Enhetlig tilgang til heterogene metadatabaser

Interoperabilitet v.h.a. OAI-PMH

Eskil Høyen Solvang

Hovedfagsoppgave i informasjonsforvaltning
Institutt for datateknikk og informasjonsvitenskap
Norges teknisk-naturvitenskapelige universitet

Veileder: Ingeborg T. Sølvberg

August 2004

I uvissa ligg der
ein skoddelagt topp.
Du undrast med skepsis
om du vil nå opp.

Du ser ikkje klårt,
du finn ingen spor.
Din beste venn Fokus
han sovna i fjor.

Men mist ikkje motet
når ting går i stå.
Når tida flyg frå deg
og angst bankar på.

For brått vil du sanse
at motvinden snur.
Og toppen kan skimtast
i uklår kontur.

Når Fokus kjem att
er han skjerpa og klår.
Han veit kvar du skal,
og han veit kvar du står.

Så skund deg og gå,
mens du enno har tid.
Om skodda kjem att,
er sjansen forbi

Du hastar av garde
med målretta steg.
Kun oppløpet gjenstår
av din lange veg.

Og når du kjem fram
så innser du brått.
At vegen var målet
og målet er nått.

Eskil Høyen Solvang

Oppsummering

Denne avhandlingen fokuserer på hvordan informasjon fra mange forskjellige samlinger kan gjøres lettere tilgjengelig for informasjonsbrukere. I dagens informasjonssamfunn finnes og produseres det store mengder digital informasjon, og mange organisasjoner og bedrifter anvender derfor beskrivende metadata til å organisere informasjonen sin i mange ulike samlinger/databaser. Dette fører til at personer som ønsker å søke etter informasjon, må forholde seg til tilsvarende mange forskjellige søkesystemer og søketeknikker. Fra disse brukernes ståsted er det ønskelig med et søkesystem som gjør det mulig å søke etter informasjon fra et stort antall samlinger/databaser på ett sted.

Problemstillingen som ligger til grunn for oppgaven er todelt. For det første skal det undersøkes hvordan man kan utforme en arkitektur som muliggjør samtidig søking i flere heterogene metadatabaser. For det andre skal det undersøkes hvordan denne arkitekturen kan anvendes til å bygge opp et enhetlig søkegrensesnitt som gir brukeren tilgang til ressursbeskrivelser fra alle de heterogene metadatabasene.

Forutsetningen for oppgaven er at arkitekturen skal baseres på metadatahøsting ved hjelp av OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting). Dessuten skal Fast Data Search anvendes som indekserings- og søkesystem.

> *I denne avhandlingen presenteres et forslag til et system som, basert på metadatahøsting fra flere forskjellige datakilder og påfølgende sentral indeksering av de innhøstede metadataene, tilbyr en enhetlig søketjeneste. Dette gjør det i praksis mulig å søke i innholdet fra mange datakilder gjennom ett felles søkegrensesnitt.*

Datagrunnlaget i prototypen utgjøres av om lag 1400 metadatabeskrivelser som blir høstet fra tre av Nasjonalbibliotekets samlinger; Digitalt Radioarkiv, Galleri NOR og Mavis. Metadatahøstingen utføres av en OAI-høster som er en videreutvikling av en fritt tilgjengelig høsterapplikasjon med

åpen kildekode. Alle metadataene er beskrevet i formatet MODS (Metadata Object Description Schema), som er et mer uttrykksfullt alternativ til Dublin Core. Etter en enkel normalisering, blir metadataene indeksert i Fast Data Search. Et webbasert søkegrensesnitt, med mulighet for både enkelt og avansert søk, gjør det mulig for brukerne å søke i alle de indekserte metadataene.

Hovedkonklusjonen er at metadatahøsting ved hjelp av OAI-PMH er en velegnet metode for å utvikle en felles søketjeneste for informasjon fra mange datakilder. Metoden fungerer også internt i en organisasjon/bedrift. Det er likevel viktig å være bevisst på at eksterne faktorer, som for eksempel variasjoner i metadatakvalitet, kan påvirke kvaliteten på søketjenesten. Når metoden anvendes internt i en organisasjon/bedrift, kan det derfor være nyttig å undersøke datakildenes kvalitet på forhånd. På denne måten kan uegnede datakilder utelukkes på et tidlig tidspunkt.

Summary

This thesis focuses on how information from a lot of different collections can be made more easily available for information users. Today's information society contains large amounts of digital information, and many organizations and enterprises therefore employ descriptive metadata when organizing their information in several different collections/databases. This diversity forces people wanting to search for information to deal with a corresponding number of different search systems and techniques. For these users, a search system which makes it possible to locate information from a large number of collections/databases at single spot is desirable.

The problem to be addressed in this thesis is split. Firstly it has to be examined how to model an architecture which supports simultaneous searching in heterogeneous metadatabases. Secondly it has to be examined how this architecture can be employed to build a uniform search interface which gives the user access to resource descriptions from all the heterogeneous metadatabases.

A premise for the thesis is that the architecture should be based on metadata harvesting by means of the OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting). In addition Fast Data Search should be used as the indexing and searching system.

> *This thesis presents a system suggestion which offers a uniform search service, based on metadata harvesting from several different data sources and subsequent central indexing of the harvested metadata. In practice this makes it possible to search the content of several data sources using one common search interface.*

The data foundation in the prototype is constituted by about 1400 metadata descriptions harvested from three collections at the National Library of Norway: Digitalt radioarkiv (Digital Radio Archive), Galleri NOR (photo gallery) and Mavis (multimedia database). The harvesting process is performed by an OAI harvester which is an adaptation of an open source harvester application. All metadata is described using MODS (Metadata Object Description Schema), which is a more

expressive alternative to Dublin Core. After a simple normalization process, the metadata is indexed by Fast Data Search. A web based search interface, enabling both basic and advanced search options, makes it possible for the users to search the indexed metadata.

The main conclusion is that metadata harvesting by means of OAI-PMH is a suitable method for developing a common search service for information from various data sources. The method is also applicable for internal use in an organization or enterprise. However, it is important to be conscious about external factors that may affect the quality of the search service, for instance variations in metadata quality. When the method is used internally in an organization/enterprise, it may be useful to examine the quality of the data sources in advance. By doing this, unsuitable data sources can be excluded early in the development process.

Innhold

Figurliste

Tabell-liste

KAPITTEL 1

Problembeskrivelse og mål

Interoperabilitet mellom heterogene samlinger er ett av mange aktuelle emner innen internasjonal forskning på digitale bibliotek. Målet med interoperabilitet er å bygge koherente tjenester for brukerne ut fra komponenter som er teknisk ulike og som har forskjellig innhold.

