukturerad information. Detta bland annat eftersom Regionarkivet vill ta emot datafiler i textformat (vilket XML-filerna kommer att vara). Dessutom är det bestämt att den fysiska lagringen skall ske på CD-skivor.

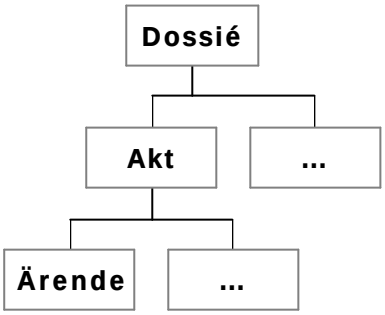
Eftersom dagens arkiveringsrutin inte fungerar speciellt bra, då Regionarkivet inte kan läsa de filer som erhålls av ADB-kontoret finns det en önskan att detta åtgärdas. En lösning på problemet vore att man gallrar ut den data som skall arkiveras ur de olika databaserna och omstrukturerar detta på ett mer överskådligt sätt i nya textfiler.

4. Vilken form av kompetens behövs?
I likhet med föregående fallstudie så krävs det kunskap i XML-programmering för att skapa den strukturerade information i XML-format, med dess eventuellt tillhörande regler.

Kompetens då det gäller gallringsrutiner av aktuell data är av betydelse. För att en bra gallring skall kunna göras krävs det även kunskap om strukturen över hur dagens systemen är uppbyggda. Den gallrade data som erhålls genom gallringen kan återskapas som nya strukturer i XML-dokument som svarar till ett arkiveringssyfte.

5. Vilken organisation behövs?
Svaret på denna frågan är detsamma som för föregående fallstudie då det är samma förutsättningar som finns för ADB-kontoret och Regionarkivet. Det är även här Regionarkivet som ansvarar för att migreringen sköts och att mediet för lagringen är hållbart.

Resultat: Eftersom man på ADB-kontoret redan talar om en dossié som ett objekt så faller det sig väldigt naturligt att låta detta utgöra ett objekt även i arkiveringen. Eftersom strukturen för en dossié också redan är (grovt) beskriven kan det även här vara lämpligt att behålla denna struktur som innebär att en dossié innehåller en eller flera underordnade objekt i form av Akter som i sin tur består av ett eller flera underordnade Ärendeobjekt (figur 38 nedan). En fördel med att behålla denna struktur är att det bland annat gör att användarna och utvecklarna redan har en gemensam bild av hur strukturen ser ut vilket kan förenkla kommunikationen och förståelsen parterna emellan.



Figur 38: Identifierade objekt och deras struktur efter förstudien

3.3.1.2. Informationsanvänding

Då vi ser till att användningsområdet, istället för problemområdet, under informationsanvändningssteget skall vi nu komplettera våra tidigare identifierade objekt med attribut. Vi har efter det första steget i Informations-struktureringsprocessen en uppfattning om vilka attribut som skall finnas med, men känner intuitivt att placeringen av dessa inte är lika självklar som objektets struktur i sig.

1. För vilka är informationen tänkt?
Arkiveringen sker med syftet att i framtiden kunna återskapa en dossié som en individ ingår i (då den personen enligt lag har rätt att ta del av detta material). Den arkiverade informationen kan också vara av väsentlig betydelse när det gäller framtida forskning.

2. Vilken information önskar de?
När det gäller den enskilde personen är det dennes personuppgifter i samband med en registrering i en viss dossié som är av intresse.
Forskare är exempelvis intresserade av vissa målgrupper inom vissa Akter och olika samband som kan erhållas ur detta.

3.Hur kan informationen användas?
Det kan vara så att den arkiverade informationen kan vara av intresse för forskare ur en statistisk synvinkel då man exempelvis vill göra utsökningar för olika målgrupper

Resultat: Efter de två inledande stegen i Informationsstruktureringsprocessen har vi nu skaffat oss en bra bild över objektstrukturen och får anse attributlistan vara komplett. Det är dock på det här stadiet fortfarande inte helt klargjort hur de tillhörande attributen skall struktureras. Idag arkiverar man data från de tre olika systemen Akt och Ärende, KIR och IFO var för sig. Vi tänker istället se till att informationen samlas på ett ställe – dvs. innefattas i dossié-objektet vilket innebär att en viss omarbetning av strukturen kommer att ske. Detta blir dock en del av nästkommande punkt (informationsmodelleringen).

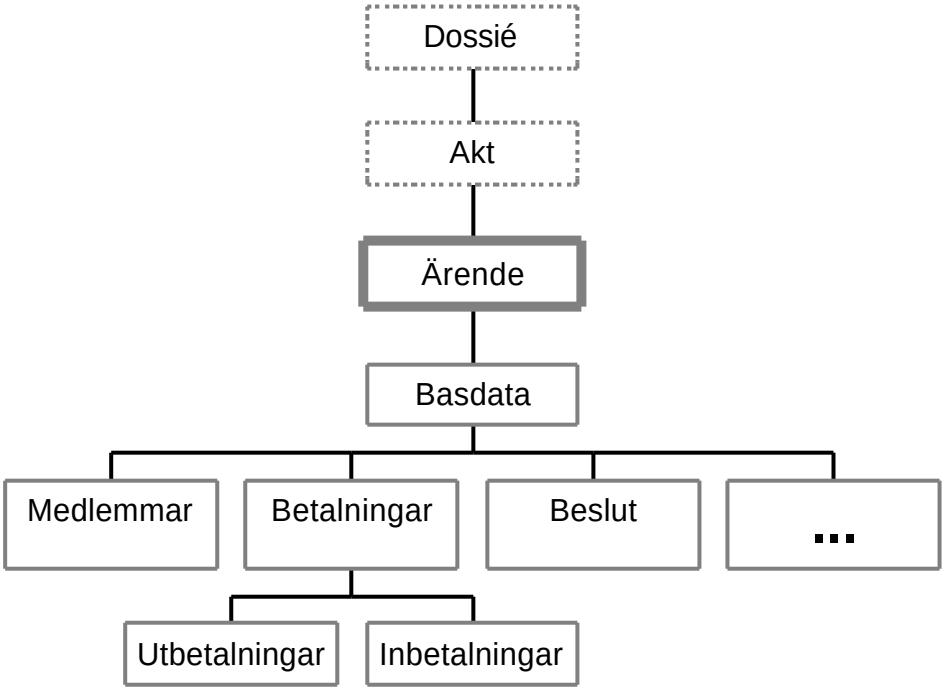
Efter dessa två steg i Informationsstruktureringsprocessen har vi skapat en lista över de objekt vi funnit samt de tillhörande attributen enligt figur 39 nedan.

Attribut		
Dossiéobjektet	**Aktobjektet**	**Ärendeobjektet**
Dossiénummer	personnr	ÄrendeID
Personnummer	Akttyp	Ärendegrp
	Aktdat	Ärendedat
		Ärendebärare
		Handläggare
		Avslutsdat
		Datum
		Personnummer
		Konto (utbet)
		Summa
		Datum
		Konto (inbet)
		Summa
		Datum
		Datum
		Text

Figur 39: Attributlista för objekten

3.3.1.3. Informationsmodellering och Metadata

Utifrån tidigare definition över hur en dossié är uppbyggd ser vår struktur över Ärende-objektet ut som i figur 40 som visas nedan. Ett Ärende är underordnad en *Akt* som i sin tur är underordnad en *Dossié*. *Ärende-objektet* har sedan en underordnad strukturdel som innehåller *Basdata* och denna del har i sin tur följande underordnade delar; *Medlemmar*, *Betalningar* och *Beslut* (samt ett antal andra delar som vi avgränsat p g a dokumentationsbrist som vi tidigare beskrivit). Betalningar innehåller även delarna *Ut*- respektive *Inbetalningar*.



Figur 40: Aggregatstruktur över objektet Ärende

Nu när objektens struktur är fastslagna skall en utökning av attributlistorna för respektive objekt göras. Som det beskrevs i modellen och vi visade i föregående fallstudie läggs nu två kolumner till tidigare skapade attributlistor. I de nya kolumnerna beskrivs strukturdelarna samt deras attribut via metadatanamnsdefinitionen och en förklarande text på attributens innebörd. De omarbetade attributlistorna för objekten visas i figur 41, 42, 43.

Objekt: Dossié (`dossie`)

Attribut	Metadata	Förklaring
Dossiénummer	`dossienummer`	Numret på dossién [ID]
Personnummer	`personnummer`	Personnumret som tillhör familjens representant

Figur 41: Utökad attributlista för dossié-objektet

Objekt: Akt (`akt`)

Attribut	Metadata	Förklaring
Persnr	`akt_personnummer`	Personnummer på person som är huvudansvarig för akten [ID]
Akttyp	`akttyp`	Vilken sorts ärende som akten gäller
Aktdat	`aktdatum`	Vilket datum akten skapades

Figur 42: Utökad attributlista för Akt-objektet

Objekt: Ärende (`arende`)

Attribut	Metadata	Förklaring
ÄrendeID	`arende_id`	Sammansatt identifierare för objektet av attributen Ärendegrp, Ärendedat, Ärendebärare (010102137805305534) [ID]
BASDATA		
BASDATA	`basdata`	Strukturdel i ärende
Ärendegrp	`arende_grp`	En kod som visar vilket typ av ärende det är.
Ärendedat	`arendedatum`	Vilket datum som ärendet skapades
Ärendebärare	`arendebarare`	Personen i hushållet som står för ärendet. (personnnummer)
Handläggare	`handlaggare`	Kod för handläggarens befattning och vart ärendet registrerades.
Avslutsdat	`avslutsdatum`	Datum då ärendet avslutats
Datum	`uppdat_datum`	senast uppdat
[Basdata] MEDLEMMAR		
MEDLEMMAR	`medlemmar`	Strukturdel som innefattar underordnade attribut
Personnummer	`m_personnummer`	Personnummer för övriga personer som ingår i ärendet.
[Basdata] BETALNINGAR		
BETALNINGAR	`betalningar`	Strukturdel som innefattar underordnade delar
[Betalningar] UTBETALNING		
UTBETALNING	`utbet`	Strukturdel som innefattar underordnade attribut
Konto	`utbet_konto`	Kontonummer
Summa	`utbet_summa`	Summan på utbetalningen

Datum	utbet_datum	Utbetalningsdatum
[Betalningar] INBETALNING		
INBETALNING	inbet	Strukturdel som innefattar underordnade attribut
Konto	inbet_konto	Kontonummer
Summa	inbet_summa	Summan på inbetalningen
Datum	inbet_datum	Inbetalningsdatum
[Basdata] BESLUT		
BESLUT	beslut	Strukturdel som innefattar underordnade attribut
Datum	beslut_datum	Datum för beslutstagandet
Text	beslut_text	Information om beslut

Figur 43: Utökad attributlista för ärende-objektet

Dessa listor är nu kompletta för att nästa steg kunna påbörjas. Som vi tidigare beskrivit används dessa listor tillsammans med informationsmodellen som underlag till strukturerad information i form av XML-dokument.

3.3.1.4. Strukturerad information

I likhet med föregående fallstudie återstår nu steget att översätta vår informationsmodell och vår beskrivna metadata till XML-kod och skapa ett XML-dokument samt en DTD där strukturen beskrivs regelmässigt. Ett exempel på hur XML-koden och DTDn för dossié-, akt- och ärendeobjektet kan se ut visas nedan i figur 44 och 45.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE root SYSTEM "sotis_struktur.dtd">
<root>
  <dossie dossienummer="12345">
    <personnummer>7805305534></personnummer>

    <akt akt_personnummer="7805305534">
      <akttyp>S</akttyp>
      <aktdatum>20010530</aktdatum>

        <arende arende_id="010102137805305534">
          <basdata>
            <arende_grp>10</arende_grp>
            <arendedatum>20010117</arendedatum>
            <arendebarare>7805305534</arendebarare>
            <handlaggare>E201</handlaggare>
            <avslutsdatum>20010521</avslutsdatum>
            <uppdat_datum>20010519</uppdat_datum>
```

Fortsättning av XML-filen

```
            <medlemmar>
              <m_personnummer>7805305534</m_personnummer>
              <m_personnummer>7604185545</m_personnummer>
            </medlemmar>

            <betalningar>
              <utbet>
                <utbet_konto>55550055555</utbet_konto>
                <utbet_summa>300</utbet_summa>
                <utbet_datum>20010417</utbet_datum>
              </utbet>
              <utbet>
                <utbet_konto>55550055555</utbet_konto>
                <utbet_summa>300</utbet_summa>
                <utbet_datum>20010517</utbet_datum>
              </utbet>
              <inbet>
                <inbet_konto>78945612378</inbet_konto>
                <inbet_summa>9000</inbet_summa>
                <inbet_datum>20010519</inbet_datum>
              </inbet>
            </betalningar>

            <beslut>
              <beslut_datum>20010216</beslut_datum>
              <beslut_text>Inget beslut fattat</beslut_text>
            </beslut>

          </basdata>
        </arende>
      </akt>
    </dossie>
  </root>
```

Figur 44: Strukturerad information i XML-format

```
<!ELEMENT dossie (dossienummer, personnummer+, akt+)>
<!ELEMENT akt (akt_personnummer, akttyp, aktdatum, arende+)>
<!ELEMENT arende (arende_id, basdata)>
<!ELEMENT basdata (arende_grp, arendedatum, arendebarare,
handlaggare, avslutsdatum, uppdat_datum, medlemmar, betalningar,
beslut)>
<!ELEMENT medlemmar (m_personnummer*)>
<!ELEMENT betalningar (utbet, inbet)*>
<!ELEMENT utbet (utbet_konto, utbet_summa, utbet_datum)>
<!ELEMENT inbet (inbet_konto, inbet_summa, inbet_datum)>
<!ELEMENT beslut (beslut_datum, beslut_text)*>