> *Users don't want to learn several search syntaxes, they don't want to install a variety of viewing applications on their desk, and they want to make a single query that accesses a variety of different repositories. Users want to access an interoperable information world, where a set of separate repositories looks to them like a single information portal. [BES02]*

Nasjonalbiblioteket har mer enn 80 forskjellige databaser/samlinger innen en rekke ulike emner. Beskrivelser av disse databasene og mulighetene for å søke i dem er tilgjengelig for publikum, blant annet gjennom world wide web *[NBDB]*. En vanlig bruker er normalt interessert i å finne informasjon - i form av dokumenter - uavhengig av hvordan dokumentene er beskrevet ved hjelp av metadata og hvor dokumentene er lagret. I dag er imidlertid objektene i de forskjellige databasene beskrevet ved hjelp av mange ulike metadataformater, og brukeren må dessuten selv finne ut hvilken database som er mest relevant å søke i før han/hun utfører selve søket. For uerfarne brukere kan det derfor være problematisk å orientere seg i Nasjonalbibliotekets databaselandskap.

1.1 Problemstilling

Oppgavens hovedmål er å gjøre informasjon fra flere forskjellige samlinger lettere tilgjengelig for brukere. Nasjonalbiblioteket ønsker i den forbindelse å bygge en løsning for interoperabilitet mellom sine databaser, slik at de framstår som mer brukervennlige for publikum. En av hovedutfordringene er å gi publikum muligheten til å søke i flere metadatabaser gjennom ett felles søkevindu. I første omgang er det viktig for Nasjonalbiblioteket å få belyst dette problemet på grunn av dets rolle som nasjonalt informasjons- og kunnskapsknutepunkt. Etter hvert vil imidlertid problemstillingen også kunne være interessant for andre større aktører innen digital informasjonsforvaltning.

Problemstillingen kan formuleres som følger:

- Hvordan kan en arkitektur som muliggjør samtidig søk i flere heterogene metadatabaser utformes?
- Hvordan kan et søkegrensesnitt som nyttiggjør seg denne arkitekturen oppbygges, slik at man gjennom ett enhetlig søkegrensesnitt kan få tilgang til ressursbeskrivelser fra alle de heterogene metadatabasene?

1.2 Forutsetninger

Det forutsettes at løsningsforslaget baseres på følgende teknologier:

- OAI-PMH (Open Archive Initiative Protocol for Metadata Harvesting)
- Fast Data Search

Løsningen skal baseres på Open Archive Initiative sin protokoll for metadatahøsting, OAI-PMH, som har to typer aktører; datatilbydere (data providers) og tjenestetilbydere (service providers). I denne sammenhengen vil Nasjonalbiblioteket fylle begge rollene, siden hver enkelt database/samling vil være en datatilbyder, mens søkegrensesnitt vil fungere som tjenestetilbyder til brukerne. Metadataene hos hver enkelt datatilbyder tilgjengeliggjøres i et felles format gjennom et såkalt OAI datalager (repository). Tjenestetilbyderen benytter en OAI-høster til å sende forespørsler til de aktuelle datatilbyderene og hente xml-kodete metadata fra deres datalagre via http-protokollen.

Løsningen skal også baseres på søkemotoren og søkeprogramvaren Fast Data Search fra Fast Search & Transfer. Når OAI-høsteren har samlet inn metadataene i en felles fil, har OAI-protokollen gjort sitt. For å kunne søke i den innhøstede metadatasamlingen, må dataene indekseres av et søkesystem. Fasts indekseringsprogram krever at data som skal indekseres foreligger i et eget format, FastXML. Det skal derfor benyttes et stilark til å omforme de innhøstede metadataene fra det opprinnelige metadataformatet til FastXML før indeksering.

Det skal dessuten utvikles et felles søkegrensesnitt som gjør det mulig å søke i de indekserte metadataene fra de forskjellige databasene på ett sted. I denne sammenheng kan det være interessant å undersøke hvilke faktorer som påvirker kvaliteten på søketjenesten.

Oppsummert kan man si at følgende konkrete oppgaver er viktige i det videre arbeidet:

- Definere ett felles format for metadata for de forskjellige databasene.
- Utvikle/videreutvikle OAI-høsteren som skal samle inn metadata fra de forskjellige databasene.
- Indeksere innsamlede metadata, slik at det blir mulig å søke i dem.
- Utvikle ett søkegrensesnitt som gjør det mulig å søke i de indekserte metadataene.

KAPITTEL 2

Informasjonsobjekter, metadata og interoperabilitet

Digitale bibliotek er - i et informatikkperspektiv - basert på ulike teknologier som virker sammen. Hvilke teknologier som benyttes, bestemmes i stor grad av hvilken funksjonalitet det digitale biblioteket skal tilby brukerne.

Målet i denne oppgaven er å gjøre informasjon lettere tilgjengelig ved å lage et felles søkegrensesnitt mot mange samlinger. I dette kapittelet beskrives flere ulike begreper og teknologier som ligger til grunn for å oppnå dette målet:

- Metadata
- Metadataformater
- Interoperabilitet
- Metadatahøsting som interoperabilitetsmetode
- OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting)

Selv om det viktigste begrepet i dette kapittelet er interoperabilitet mellom samlinger, kreves først en introduksjon av begrepene metadata og metadataformater. Protokollen OAI-PMH beskrives også i nærmere detalj.

2.1 Metadata

Begrepet metadata kan kort defineres som ”data om data”. I informatikksammenheng har imidlertid begrepet ingen entydig definisjon, og det tillegges forskjellige betydninger både av forskjellige brukergrupper og i forskjellige sammenhenger. For å unngå tvetydigheter, er det viktig å beskrive hvilken forståelse av begrepet som ligger til grunn i denne oppgaven, hvor konteksten er digitale bibliotek.

2.1.1 Definisjon

Generelt innen informasjonsteknologi betyr prefikset *meta-* en underliggende definisjon eller beskrivelse, og *metadata* kan således betraktes som en underliggende definisjon eller beskrivelse av

data. Når konteksten er digitale bibliotek, betegnes ofte datainnholdet som informasjonsobjekter, og dette utgjør også grunnlaget for en mer spesifikk definisjon av begrepet metadata. I forbindelse med digitale bibliotek kan derfor metadata defineres som strukturerte data som er assosiert med informasjonsobjekter på en slik måte at potensielle brukere får kunnskap om objektenes eksistens eller karakteristikk.

Et informasjonsobjekt[1] er i denne sammenhengen en enhet som inneholder informasjon, og defineres av *[GIL00]* som alt som kan identifiseres og manipuleres som en atskilt enhet av et menneske eller et system. Informasjonsobjektet kan bestå av ett enkelt element, for eksempel et bilde, eller det kan være en sammensetning av flere elementer, for eksempel en webside. Alle informasjonsobjekter har følgende tre egenskaper, som alle kan reflekteres gjennom metadata:

- Innhold. Beskrivelse av innholdet i informasjonsobjektet.
- Kontekst. Aspekter knyttet til skapelsen av informasjonsobjektet.
- Struktur. Assosiasjoner innen et informasjonsobjekt og mellom objekter.

I denne oppgaven er det spesielt metadata som beskriver informasjonsobjekters innholdsegenskaper som er interessante, ettersom denne typen metadata har størst nytteverdi i søkesammenheng.

2.1.2 Oversikt over ulike typer metadata

Selv om definisjonen ”strukturerte data assosiert med informasjonsobjekter” skaper en oppfatning av hva begrepet metadata betyr, er dette imidlertid en relativt generell definisjon. I realiteten eksisterer det flere forskjellige typer strukturerte data for forskjellige bruksområder og sammenhenger. *[GIL00]* inneholder en oversikt over forskjellige metadatakategorier og deres egenskaper.

Tabell 2-1. *Forskjellige typer metadata og deres funksjon*

Type	**Definisjon**	**Eksempel**
Administrative	Metadata som brukes til forvaltning og administrasjon av informasjonsressurser	Informasjon om tilegnelse Rettigheter Lokaliseringsinformasjon
Deskriptive	Metadata som brukes for å beskrive eller identifisere informasjonsressurser	Katalogposter “Finding aids” Spesialiserte indekser Hyperlenkede relasjoner mellom ressurser
Bevaring	Metadata relatert til bevaringsforvaltning av informasjonsressurser	Dokumentasjon av hva som er gjort for å bevare ressurene

1. I denne oppgaven benyttes også begrepet informasjonsressurs, eller bare ressurs, i samme betydning som informasjonsobjekt.

Tabell 2-1. *Forskjellige typer metadata og deres funksjon (forts.)*

Type	Definisjon	Eksempel
Tekniske	Metadata relatert til hvordan et system virker eller metadata oppfører seg	Maskinvare og programvare Digitaliseringsinformasjon Sikkerhetsdata
Bruk	Metadata relatert til bruksnivå og -type for informasjonsressurser	Bruk og brukersporing

2.1.3 Metadata og søking

Det faktum at metadata kan grupperes i forskjellige kategorier, kan anses som en indikasjon på at det eksisterer flere ulike hensikter med slike strukturerte beskrivelser. Ettersom de forskjellige kategoriene dessuten kan ha forskjellig nytteverdi for forskjellige brukergrupper, er det ikke mulig å fastslå ett entydig formål med metadata. Eksempler på relevante formål er:

- Etablere konteksten informasjonsobjekter opptrer i
- Identifisere relasjoner til andre informasjonsobjekter
- Beskrive hvordan informasjonsobjekter skal forvaltes
- Tilby intellektuelle aksesspunkter til informasjonsobjektene for forskjellige typer brukere
- Øke tilgjengelighet
    - Ressursoppdaging/-gjenfinning
    - Lokalisering
- Muliggjøre evaluering og eventuelt utvelgelse av bestemte ressurser

Det overordnede målet med dette prosjektet er å lage et samlet søkesystem for metadata fra flere forskjellige databaser, det vil si å gjøre det mulig for en bruker å søke i metadata fra forskjellige kilder på ett sted.

Når metadata skal brukes i søkesammenheng, har deskriptive metadata størst nytteverdi siden denne typen metadata beskriver innholdet i informasjonsobjekter. I fortsettelsen vil det derfor i hovedsak bli fokusert på denne typen metadata. I *[GILS]* konstateres det dessuten at selv om metadata kan brukes til å beskrive svært mange egenskaper ved et informasjonsobjekt, er det kun et lite antall av disse egenskapene som regelmessig brukes i informasjonssøking. Disse egenskapene fungerer som aksesspunkter til informasjonsobjektene, noe som igjen resulterer i økt tilgjengelighet.

I et søkesystem hvor det skal være mulig å søke i metadata for forskjellige typer ressurser, bør det benyttes så generelle aksesspunkter at en bruker kan få tilgang til beskrivelsene for alle typer ressurser gjennom de samme aksesspunktene. Eksempler på slike aksesspunkter som er felles for alle typer ressurser er:

- Tittel
- Navn
- Emneord
- Dato
- Sted
- Abstract (tekstlig ressurssammendrag)

Disse metadataelementene er godt egnet som aksesspunkter både fordi de er av en så generell karakter at de beskriver egenskaper som de aller fleste informasjonsressurser har og fordi de er relativt selvforklarende. De aller fleste metadataelementene er imidlertid mindre egnet som aksesspunkter til informasjonsobjekter, hovedsakelig fordi de beskriver egenskaper som man vanligvis ikke søker etter, eller fordi de er knyttet til bestemte ressurstyper. Dersom et søkesystem inneholder aksesspunkter som bare er knyttet til bestemte ressurstyper, eller som har forskjellig betydning for forskjellige ressurstyper (for eksempel *extent* som vanligvis betyr varighet for lydopptak og antall sider for tekst), kan dette resultere i utelukkelse av enkelte ressurstyper allerede før søkeprosessen starter.

2.1.4 Forholdet mellom metadata, data og informasjon

Begrepet metadata bør betraktes som et kontekstavhengig begrep. Mer konkret betyr dette at det som er metadata i én sammenheng, kan framstå som data i en annen sammenheng og informasjon i en tredje sammenheng. Dette påpekes også i *[GIL00]*, hvor det står at "One information object's metadata can simultaneously be another information object's data."

Begrepene kan kort forklares som følger:

- Data: Informasjonsbærere (tall/bokstaver) som eksisterer uten noen form for kontekst. Begrepet data benyttes først og fremst i forbindelse med lagring og prosessering.
- Metadata: Strukturerte data assosiert med et informasjonsobjekt.
- Informasjon: Data plassert i en kontekst. Begrepet informasjon brukes først og fremst i forbindelse med presentasjon av opplysninger.

I dette prosjektet er utgangspunktet metadata for informasjonsobjekter som befinner seg i forskjellige databaser. Når disse opplysningene høstes inn, indekseres og til slutt gjøres søkbare, plasseres imidlertid disse metadataene i en brukskontekst hvor opplysningene framstår som selvstendige og uavhengige av de opprinnelige objektene. I dette perspektivet kan metadataene derfor betraktes som informasjon. I løpet av oppgaven vektlegges en konsistent bruk av begrepet metadata, men ordene data og informasjon benyttes også i enkelte sammenhenger der dette er mer passende.

2.2 Metadataformater

Et metadataformat består ifølge *[KRI01]* av to komponenter:

1. Et vokabular av termer og deres betydning (semantikk), for eksempel title og ”a name given to the resource” (Dublin Core).
2. En syntaks som binder disse termene sammen, for eksempel uttrykt i xml.

I tillegg eksisterer det ofte et sett regler som definerer hvordan innholdet i de forskjellige termene skal uttrykkes.

I likhet med at det eksisterer mange kategorier av metadata, eksisterer det også mange ulike metadataformater med forskjellige vokabularer og syntakser. Mange av metadataformatene gir rom for å beskrive metadata fra flere forskjellige kategorier, men det eksisterer også metadataformater som kun er konsentrert om én metadatakategori. Enkelte av metadataformatene er generelle, det vil si de kan brukes til å beskrive strukturerte data om alle typer informasjonsobjekter. Andre metadataformater er spesielle, det vil si tilpasset til et bestemt emneområde eller en bestemt type informasjonsobjekter, for eksempel video.

I tillegg eksisterer det variasjoner mellom de ulike formatene når det gjelder detaljnivået - granulariteten - i metadatabeskrivelsene. Dette vil si at enkelte metadataformater (for eksempel Dublin Core) kan brukes til å lage relativt overflatiske beskrivelser av informasjonsobjekter, mens andre formater (for eksempel MARC) kan brukes til å beskrive mer detaljerte opplysninger om informasjonsobjekter. I *[WEI02]* påpekes det at detaljerte metadatabeskrivelser kan medvirke til økt søkepresisjon, men at det ofte kreves relativt store ressurser for å utvikle denne typen beskrivelser. Enkle metadatabeskrivelser øker sannsynligheten for kryssdisiplinær interoperabilitet, men kan samtidig føre til at informasjonssøkere må yte en større innsats for å identifisere de mest relevante søkeresultatene.