<!ATTLIST dossie dossienummer PCDATA default>
<!ELEMENT personnummer (#PCDATA)>

<!ATTLIST akt akt_personnummer PCDATA default>
<!ELEMENT akttyp (#PCDATA)>
<!ELEMENT aktdatum (#PCDATA)>

<!ATTLIST arende arende_id PCDATA default>

<!ELEMENT arende_grp (#PCDATA)>
<!ELEMENT arendedatum (#PCDATA)>
<!ELEMENT arendebarare (#PCDATA)>
<!ELEMENT handlaggare (#PCDATA)>
<!ELEMENT avslutsdatum (#PCDATA)>
<!ELEMENT uppdat_datum (#PCDATA)>

<!ELEMENT m_personnummer (#PCDATA)>

<!ELEMENT utbet_konto (#PCDATA)>
<!ELEMENT utbet_summa (#PCDATA)>
<!ELEMENT utbet_datum (#PCDATA)>

<!ELEMENT inbet_konto (#PCDATA)>
<!ELEMENT inbet_summa (#PCDATA)>
<!ELEMENT inbet_datum (#PCDATA)>

<!ELEMENT beslut_datum (#PCDATA)>
<!ELEMENT beslut_text (#PCDATA)>
```

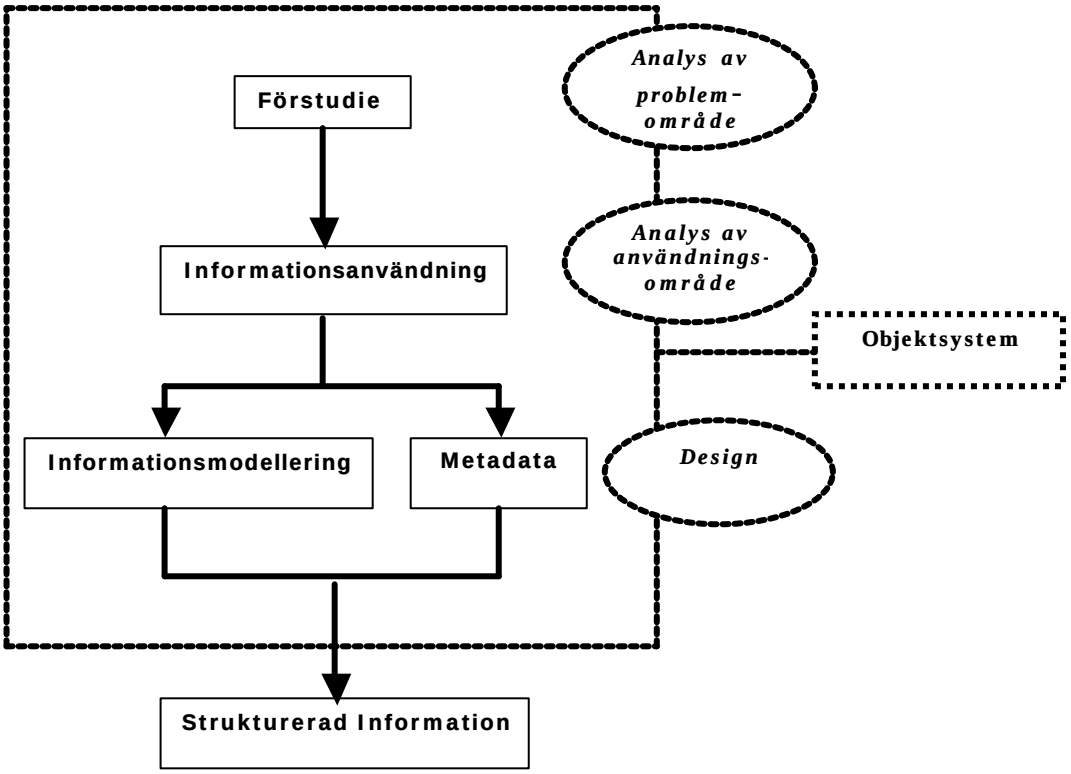
Figur 45: DTD över strukturen i XML-dokumentet

Vi har även här valt att inkludera några andra regler än de som beskriver hierarkin för de olika objekten. Exempelvis slår vi fast att en dossié måste innehålla minst en eller flera akter och detsamma gäller för en akt relaterat till ärende.

XML-koden refererar till DTD-filen kallad sotis_struktur.dtd vars innehåll återfinns i figur 45 och alla elementen innehåller värden av typen `#PCDATA` i likhet med tidigare fallstudie.

4. Diskussion

Vår diskussion baseras på hur vi funnit att vår Informationsstruktureringsprocess fungerat under de båda fallstudierna som utförts. Detta gör vi genom att föra en diskussion kring de olika punkterna i modellen enligt vår figur 46 nedan.



Figur 46: Informationsstruktureringsprocessens delar

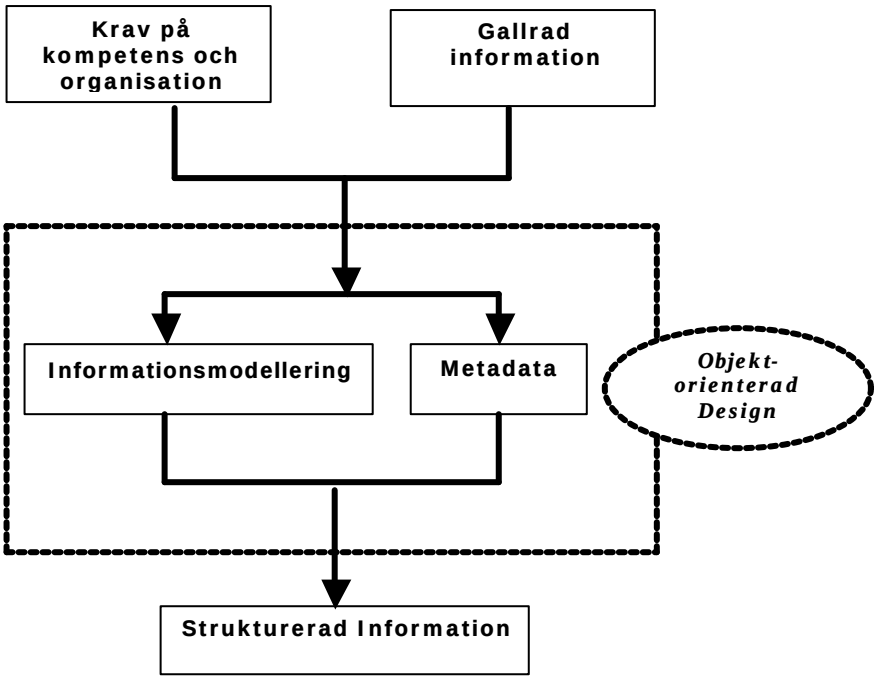
Övergripande tycker vi att modellen har fungerat väl och uppfyllt sitt mål som en generell modell som kan användas vid ett arkiveringsprojekt för datasystem med olika uppbyggnad. I och med införandet av objekttänkande och abstrahering av den data som skall arkiveras kan man frångå dagens datastrukturer i systemet, även om man stundom kan låta dem agera vägvisande, vilket vi utnyttjat i våra fallstudier.

Under arbetet med förstudierna fann vi svårigheter med att få tag i den kompetens som behövdes för att klargöra exakt *vilken* data som skulle arkiveras (detta gäller främst arbetet med de sociala systemen). Under detta delsteg av processen uppmärksammade vi hur viktigt det är att fullständig systemdokumentation finns tillgänglig då man behöver sätta sig in i ett datasystem. Det är viktigt att man som systemutvecklare kommer i kontakt med de människor som besitter den kunskap man behöver samt har tillgång till en fullständig dokumentation över systemen. Problematiken med bristande systemdokumentation är något man är medveten om på ADB-kontoret. Ett nytt arkiveringsprojekt skulle kunna vara ett bra tillfälle att se till att kompletterande systemdokumentation framställs.

I det stora hela har vi funnit att just kompetensaspekten är mycket viktig i arkiveringssammanhang. Hur väl strukturerad informationen man än arkiverar förlorar detta sin betydelse om det visar sig att man arkiverat ”fel”, eller att man inte arkiverat ”nödvändig”, data.

Vi uppmärksammade även under förstudien samt informationsanvändningssteget att vissa frågeställningar antingen varit lite svåra att besvara alternativt i princip redan varit besvarade under ett tidigare skede i processen. Ännu en gång kommer just kompetensaspekten in och här påvisas tydligt hur viktigt det är att de människor som besitter olika kunskap involveras. Alla de personer som berörs av arkiveringen, exempelvis systemanvändare, programmerare, systemansvariga samt personer som med kunskap om lagar och förordningar vid arkivering, skall tas med i arkiveringsprojektet. Om frågor i modellens första delsteg redan anses vara besvarade, eller inte kan besvaras, kan det bero på att man i tidigare steg gått händelserna i förväg eller att ett bra svar ej kan ges. Vi vill med detta påvisa att frågorna under de första delstegen inte får ses som helt statiska utan att en viss omarbetning kan göras med hänsyn till olika system.

För de system där man redan har en väl fungerande gallringsrutin för arkivering kan man i princip bortse från informationsstruktureringsprocressens två första delsteg dvs. förstudien och informationsanvändningen. Eftersom man redan vet vilken data som skall arkiveras, samt syftet med arkiveringen, kan man istället se till att rätt personer finns involverade i arkiveringsprojektet. Den kompetens som krävs är hur gallrad data är strukturerad på makronivå, dvs. hur objektens externa struktur ser ut samt objektens tillhörande attribut. Informationsstruktureringsprocessen skulle i detta fall kunna få utseendet enligt figur 47, dvs. man går direkt in i designfasen av processen.



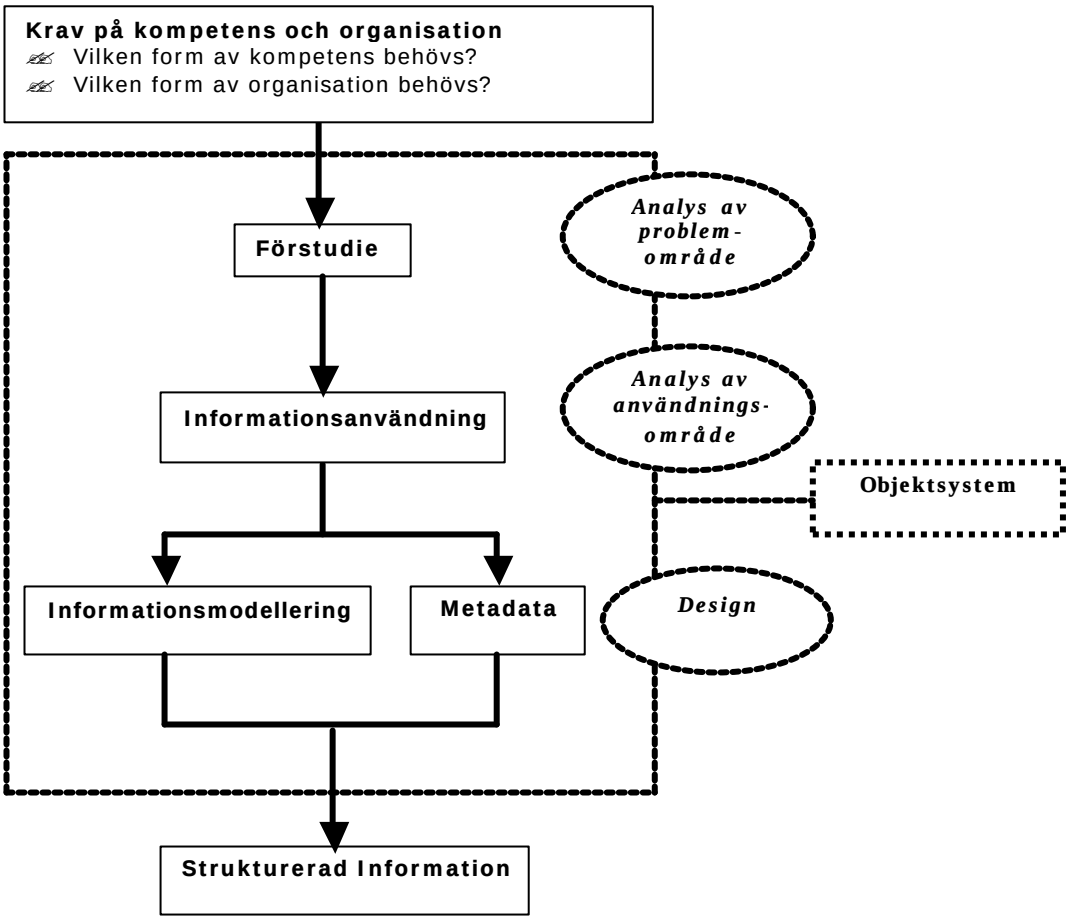
Figur 47: Informationsstruktureringsprocessen då gallringsrutin redan existerar