Som beskrevet i *kap. 2.1.3, s. 5*, er det i dette prosjektet formater for deskriptive metadata som er interessante. Det eksisterer flere forskjellige formater for denne typen metadata, men i denne sammenhengen anses følgende formater som mest relevante:

- Dublin Core og Dublin Core Library Application Profile
- MARC
- MODS

2.2.1 Dublin Core

Dublin Core (DC) er et metadataformat som er utviklet med tanke på å lette søkingen etter og gjenfinningen av elektroniske ressurser, hovedsakelig på internett. Dublin Core er et generelt og relativt enkelt format som kan brukes til å beskrive alle typer ressurser. Vokabularet i Dublin Core Metadata Element Set (DCMES) består av 15 termer[2], og kan uttrykkes ved hjelp av forskjellige syntakser (tekst, html meta-tag, xml eller rdf/xml).

2. Metadataelementet *Audience* (“A class of entity for whom the resource is intended or useful”) inngår ikke direkte i DCMES, men beskrives likevel som et ekstra Dublin Core-element i offisielle dokumenter.

Tabell 2-2. *Metadataelementer i Dublin Core Metadata Element Set*

Metadataelement	**Definisjon**
Title	Et navn tilordnet ressursen
Creator	En entitet som er ansvarlig for å lage innholdet i ressursen
Subject	Emneord for innholdet i ressursen
Description	En beskrivelse av innholdet i ressursen
Publisher	En entitet som er ansvarlig for å gjøre ressursen tilgjengelig
Contributor	En entitet som er ansvarlig for å bidra til innholdet i ressursen
Date	En dato for en hendelse i ressursens livssyklus
Type	Ressursinnholdets sjanger/natur
Format	Den fysiske eller digitale manifestasjonen av ressursen
Identifier	En utvetydig referanse til ressursen innenfor en gitt kontekst
Source	En referanse til en ressurs som den beskrevne ressursen er avledet fra
Language	Språket i ressursens intellektuelle innhold
Relation	En referanse til en relatert ressurs
Coverage	Omfang eller definisjonsområde for ressursens innhold
Rights	Informasjon om rettigheter knyttet til ressursen

Som det framgår av *tabell 2-2*, beskriver metadataelementene i Dublin Core egenskaper ved ressursers innhold, opphavsrett og manifestasjon. Ingen av metadataelementene er obligatoriske i en metadatabeskrivelse, og elementene kan enten brukes i den enkle formen eller de kan beskrives mer spesifikt ved hjelp av *kvalifikatorer*. Dette er en mekanisme hvor man enten benytter elementraffinering (creator.photographer) eller enkodingsskjemaer (language i hht. ISO 639-1) for å beskrive semantikken i et element mer spesifikt. På grunn av fare for misforståelser, anbefales det i [LAG01a] å benytte den enkle formen av Dublin Core i interoperabilitetsprosjekter.

Eksempel på metadatabeskrivelse i Dublin Core, uttrykt ved hjelp av xml:

```
<metadata>
     <dc:title>
          Norsk Jernverk - Gøthe-utvalget foreslår nedleggelse. EKKO
     </dc:title>
     <dc:creator>Gravir, Ketil</dc:creator>
     <dc:subject>Jernverk</dc:subject>
```

```
        <dc:subject>Jern- og stålindustri</dc:subject>
        <dc:subject>Industripolitikk</dc:subject>
        <dc:subject>RANA Mo</dc:subject>
        <dc:description>
             Gøthe-utvalget (Statens forhandlingsutvalg) foreslår å
             legge ned stålproduksjonen ved Norsk Jernverk i Mo i Rana.
             Odd GØTHE (mv) kommenterer kritikken til utvalgets
             innstilling. Om de økonomiske sidene. Den største ut-
             fordring industri-Norge har stått overfor etter krigen. Om
             mulighetene for annen industri i Mo, med lønnsom produk-
             sjon, f.eks. et aluminiumsverk med privat finansiering.
        </dc:description>
        <dc:publisher>NRK</dc:publisher>
        <dc:date>1988-04-05</dc:date>
        <dc:type>Radioprogram</dc:type>
        <dc:identifier>URN:NBN:no-nb_drl_20427 1787</dc:identifier>
        <dc:rights>NRK</dc:rights>
    </metadata>
```

2.2.2 Dublin Core Library Application Profile

Dublin Core Library Application Profile *[DC-Lib]* er kort forklart en spesifikasjon av hvordan Dublin Core bør brukes i bibliotekssammenheng, det vil blant annet si hvilke metadataelementer som kan brukes, hvilke verdier som er tillatte og eventuelt hvilke eksterne metadataelementer som kan benyttes. Før ytterligere detaljer om DC Library Application Profile beskrives, introduseres imidlertid begrepet applikasjonsprofil.

Applikasjonsprofil

Begrepet applikasjonsprofil defineres i *[HEE00]* som et metadataskjema som består av dataelementer hentet fra ett eller flere navnerom, og hvor disse dataelementene er kombinert av utviklere og optimalisert for en bestemt lokal applikasjon.

Ifølge *[BAK02]* besvarer en applikasjonsprofil spørsmålene

- Hvilke termer benyttes i metadataene dine?
- Hvordan benyttes disse termene?

En applikasjonsprofil har følgende kjennetegn (*[HEE00]* og *[DCMI03]*):

- Gjenbruker dataelementer fra ett eller flere eksisterende navnerom/elementsett
- Introduserer ingen nye dataelementer og definisjoner
- Kan spesifisere tillatte skjemaer og verdier
- Kan raffinere/forfine standarddefinsjoner (semantisk innstramming)
- Kan inneholde tilleggsdokumentasjon om begrensninger for, samt enkodingsmåter eller tolkninger for elementene

Applikasjonsprofiler er nyttige siden de gjør det mulig for utviklere å lage retningslinjer for hvordan metadataelementer faktisk blir brukt, og for hvordan de kombineres med metadataelementer

fra andre formater. Hensikten med slike applikasjonsprofiler er med andre ord å etablere en felles kontekst for hvordan metadataformater, som for eksempel Dublin Core kan brukes til bestemte formål.

Ønsket om å oppnå interoperabilitet (se *kap. 2.3, s. 20*) er hovedmotivet for å ta i bruk applikasjonsprofiler. I *[HEE00]* hevdes det at "By defining application profiles and, most importantly by declaring them, implementors can start to share information about their schemas in order to inter-work with wider grouping". Det samme poenget blir også fastslått i *[DCMI03]*, hvor applikasjonsprofiler anses som en form for dokumentasjon hvor hensikten er å hjelpe ulike implementasjonsgrupper til å etablere en felles praksis.

Det endelige målet er å utvikle maskinprosesserbare applikasjonsprofiler basert på datamodeller som for eksempel RDF (Resource Description Framework). På denne måten vil funksjoner for metadatainteroperabilitet, for eksempel semantiske kryssinger eller formatkonvertering, kunne automatiseres.

DC Library Application Profile

Dublin Core Library Application Profile (DC-Lib) er i denne sammenhengen et forslag til retningslinjer for hvordan metadataelementene i Dublin Core kan brukes til å beskrive biblioteksressurser. Applikasjonsprofilen er fremdeles under utvikling, og har foreløpig status som “DCMI working draft”.

I DC-Lib defineres:

- Påkrevde, anbefalte og valgfrie DC-elementer
- Tillatte DC-kvalifikatorer
- Tillatte enkodingsskjemaer og verdier (f.eks. bruk av spesifikke kontrollerte vokabularer)
- Innholdsregler/god skikk
- Tilleggselementer for biblioteksdomenet fra andre navnerom enn DC, for eksempel MODS
- Raffinering av standarddefinisjoner

Dublin Core Library Application Profile (DC-Lib) består av elementer fra følgende navnerom:

- Dublin Core Metadata Element Set, versjon 1.1 [http://purl.org/dc/elements/1.1/]
- Dublin Core Qualifiers [http://purl.org/dc/terms/]
- Dublin Core Type Vocabulary [http://dublincore.org/usage/terms/dcmitype/]
- Dublin Core encoding schemes [http://dublincore.org/usage/terms/dc/current-schemes/]
- MODS (Metadata Object Description Schema) [http://www.loc.gov/mods/]

I tillegg til de 15 metadataelementene som er standard i Dublin Core (*tabell 2-2*), inneholder applikasjonsprofilen dessuten tre tilleggselementer:

- Audience
- Edition (hentet fra MODS)

- Location (hentet fra MODS)

Disse elementene er tatt med for å kunne beskrive flere egenskaper ved informasjonsobjekter i bibliotek enn det som er tilfellet med normal Dublin Core. Bortsett fra disse tre elementene, ser en metadatabeskrivelse utformet i henhold til applikasjonsprofilen i praksis ut som en hvilken som helst Dublin Core-beskrivelse. Det eksisterer imidlertid en viktig forskjell: Alle metadatabeskrivelser som er laget i henhold til retningslinjene i applikasjonsprofilen er laget ut ifra en felles ramme for innhold og oppbygning. Dette er ikke tilfelle for generelle Dublin Core-beskrivelser.

Standardiseringsprosessen av Dublin Core, hvor ANSI/NISO (American National Standards Institute) har pekt på utilstrekkeligheten ved Dublin Core-elementer uten retningslinjer, er en av hovedårsakene bak Dublin Core Library Application Profile. NISO hevder det eksisterer et behov for en erklæring av obligatoriske og anbefalte elementer, og derfor er applikasjonsprofilen DC-Lib opprettet for å fortelle systemer hva som forventes av en DC-beskrivelse i bibliotekssammenheng. Selv om applikasjonsprofilen DC-Lib er et viktig skritt på veien i å etablere en helhetlig Dublin Core-basert metadataprofil for bruk i biblioteker, gjenstår det ifølge *[GUE02]* fremdeles utfordringer som må løses. Opphavsperson Rebecca Guenther peker spesielt på at det i dag ikke eksisterer muligheter for raffinering av metadataelementer.

Til tross for at DC-Lib fremdeles er i utviklingsstadiet, eksisterer det flere implementasjoner basert på denne applikasjonsprofilen. Rebecca Guenther *[GUE02]* nevner prosjektene *The European Library* [TEL], *ScreenSound Australia* og *Virtual library of Anglo-American culture* som prosjekter hvor Dublin Core Library Application Profile er anvendt.

2.2.3 MARC (Machine-Readable Cataloging)

MARC definerer et dataformat som gjør det mulig for datamaskiner å utveksle, bruke og tolke bibliografisk informasjon som vanligvis er beskrevet i henhold til AACR2 (Anglo American Cataloging Rules 2). Selv om MARC oppstod allerede på 1960-tallet, utgjør dataelementene grunnlaget for de fleste bibliotekskataloger som brukes i dag. I forbindelse med en "harmonisering" av ulike MARC-dialekter, ble navnet MARC endret til MARC21 i 1997.

I praksis definerer MARC en filstruktur hvor hver post kan ha et udefinert antall felter og størrelse på feltinnholdet. På toppnivå er MARC inndelt i ti feltkategorier.

Tabell 2-3. *MARC-elementer på toppnivå*

Elementkategori	Beskrivelse
0XX	Informasjonskoder, dvs. kontrollinformasjon, tall og andre koder
1XX	Hovedbeskrivelse, dvs. navn, standardtittel, etc.
2XX	Titler, utgave, publiseringsinformasjon
3XX	Fysisk beskrivelse, etc.
4XX	Seriebeskrivelser
5XX	Bemerkninger
6XX	Emneinnførsler
7XX	Bi-innførsler, dvs. andre opplysninger enn emneord og serie
8XX	Serieinnførsler, elektronisk lokalisering og tilgang
9XX	Åpen for lokal bruk

Et enkelt eksempel viser hvordan en metadatabeskrivelse med feltene forfatter (100), tittel (245) og utgiver (260) uttrykkes i MARC21.

```
100 $a Gravir, Ketil
245 $a Norsk Jernverk - Gøthe-utvalget foreslår nedleggelse. EKKO
260 $b NRK
```

MARCXML er en variant av MARC hvor innholdet i de bibliografiske postene er uttrykt ved hjelp av xml. Konverteringen mellom standard MARC21 og MARCXML foregår uten tap av semantisk informasjon, det vil si MARCXML-data kan konverteres tilbake til den originale MARC21-formen uten at noe informasjon har gått tapt. MARC21-eksempelet kan også uttrykkes som MARCXML.

```
<record>
     <datafield tag="100">
          <subfield code="a">Gravir, Ketil</subfield>
     </datafield>
     <datafield tag=”245”>
          <subfield code=”a”>
               Norsk Jernverk - Gøthe-utvalget foreslår nedleggelse
          </subfield>
     </datafield>
     <datafield tag=”260”>
          <subfield code=”b”>NRK</subfield>
     </datafield>
</record>
```

MARC-poster med xml-struktur muliggjør enklere presentasjon og konvertering av dataene enn standard MARC21.

2.2.4 MODS (Metadata Object Description Schema)

Metadataformatet MODS er basert på elementer og semantikk fra MARC, og er i likhet med MARC utviklet av Library of Congress. MODS kan betraktes som en forenklet MARC-dialekt hvor et subsett av elementene benyttes, og hvor de tallbaserte taggene er erstattet av språkbaserte tagger. På grunn av felles semantikk med MARC, er MODS ifølge *[GUE03b]* spesielt egnet for digitale biblioteksobjekter som krever rike metadatabeskrivelser som er kompatible med eksisterende beskrivelser i bibliotekskataloger. Alle metadatabeskrivelser i MODS uttrykkes ved hjelp av xml-syntaks.