Då det gäller själva modelleringsbiten i informationsstruktureringsprocessen handlar detta ibland, liksom systemutveckling gör, nästan om ett hantverk och man kanske snarare skulle tala i termer om bättre eller sämre modelleringar än om rent felaktiga. Systemutveckling är något som kräver en viss erfarenhet och det är nog en fördel att involvera någon relativt erfaren systemutvecklare även i ett arkiveringsprojekt.

Vår modell är fokuserad på att arkivera information ur ett återskapningsperspektiv som riktar sig mot visualisering av den data som skall arkiveras. För att lyckas med detta, samt att skapa en generell modell, valde vi att abstrahera data och sedan strukturera den för att kunna återskapas ur ett presentationsperspektiv. Ett annat angreppssätt vore dock att försöka spara data med samma struktur som den idag är lagrad i systemet. Detta är i viss mån på det sätt som arkiveringen går till idag hos systemen i våra fallstudier och där resultatet i vissa fall är bristande då data gallras och arkiveras men ej kan återskapas. Om man dock involverar arkiveringsaspekten i systemutvecklingsprocessen skulle detta kanske kunna göras enklare. Man skulle kunna tänka sig att det kanske fanns en separat databas som endast innehöll utgallrad data för arkivering, vilket skulle förenkla arkiveringsprocessen genom att den aktuella data som skall arkiveras är samlad på ett ställe. Detta skulle kunna ge upphov till en enkel och effektiv generering av strukturerad information exempelvis i form av XML-dokument. Vi vill poängtera att det är en mycket god idé att bära med sig arkiveringsperspektivet redan under en systemutvecklingsprocess.

Vår uppsats har i huvudsak handlat om att ta fram och arbeta utefter en modell, en informationsstruktureringsprocess, i arkiveringssammanhang. En förutsättning vi hade var att den strukturerade informationen skulle realiseras, eller kodas, i XML – vilket vi också gjort. Vi fann att XML-formatet, för den strukturerade informationen, är ett bra sätt att beskriva strukturer och data (i form av metadata). Som vi skrivit tidigare är XML-dokumenten rena text- eller datafiler och på grund av det sätt som de struktureras på, i kombination med att man valt bra namn för metadata, är de mycket enkla att intuitivt tolka då man tittar på själva koden. Dessutom spås XML vara ett framtidsformat med alla de fördelar som det besitter vilket gör det till ett bra val. Vi hade gärna satt oss in mer i XML om tid hade funnits och inser att den XML-kod som presenteras i uppsatsen stundom kan te sig lite simpel för den som är mer insatt, men vår intention har snarare varit att exemplifiera (med en enklare kod) än att skriva avancerad XML-kod.

Om vi ser lite mer specifikt till vår modell, utifrån de olika steg som informationsstrutkureringsprocessen innefattar, skulle vi vilja påpeka följande; Vi har valt att inkludera frågan gällande vilken kompetens som behövs i arkiveringsprojektet i förstudieskedet. Detta anser vi i sig inte vara fel, men en variant vore att man bröt ut frågorna gällande organisationen samt kompetensen i ett steg före förstudien, enligt figur 48 nedan. En utbrytning av dessa frågor skulle innebära att man vet vilken kompetens och typ av organisation som krävs då arkiveringsprocessen startar.



Figur 48: Informationsstruktureringsprocessen med utbrutna kompetens- och organisationskrav

Efter uppdelningen av förstudien kan man nu lättare få svar på de frågor som inleder denna – dvs. *”Vilken data skall arkiveras?”* samt *”Varför skall man arkivera?”*. Detta stämmer bra överens med Magnus Whålbergs 3-stegsraket, som togs upp i 1.7.1.1, där han beskriver att *alla* delarna av raketen måste finnas med eftersom den annars inte går att ”starta”. Liksom Whålberg anser vi att *Medvetenhet*, *Kunskapskompetens* samt *Resurser* måste integreras i processen om projektet skall få ett bra utfall.

Vi är medvetna om att det kan finnas delar av processen som kanske inte är tillräckligt tydliga, eller att det finns utrymme för missuppfattningar, som vi som skapare av informationsstruktureringsprocessen inte är medvetna om.

Som svar på vår första fråga,

”Hur skulle en informationsstruktureringsprocess kunna se ut som bygger på beskrivna teorier och tankar om strukturering av data och ett objektorienterat synsätt?”

presenteras vårt förslag till en informationsstruktureringsprocess, som är sammansatt av de teorier och tankar som tidigare tagits upp, i resultatdelen under rubrik 3.1. (figur 16)

På den andra frågan,

”Kan vår framtagna informationsstruktureringsprocess tillämpas som en generell modell för en arkiveringsrutin, vid strukturering av data med hjälp av XML?”

skulle vi, baserat på de båda genomförda fallstudierna, vilja avge ett jakande svar. Vi har funnit den tillämpbar på två helt olikt uppbyggda system vilket också påvisar att den besitter den önskvärda generella karaktär som vi eftersträvade. I en framtid skulle man även kunna pröva vår modell på andra typer av system för att vidare utvärdera, och eventuellt förändra eller omarbeta den på bristande punkter. Detta skulle förhoppningsvis leda till att den blev ett universellt verktyg för den organisation som har ett behov av att ta fram en arkiveringsrutin för sina system. Det skulle i så fall kunna röra sig om en organisation som redan har en fungerande gallringsrutin och vill förändra det sätt som data nu lagras på (sett ur ett formatperspektiv) eller en organisation som står inför ett helt nytt arkiveringsprojekt.