Ifølge *[GUE03a]* skal MODS utfylle andre metadataformater ved å være et alternativ mellom et enkelt metadataformat med et minimum antall felt og liten eller ingen substruktur, slik som Dublin Core, og et veldig detaljert format med mange dataelementer og en mer kompleks struktur, slik som MARC21. MODS ble med andre ord utviklet for å tilfredsstille behovet for en forkortet xml-versjon av MARC21. Ifølge *[GUE03b]* anses MARC21 som stort og komplekst, med for mange felt, i tillegg til at tallbaserte tagger både er vanskelige å lære og ikke spesielt brukervennlige. Med bakgrunn i ønsket om et mindre og mer letthåndterlig sett av dataelementer, ble første versjon av MODS offisielt gjort tilgjengelig i juni 2002. I januar 2003 lanserte Library of Congress MODS v2.0, som er beskrevet i denne oppgaven. Og i desember 2003 ble foreløpig siste versjon, MODS v3.0, lansert.

MODS har en hierarkisk oppbygning, og inneholder 19 termer/elementer på toppnivå. Mange av disse elementene fungerer kun som en innpakning av subelementer og attributter, hvor alle dataene uttrykkes. Alle detaljene er beskrevet hos *[MODS]*, som inneholder en illustrert oversikt over alle elementer, subelementer og attributter.

Tabell 2-4. *Metadataelementer på toppnivå i MODS*

Metadataelement	**Definisjon**
titleInfo (obligatorisk)	titleInfo er et emballasjeelement (wrapper element) som inneholder alle subelementer relatert til tittelinformasjon
name	name er et emballasjelement for alle navn som er assosiert med ressursen. Tilsvarer MARC21-feltene 1xx og 7xx eller Creator og Contributor i Dublin Core
typeOfResource	typeOfResource beskriver ressurstype på et generelt nivå, for eksempel tekst, lyd eller bilde
genre	genre beskriver mer spesifikt hvilken kategori ressursen tilhører, for eksempel radioprogram

Tabell 2-4. *Metadataelementer på toppnivå i MODS (forts.)*

Metadataelement	Definisjon
originInfo	originInfo er et emballasjeelement som inneholder alle subelementene relatert til publisering eller opprinnelsesinformasjon, for eksempel utgiver og publiseringssted
language	language beskriver språket som er benyttet i ressursen
physicalDescription	physicalDescription er et emballasjeelement som inneholder alle subelementene relatert fysiske egenskaper ved ressursen, for eksempel MIME-type, varighet (extent), etc.
abstract	abstract inneholder et sammendrag av ressursen, og er omtrent ekvivalent med MARC21-feltet 520 eller Dublin Core Description
tableOfContents	tableOfContents inneholder en beskrivelse av innholdsfortegnelsen for en ressurs
targetAudience	targetAudience beskriver det intellektuelle nivået hos mottakerne som ressursen er myntet på
note	note inneholder enhver bemerkning relatert til ressursen, og tilsvarer omtrentlig MARC21-feltene 5xx
subject	subject er et emballasjeelement som binder sammen subelementer som beskriver ressursen ved hjelp av emneord. Eksempler på subelementer er topic, geographic og temporal
classification	classification inneholder klassifiseringsnummeret for ressursen, for eksempel lcc (library of congress classification) eller ddc (dewey decimal code)
relatedItem	relatedItem inneholder informasjon som identifiserer en relatert ressurs
identifier	identifier inneholder en identifikator, for eksempel URI, som unikt identifiserer ressursen
location	location identifiserer institusjonen eller datalager hvor ressursen oppbevares, eventuelt en lokasjon i form av en URL
accessCondition	accessCondition inneholder informasjon om restriksjoner når det gjelder tilgangen til en ressurs, for eksempel kopirettigheter
extension	extension brukes for utvidelse av MODS gjennom å formidle tilleggsinformasjon som ikke dekkes av MODS
recordInfo	recordInfo er et emballasjeelement som inneholder administrative opplysninger om selve metadatabeskrivelsen

Egenskaper

De viktigste egenskapene ved MODS er ifølge *[GUE03b]*:

- Kompatibilitet med MARC: De fleste MODS-elementer har tilsvarende elementer i MARC 21-formatet, og MODS xml schema spesifiserer hvor semantikken for hvert element kan lokaliseres i MARC21.
- Integrasjon av MARC-elementer: I enkelte tilfeller er flere MARC-elementer slått sammen til ett MODS-element.
- Ingen krav til katalogiseringsregler, men AACR2 (Anglo American Cataloging Rules) anbefales.
- Ulike typer relasjoner (MARC 76X-78X) uttrykkes i MODS v.h.a. elementet *relatedItem*, samt et typeattributt som angir type relasjon.
- Gjenbruk av definisjoner: Enkelte MODS-elementer definerer konsepter som gjentar seg som subelementer i flere enn et element. Ettersom den underliggende enkodingen er XML, er disse definert som XML complex types, som tillater bruk av den samme definisjonen i flere elementer.

Som nevnt har MODS høy grad av kompatibilitet med MARC-beskrivelser siden det arver semantikken til tilsvarende dataelementer i MARC21-formatet. MODS framstår derfor som rikere enn Dublin Core, men samtidig enklere enn fullstendig MARC-format.

En viktig hensikt med MODS er at formatet skal komplementere andre metadataformater og at det skal være et alternativ mellom et veldig enkelt metadataformat med et minimum antall felt og liten eller ingen substruktur (for eksempel Dublin Core) og et veldig detaljert format med mange dataelementer med forskjellige strukturelle kompleksisteter (for eksempel MARC21).

Eksempel på metadatabeskrivelse i MODS:

```
<mods>
     <titleInfo>
          <title>
          Norsk Jernverk - Gøthe-utvalget foreslår nedleggelse. EKKO
          </title>
     </titleInfo>
     <name>
          <namePart>Gravir, Ketil</namePart>
          <role><text>programleder</text></role>
     </name>
     <typeOfResource>sound recording</typeOfResource>
     <genre>Radioprogram</genre>
     <originInfo>
          <publisher>NRK</publisher>
          <dateCreated>1988-04-05</dateCreated>
     </originInfo>
     <physicalDescription>
          <extent>00:09:26</extent>
     </physicalDescription>
```

```
        <abstract>
        Gøthe-utvalget (Statens forhandlingsutvalg) foreslår å legge
        ned stålproduksjonen ved Norsk Jernverk i Mo i Rana. Odd GØTHE
        (mv) kommenterer kritikken til utvalgets innstilling. Om de
        økonomiske sidene. Den største utfordring industri-Norge har
        stått overfor etter krigen. Om mulighetene for annen industri
        i Mo, med lønnsom produksjon, f.eks. et aluminiumsverk med pri-
        vat finansiering.
        </abstract>
        <identifier type="uri">
        URN:NBN:no-nb_drl_20427 1787
        </identifier>
        <accessCondition>NRK</accessCondition>
</mods>
```

Bruksområder

Behovet for MODS er uttrykt av medlemmer innen digitale bibliotek og relaterte brukerfellesskap i forbindelse med prosjekter som involverer søk og gjenfinning, forvaltning av komplekse digitale objekter, samt integrering av metadata fra biblioteksdatabaser med andre ikke-MARC kilder.

OAI-PMH høster MARC-poster fra mange systemer og gjør dem vidt tilgjengelige. Vanligvis har disse postene enten vært i MARC (MARC tagging og MARCXML syntaks) eller enkel Dublin Core. Library of Congress har i prosjektet American Memory tatt i bruk MODS som et alternativt format for sine 100,000 metadataposter over digitaliserte elementer, blant annet bøker, kart, fotografier, tidlige filmer, notemusikk og trykt ”døgnlitteratur”. Ved å bruke MODS får man mulighet til å eksportere rikere metadata enn i en Dublin Core-post, som utelater mye av metadataene til fordel for enkelheten.

MODS kan dessuten benyttes direkte til å lage originale ressursbeskrivelser som er kompatible med eksisterende biblioteksdata og uttrykt i xml-syntaks.

Relasjon til MARC og MARCXML

Siden MODS-skjemaet er et subsett av MARC, takler det ikke ”round-tripability” med MARC21. En original MARC21-post som konverteres til MODS og deretter tilbake til MARC21 kan med andre ord miste noe av taggingen og/eller dataene. Rekonvertering kan dessuten føre til at data blir plassert i et annet felt enn der de opprinnelig befant seg.

Dersom man ønsker tapsfri konvertering av MARC til xml-format, anbefales det i *[GUE03b]* å benytte MARCXML. Konvertering fra MARC til MODS bør foregå på følgende måte: ”For any conversion between MARC in ISO2709 format and MODS, it is expected that the record would first og through a conversion to MARCXML before a transformation to the subset that is MODS.” Library of Congress tilbyr verktøy for en slik konvertering.

2.2.5 Sammenligning Dublin Core – MODS

Dublin Core, uttrykt i henhold til retningslinjene i applikasjonsprofilen for bibliotek, og MODS framstår som de mest aktuelle metadataformatene å jobbe videre med i dette prosjektet. For å kunne avgjøre hvilket av de to formatene som er best egnet som felles metadataformat, kreves det en sammenligning av formatenes egenskaper.

En slik sammenligning vil imidlertid bli litt forskjellig avhengig om man sammenligner MODS med enkel eller kvalifisert Dublin Core. I *[GUE03b]* sammenlignes MODS og kvalifisert Dublin Core Metadata Element Set (DCMES) på et overordnet nivå.

Tabell 2-5. *Sammenligning av Dublin Core Metadata Element Set og MODS*

Dublin Core (kvalifisert)	**MODS**
16 elementer[a] på toppnivå	19 elementer på toppnivå
28 subelementer[b]	47 subelementer
Ingen substruktur	Hierarkisk substruktur

a. inkluderer ”audience” som ble lagt til i oktober 2001
b. i form av kvalifikatorer

Både Dublin Core og MODS uttrykkes ved hjelp av xml, men mens Dublin Core har en ”flat” oppbygning (alle metadataopplysningene på ett nivå), er MODS hierarkisk oppbygd (metadataopplysninger på flere nivåer). Dette gjør det mulig å uttrykke sammenhenger mellom metadataopplysninger som er relatert til hverandre, for eksempel navn og rolle.

I enkelte tilfeller, for eksempel tittel, er DC-elementet veldig likt tilsvarende MODS-element, men semantikken for flere av DC-elementene er likevel bredere enn for tilsvarende MODS-elementer, siden MODS har arvet semantikken fra MARC. MODS kan med andre ord romme svært omfattende metadatabeskrivelser, og det er legitimt å stille spørsmål om MODS blir for detaljert for å være et ”minste felles multiplum”-format.

Tabell 2-6. *Sammenligning av elementene i MODS og Dublin Core. Hentet fra [GUE03b].*

MODS	Dublin Core
titleInfo [obligatorisk]	Title
name	Creator
	Contributor
typeOfResource	Type
genre	
publicationInfo	Publisher
	Date
physicalDescription	Format
language	Language
abstract	Description
tableOfContents	Description
note	Description
subject	Subject
classification	Subject
relatedItem	Relation
identifier	Identifier
targetAudience	Audience
cartographics	Coverage
accessConditions	Rights

MODS-elementene *extension*, *recordInfo* og *location* har ingen tilsvarende elementer i Dublin Core. I tillegg til å sammenligne element for element, fokuseres det i *[GUE03b]* på noen få elementer som illustrerer de største forskjellene mellom Dublin Core og MODS.

Creator og Contributor

DC-elementene Creator og Contributor er beregnet for navn assosiert med en ressurs, og skillet mellom de to elementene er basert på om bidraget til ressursen er primært eller sekundært. MODS inneholder kun ett element, ”name”, man kan i tillegg oppgi subelementet ”role” for å spesifisere hvilken rolle et navn spiller i forhold til den beskrevne ressursen. MODS har i tillegg et typeattributt som gjør det mulig å skille mellom ulike typer navn (person, organisasjon, konferanse). Dette

skillet er ikke mulig å beskrive i Dublin Core, siden elementene creator og contributor ikke har assosierte kvalifikatorer.

Publisher

DC-elementet publisher har i likhet med creator og contributor ingen definert substruktur eller kvalifikatorer, som gjør det mulig å ytterligere spesifisere publisher. Problemer med denne løsningen er at publisher-elementet ikke kan assosieres med publikasjonssted eller –dato. Dessuten kan det eksistere flere forskjellige publiserere og publiseringsdatoer, ettersom alle DC-elementer kan gjentas.

MODS benytter elementet ”publicationInfo” med subelementer for sted, publisererer og ulike typer datoer. Elementet publicationInfo er repeterbart, og kan derfor benyttes for mange publiserere med passende informasjon for sted og publiseringsdato.

Date

Enkel Dublin Core inneholder det generelle datoelementet date, som skal dekke alle typer datoer i livssyklusen til en ressurs. Avansert Dublin Core inneholder kvalifikatorer som gjør det mulig å skille mellom ulike typer datoer, men datoer kan ikke assosieres med andre elementer (for eksempel publisher). I MODS beskrives datoer ved hjelp subelementer under publicationInfo-elementet. Det eksisterer forskjellige subelementer for forskjellige typer datoer (dateIssued, dateCreated, dateCaptured).

Relation

DC-elementet Relation tilsvarer omtrent MODS-elementet relatedItem, men siden DC ikke har noen form for substruktur, kan metadataene om en ressurs kun uttrykkes ved hjelp av fritekst. Dette gjør det umulig for applikasjoner å parse elementene, og gjenfinning kan derfor bare skje på bakgrunn av nøkkelord. Dessuten kan en DC relation *enten* bare peke på relatert element *eller* beskrive det; begge deler samtidig går ikke.

For å muliggjøre en mer presis gjenfinning og identifisering, definerer MODS-formatet flere sublementer for relatedItem, blant annet name, title, physicalDescription, identifier og note.

Andre typer metadata

MODS-elementet recordInfo er laget for å inneholde data om selve metadataposten, blant annet for å kunne forvalte selve metadataene. Dublin Core inneholder ingen elementer for denne typen administrative metadata. MODS har dessuten et extension-element for lokale felt eller for andre skjemaer.