Referenser

Böcker

Jarl Backman, 1998, *Rapporter och uppsatser*, Lund, Studentlitteratur

David Brown, 1997, *Object-Oriented Analysis,* USA, R.R Donelly/Crawfordsville

Alex Ceponkus, Faraz Hoodbhoy, 1999, *"Applied XML: A Toolkit for programmers"*, USA, John Wiley & Sons

Easterby-Smith et al., 1991, *"Management research – An Introduction"*, Sage publications

Lars Mathiassen et al., 1998, *Objektorienterad systemutveckling*, LUND, Studentlitteratur

Jan Skansholm, 1996, *"C++ direkt"*, Lund, Studentlitteratur

Web-dokument

Magnus Eriksson, XMLGuiden
http://www.acc.umu.se/~alpha/xml (2001-02-18, 11:13)

Gustav Liljegren, XML-Sweden
http://www.xml.se/xml/varfor.html (2001-02-18, 11:05)

Region och Stadsarkivet i Göteborg
http://www.arkivnamnden.org/intro/gbginfo.html (2001-05-12, 17:33)

Unicode Home Page
http://www.unicode.org/ (2001-04-16, 15:38)

World Wide Web Consortium
http://www.w3.org (2001-01-25, 14:23)

Övriga källor

Miriam Björkhem, Jessica Lindhom, 2000, *"Metadata för det digitala biblioteket"*, Examensarbete (20p), Bibliteks och informationsvetenskap, Lunds universitet

Maria Lundgren, VT 1994, *"Dokumenthantering med SGML"*, Examensarbete, Systemvetenskapliga linjen, Handelshögskolan i Göteborg

Jerzy Misiewics, 1999, *"Kravspecifikation för arkivdatafiler"*, Göteborg, Regionarkivet

Marie-Louise Samuelsson, 1999, *"Lagra information på CD-R för framtiden*", Borås, SP Sveriges Provnings- och forskningsinstitut

Seminarieföreläsare, 2001, *"Digital Långtidslagring"*, Stockholm, Dokumenthuset

XML-beskrivning

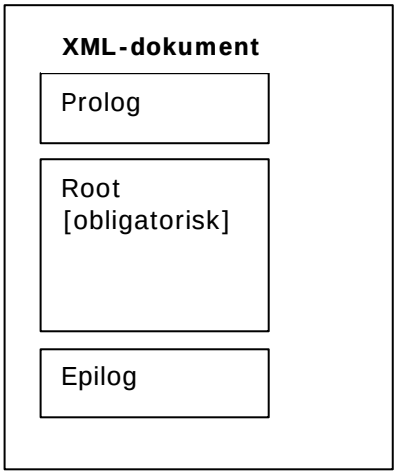
INNEHÅLLSFÖRTECKNING

1. Uppbyggnad

Ett XML-dokument är strukturerat likt ett HTML-dokument, eller kanske snarare som ett SGML-dokument. Detta innebär att det består av ”markeringar” (eng. markup) som beskriver dokumentet och datan (så kallad metadata), samt själva textinnehållet (data).

En XML-fil är egentligen inget annat än en ren textfil (eller datafil) som skrivits i antingen ASCII- eller Unicode-format, och som har ändelsen .xml.

Ett XML-dokument kan bestå av tre delar: ”prolog”, ”root” samt ”epilog”. Den enda delen som *måste* vara med är root-delen även om man som grundregel bör innefatta även en prolog.



Figur XX: XML-dokument struktur

1.1. Prolog

I prologen ges utrymme för information om XML-dokumentet som kan vara av nytta både för de program som skall hantera koden, men även underlätta för människor som skall försöka tyda den. I prologen anges exempelvis vilken version av XML-dokumentet som följer, vilken teckenstandard som används och hur dokumentet är strukturerat.

1.1.1. XML-deklarationen

Den första raden i ett XML-dokument, och i prologen, bör vara själva XML-deklarationen – dvs. en rad som talar om att det är just ett XML-dokument. Den kan se ut som följer:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

Att man inleder raden med just tecknen `<?` samt avslutar den med `?>` innebär att man deklarerar att det som står mellan dessa tecken är så kallade *Processing Instructions* (PIs). Detta är instruktioner för det program, eller den applikation, som skall behandla dokumentet. I exemplet ovan beskrivs först att det är version 1.0 av XML som används (i dagens läge finns det endast *en* version av XML men det är troligt att W3C kommer att utarbeta en uppdaterad version i framtiden).
Efter att man angivit vilken version av XML som används beskrivs vilken teckenuppsättning programmet som behandlar XML-dokumentet kommer få använda (`encoding="UTF-8"`). ”Default-värdet” (det värde som används om inget annat används) är UTF-8, vilket är komprimerad Unicode, så det behöver man egentligen inte ange. Skall man använda exempelvis svenska eller andra västeuropeiska tecken bör man kanske använda Latin-1 (ISO-8859-1 är det då som anges istället för UTF-8) även om UTF-8 dock borde fungera.

Det sista som anges på exempelraden (`standalone="yes"`) anger om dokumentet är komplett i sig självt eller om det exempelvis importerar en Document Type Definition från en annan fil (det verkar dock som om standalone-attributet till deklarationen kan komma att tas bort från XML-specen).

1.1.2. Kommentarer

Man kan även välja att skriva kommentarer i prologen. Dessa kan exempelvis beskriva vem som skapat filen, vilken version av filen det är och kanske också *när* den skapades (eller uppdaterades). Dessa kommentarer sätter man mellan taggarna `<!--` och `-->` (precis som man gör i HTML även om de där också ofta används till att inkludera exempelvis JavaScript eller VBScript). Ett exempel på detta kan vara:

```
<!-- XML-fil
skapad: 2001-03-27
av: Kalle Karlsson
-->
```

1.1.3. Processing Instructions (PIs)

Processing Instructions är, som nämndes tidigare, instruktioner till det program som skall hantera XML-dokumentet och det är ett sätt att kunna kommunicera med ett specifikt program utan att behöva referera till DTDn (beskrivs nedan) eller påverka hur andra program behandlar dokumentet. Som ett exempel på en PI kan man ta just XML-deklarationen som ju exempelvis skrivs på följande sätt:

```
<? xml version="1.0" encoding="UTF-8" standalone="yes" ?>
```

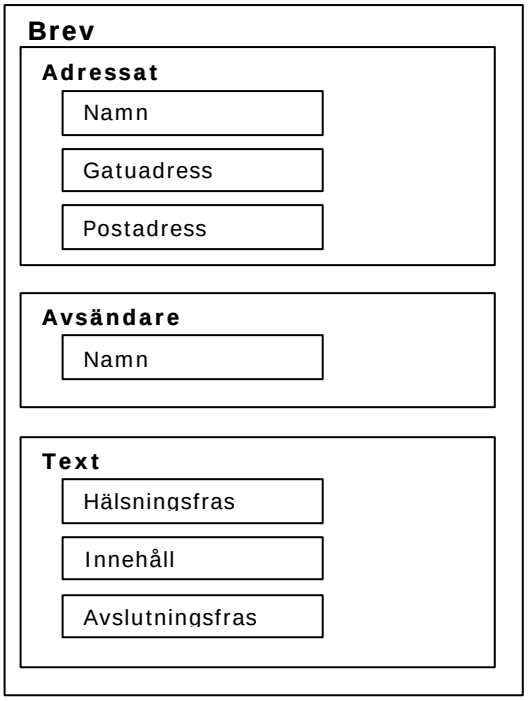
Den första strängen i en PI kallas PITarget, och hänvisar till det program som skall behandla filen, och de strängar som följer är data för instruktionen (beskrivs ovan under rubriken xml-deklarationen). PIs kan även skrivas i rooten och i epilogen och man kan rent hypotetiskt ange vilket program som helst som skall behandla xml-dokumentet men just xml-specifikationen är reserverad för xml-parsern.

1.1.4. Documemt type declaration

En *Document Type Declaration* är en deklaration i xml-dokumentet vars syfte är att det *finns* en *Document Type* ***Definition*** (DTD) samt var den finns. Det gäller här att hålla isär begreppen Document Type Declaration samt Document Type Definition. Den sistnämnda är en samling regler för hur xml-dokumentet får byggas upp och den förstnämnda är en deklaration som talar om vilken DTD som skall användas.

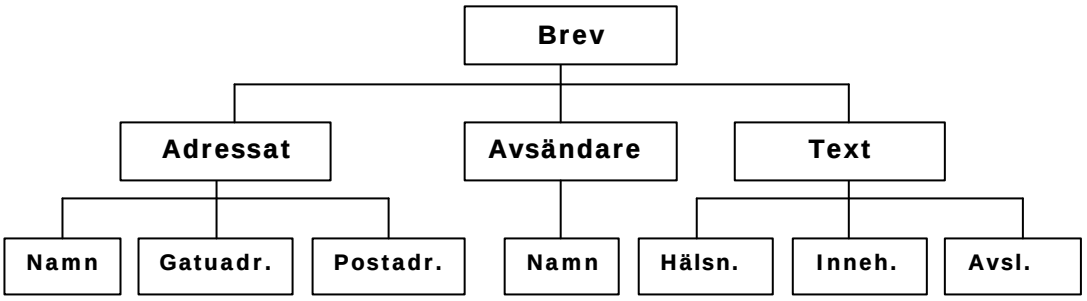
I DTDn deklarerar man alltså struktur och element (och relationen dessa emellan) för ett giltigt xml-dokument. Detta innebär att man kan ange vilka element som är tillåtna i xml-dokumentet och hur relationen mellan dem *måste* vara. Ett exempel att beskriva detta på skulle kunna vara följande: Man tänker sig att man har ett brev vars struktur och innehåll skall beskrivas med hjälp av xml.

Brevet är (enkelt sett) uppbyggt på följande vis:



Figur XX: Exempel på strukturering av ett brev

Man skulle självklart kunna gjort en mer detaljerad uppdelning – men denna kommer ändå kunna exemplifiera det vi vill. Brevet består således av tre stycken huvududelar; *Adressat*, *Avsändare* och *Text*. *Adressat* består av *Namn*, *Gatuadress* och *Postadress*. *Avsändare* består av *Namn* och *Text* består av *Hälsningsfras*, *Innehåll* och *Avslutningsfras*. En hierarkisk avbildning av detta ser ut som följer:



Figur XX: Aggregatstruktur

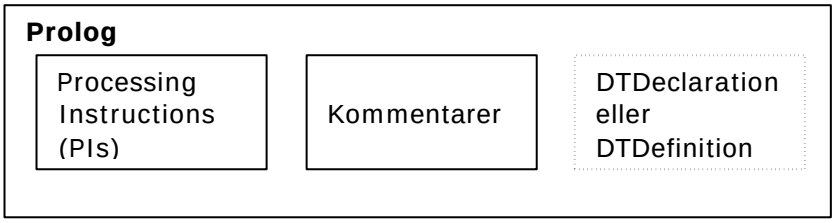
Man kan även ange huruvida ett element *måste* innehålla ett annat element eller om det är så att det *kan* innehålla det. Likaså går det att specificera om ett element skall innehålla exakt *ett*, eller *ett eller fler* element – men detta lämnar vi därhän i detta exemplet.

I en DTD definierar man alltså de element som kan/skall finnas med i xml-dokumentet samt relationen dem emellan. I vårt exempel med brevet skulle vi alltså kunna ange att vi har ett element som vi kallar *Letter* (Brev) som består av *Addressee*

(Adressat), *Sender* (Avsändare) samt *Text* (Text). Inga andra element får ligga direkt under elementet Letter. Likadant skulle vi ange reglerna för de element som är underordnade de tre element vi just beskrivit osv. På detta sätt kan man således försäkra sig om att xml-dokumentet är giltigt (*valid*) – dvs. det följer de regler som beskrivs i DTDn. Man kan även inkludera DTDn direkt i xml-dokumentet om man vill – men detta kan vara opraktiskt om man skulle behöva ändra något i DTDn och man har samma DTD för ett antal olika xml-dokument eftersom man då måste in och ändra i allihopa istället för att endast ändra i DTD-filen.

Ett prolog-exempel samt en schematisk bild av prologen ges nedan:

```
XML-deklaration:  <?xml version=”1.0” encoding=”UTF-8”?>
Kommentar:  <!-- XML-dokument skapat 2001-02-30 -->
PIs (även xml-deklarationen är en PI):    <?cb .epd?>
DTDeklaration:    <!DOCTYPE SYSTEM ”external_file.dtd”>
```



Figur XX: Prologens uppbyggnad

1.2. Root

De huvudsakliga byggstenarna i *rooten* är *element*, *entiteter*, *kommentarer*, *PIs* samt *PCData* och *CDATA*. Rooten i sig är ett element som innesluter alla tidigare nämnda delar. I följande exempel är `<exempel>` root – men det vanligaste är att man namnger rooten just ”root” eller ”document”.

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<exempel>
Detta är ett enkelt exempel.
</exempel>
```

1.2.1. Element

Nedan följer ett exempel på vad ett element är för något då vi beskriver en skiva:

```
<album>
    <artist>Foo Fighters</artist>
    <titel>The color and the shape</titel>
    <format typ=”CD”/>
    <antal_låtar>13<antal_låtar/>
</album>
```

Här är ”titel” en så kallad *tag*. Området från start-tag till slut-tag – `<titel>The color and the shape</titel>` - är ett element och texten ”The color and the shape” är elementets innehåll. Man kan själv välja vad elementet skall heta så länge som man ser till att ha en start-tag och slut-tag som är riktiga och det namn man väljer har endast semantisk betydelse för en själv (och de andra som eventuellt skall läsa koden). Man måste dock se till att man inte har några mellanslag i namnet. Således skulle inte följande element-namn vara godkänd:

```
<titel på skivan>
```

medan

```
<titel_på_skivan>
```

samt

```
<titel.på.skivan>
```

går bra. Dock kan man ju diskutera lämpligheten med att ha svenska tecken med. I dagsläget är det endast svenska försvaret som har å, ä och ö med i sina elementnamn i sina SGML-dokument.

Ett element behöver dock inte innehålla någon text alls. Exempel på detta är där man anger formatet på skivan:

```
<format typ=”CD”/>
```

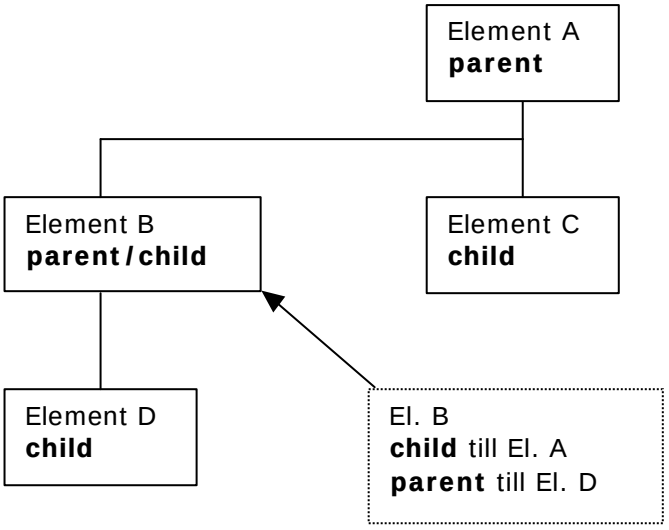
Observera då att man måste avsluta elementnamnet med ”/” –tecknet men också att elementet inte ens behöver innehålla något *attribut* så man hade även kunnat skriva:

```
<format/>
```

fast då hade det ju inte framgått att det var just en CD-skiva.

Ett element är den minsta delen av ett dokument som XML behandlar, och det är den huvudsakliga byggstenen av dokumentet. Ett element kan innehålla data (text), subelement, data och subelement eller helt enkelt vara tomt. När ett element befinner sig inom ett annat element kallar man detta element för *child* (barn) och det andra elementet för *parent* (förälder). I exemplet ovan är `album` ”parent” och exempelvis

`artist` ”child”. Det finns dock inget som hindrar att ett element är både parent och child om man har en hierarki med tre eller fler led. Beroende på vilka element som relateras till varandra blir de således antingen parent eller child vilket visualiseras i figur XX.



Figur XX: Hierarkisk struktur med ”parent”-”child” förhållande

1.2.2. Attribut

Vi behandlade attribut i exemplet med skivan ovan men skall nu komplettera detta lite grand. Attribut har ett *värde* och angavs alltså inom starttagen – exempelvis
`<album typ="CD">`
Restriktioner för värden som attribut får ha anger man i DTDn där man också kan ange om ett attribut skall ha något *default* -värde – dvs. ett värde som antas om inget annat anges.

Det finns idag två fördefinierade attribut enligt XML-specifikationen vilka kan användas till vilka element som helst, var som helst utan att det specificeras i DTDn. Dessa är:
`xml:space`
samt
`xml:lang`
där namnet **`xml:`** är reserverat för framtida inbyggda attribut.

I XML bevaras mellanslag automatiskt men ibland ignoreras detta av applikationen som hanterar dokumentet. Vill man således tala om för exempelvis en webbläsare att den skall bevara alla ”mellanslag” när den visar innehållet i ett xml-dokument kan man göra detta genom att inkludera attributet `xml:space` samt tilldela det värdet `"preserve"`. Detta kan vara en bra idé att göra om man exempelvis vill bevara läsbarheten i programkod. Jämför hur de två programsnuttarna skulle visas om `xml:space="preserve"` inkluderas eller ej:

```
<prog.kod xml:space="preserve">
  for(i=0; i<=12; i++) {
    write(Ord(i));
  }
</prog.kod>
xml:space="preserve" inkluderas:
for(i=0; i<=12; i++) {
    write(Ord(i));
  }
```

xml:space=”preserve” inkluderas *inte*:
`for(i=0; i<=12; i++) { write(Ord(i)); }`

När det gäller `xml:lang` så avser detta att specificera vilket språk dokumentet är skrivet i. Idealt skall värdet sättas till den förkortning som anges i ISO-639 standarden – exempelvis `"sv"` för svenska osv.

Detta används till exempel för att program som hanterar rättstavning av svenska ord skall veta vilka delar av en text det skall rätta, eller för en sökmaskin som skall kunna söka endast på delar i ett visst språk.

1.2.3. Entiteter

Entiteter är innehåll, till skillnad mot bland annat element. Detta innehåll är vanligtvis text, men det kan även vara binär data som exempelvis bilder och ljud. Man kan säga att deras huvudsakliga användning är när man vill förenkla inmatning i ett dokument. Exempelvis vill man kanske föra in strängen `”Institutionen för Informatik”` på ett antal ställen i ett dokument men kan då deklarera följande i DTDn:

```
<!ENTITY ifi ”Institutionen för Informatik”>
```

Vill man sedan införa strängen `”Institutionen för Informatik”` någonstans i rooten på dokumentet behöver man endast skriva `&ifi;` (`&entitetens_namn;`) så ersätts detta med entitetens deklarerade innehåll.

1.2.4. Kommentarer

Liksom i prologen kan man lägga in kommentarer för att förklara eller förtydliga information i ett dokument. Läs mer om hur kommentarer fungerar under Prolog-avsnittet.

1.2.5. Processing Instructions (PIs)

Processing Instructions kan liksom kommentarer inkluderas både i prologen och rooten. Läs mer om hur PIs fungerar under Prolog-avsnittet.

1.2.6. PCDATA

PCDATA står för Parsed Character Data och är default-typen för all text i ett XML-dokument. Man kan säga att all text i rooten som exkluderat den inom vinkelparenteserna (< och >) är PCDATA (och det övriga är så kallad *Markup*).

1.2.7. CDATA

CDATA står för Character Data och används när man vill innefatta text i ett XML-dokument som skulle kunna tolkas som *markup* – men som inte är det. Om man exempelvis skulle vilja ha med vinkelparenteser (< eller >), som vanligtvis tolkas som det som inleder eller avslutar en tag, skulle man vara tvungen att ange deras entitetsreferenser (`<` samt `>`) istället. Att sitta och ändra all ställen i ett XML-dokument kan vara ganska arbetsamt och istället för att ha en kod som ser ut så här:

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<exempel>
Så här kan ett XML-dokument inledas:
&lt;?xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot;
standalone=&quot;yes&quot;&gt;
</exempel>
```

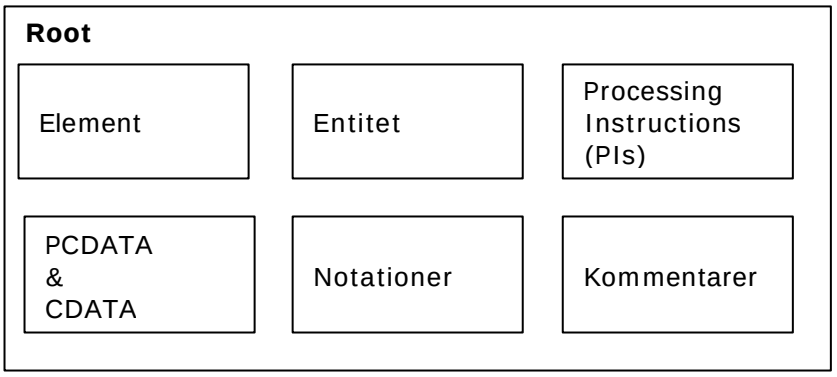
kan man istället ange att en sektion som följer inte skall tolkas som annat än text vilket skulle ge ovanstående kod följande utseende;

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<exempel>
<!CDATA[
Så här kan ett XML-dokument inledas:
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
]]>
</exempel>
```

1.2.8. Notationer

En XML-parser kan bara förstå och hantera text. Vill man inkludera annan typ av innehåll som exemeplvis bilder eller ljudfiler måste dessa refereras till genom notationer. Markup innehåller ju inte bilder eller ljud i sig självt utan talar bara om för exempelvis webbläsaren var bilden finns. Genom att använda notationer kan man specificera vilket program som skall ta hand om viss data. Notationer deklareras också i DTDn.

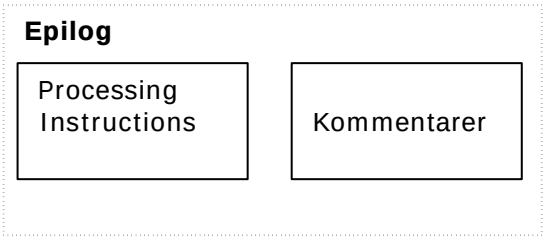
Följande är en schematisk bild över rooten:



Figur XX: Rootens uppbyggnad

1.3. Epilog

Epilogen kan inte sägas fylla någon egentlig funktion eftersom man faktiskt kan skriva allt det som skulle kunna ha stått i epilogen i prologen. Termen epilog är inte heller direkt använd i XML-specifikationen där det endast står att viss markup får förekomma efter rooten. Av dess skäl rekommenderas egentligen inte att man skriver någon epilog, även om det är tillåtet enligt specifikationen. Det som får stå i epilogen, dvs. efter rooten, är Processing Instructions och kommentarer.



Figur XX: Epilogens uppbyggnad

2. XML - dokumentens struktur (Översikt)