Oppsummering

Oppsummering av sammenligningen og valg av metadataformat er beskrevet i *kap. 4.2.3, s. 52.*

2.3 Interoperabilitet

Begrepet interoperabilitet kan i informatikksammenheng oversettes til samspillevne på norsk. Mer konkret betyr dette evnen til å få forskjellige systemer og komponenter med forskjellig teknisk oppbygning, og som administreres av forskjellige organisasjoner, til å virke sammen. Begrepet interoperabilitet har sin opprinnelse i databaseverdenen, hvor det brukes for å betegne teknikker og metoder for å integrere datainnholdet fra flere forskjellige databasesystemer. Etter hvert har imidlertid begrepet blitt tatt i bruk innenfor flere andre områder, blant annet digitale bibliotek.

Det overordnede målet med interoperabilitet er å lage koherente tjenester for brukerne. Ifølge *[ARW02]* krever dette samarbeidsavtaler på tre nivåer: Teknisk, innholdsmessig og organisasjonsmessig.

- Teknisk: Formater, protokoller, sikkerhetssystemer, etc., slik at meldinger kan utveksles.
- Innhold: Data og metadata, inkludert semantiske avtaler for hvordan informasjonen skal tolkes.
- Organisasjon: Basisreglene for tilgang, bevaring av samlinger og tjenester, betaling, autentisering, etc.

2.3.1 Ulike perspektiver på interoperabilitet

I den tradisjonelle oppfatningen av begrepet interoperabilitet, slik det også er beskrevet ovenfor, er ideen å få til samvirke mellom forskjellige organisasjoners systemer og komponenter. Når det er snakk om store organisasjoner med mange forskjellige systemer og komponenter, kan imidlertid prinsippene for interoperabilitet også benyttes internt i organisasjonen.

I dette prosjektet fokuseres det på interoperabilitetsspørsmålet internt i en organisasjon som forvalter store informasjonsmengder; Nasjonalbiblioteket. Konsekvensen av dette fokuset blir at organisasjonsmessige samarbeidsavtaler spiller en mindre rolle enn hva de ville gjort dersom målet var å oppnå interoperabilitet mellom forskjellige organisasjoner. Samtidig blir det viktigst å fokusere på samarbeidsavtaler som gjelder tekniske og innholdsmessige forhold.

Interoperabilitet kan med andre ord betraktes både i et globalt og et lokalt perspektiv

- Globalt: Mellom organisasjoner.
- Lokalt: Internt i en organisasjon.

Felles for begge perspektivene er ideen om å få komponenter med forskjellig teknisk oppbygning til å virke sammen. Forskjellen mellom de to perspektivene kan sammenlignes med forskjellen mellom internett og intranett.

2.4 Metoder for interoperabilitet mellom samlinger

Det eksisterer flere ulike tilnærmingsmåter til interoperabilitetsspørsmålet. *[ARW02]* beskriver tre alternative metoder for å oppnå interoperabilitet mellom samlinger i digitale bibliotek:

- Sammenslutning (federation)
- Metadatahøsting (harvesting)
- Innsamling (gathering)

Distribuert søking (metasøking) kan dessuten betraktes som en metode for interoperabilitet, hvis det underliggende motivet med å oppnå interoperabilitet er søking.

2.4.1 Sammenslutning (federation)

Sammenslutning er den tradisjonelle tilnærmingsmåten for å oppnå interoperabilitet. Denne metoden går ut på at en gruppe organisasjoner blir enige om at tjenestene deres samsvarer med felles spesifikasjoner. Bibliotek som deler katalogposter i MARC-formatet online ved hjelp av informasjonsgjenfinningsprotokollen Z39.50 er et eksempel på en sammenslutning. Denne metoden utgjør den sterkeste formen for interoperabilitet, men krever samtidig mest av deltakerne. Ifølge *[ARW02]* blir konsekvensen at "federated services decline in responsiveness and reliability as the number of independent servers grow."

2.4.2 Metadatahøsting (harvesting)

På grunn av at det er ressurskrevende å lage store sammenslutninger, er høsting lansert som et enklere alternativ for å oppnå interoperabilitet mellom flere aktører. Høsting går ut på at hvert digitale bibliotek gjør metadata fra samlingene sine tilgjengelig i et felles utvekslingsformat, og at disse metadataene deretter kan høstes av tjenestetilbydere. De innhøstede metadataene kan bygges inn i tjenester som for eksempel informasjonssøking. Open Archives Initative Protocol for Metadata Harvesting (OAI-PMH, se *kap. 2.6, s. 25*), som anvendes i dette prosjektet, er basert på høsting som interoperabilitetsmetode.

Denne interoperabilitetsmetoden er utdypet i kap. 2.5.

2.4.3 Innsamling (gathering)

Den svakeste interoperabilitetsmetoden, innsamling, går ut på å samle inn åpent tilgjengelig informasjon ved hjelp av en web-crawler, på samme måte som en websøkemotor gjør. Innsamling stiller ingen krav til samarbeid, standarder eller tjenester hos dem som dataene blir samlet inn fra, og gir derfor også en begrenset form for interoperabilitet.

2.4.4 Distribuert søking

Distribuert søking (metasøking) kan betraktes som et alternativ til metadatahøsting når interoperabilitetsformålet er å bygge et enhetlig søkesystem mot mange samlinger. Tilnærmingen til problemstillingen er imidlertid relativt forskjellig fra metadatahøsting, hvor metadataene fra alle relevante kilder høstes inn og indekseres på ett sted, og hvor brukerene deretter kan sende spørringer til denne sentrale indeksen fra et enhetlig søkegrensesnitt.

I distribuert søking framstår også søkingen som enhetlig for brukeren, men hovedprinsippet er at selve søkeprosessen skal utføres lokalt hos alle de relevante datakildene. Distribuert søking kan beskrives som en trinnvis prosess med fire hovedtrinn:

1. Alle spørringer kringkastes i sanntid til mange distribuerte datakilder
2. Søkeprosessen utføres lokalt hos hver av disse kildene
3. Hver kilde returnerer sine søkeresultater til kringkasteren (felles søkegrensesnitt)
4. Alle søkeresultatene flettes sammen og vises som en enhetlig resultatliste for brukeren

I *[VEE03]* er distribuert søking sammenlignet med søking i en sentral indeks som inneholder innhøstede metadata. Som det går fram av tabellen, har begge metodene både sterke og svake sider, og det er ikke mulig å si at den ene metoden er bedre egnet enn den andre.

Tabell 2-7. *Sammenligning av sentral indeksering og distribuert søking*

Sentral indeksering	**Distribuert søking**
Søkemulighetene gjelder indeksen som helhet	Søkemulighetene kan variere mellom de forskjellige søkeindeksene
Forutsigbar ytelse	Ytelse avgjøres av tregeste søkeindeks
Lett integrasjon av resultater	Vanskeligere integrasjon av resultater
Sentralt kontrollert indeks	Ingen kontroll over distribuerte indekser
Mindre systembelasting (ett søk, én henvendelse)	Større fordeling av systembelastning
Flere ressurser kreves ved indeksering	Færre ressurser kreves ved indeksering

Search and Retrieve Web Service *[SRW]* er en protokoll for distribuert søking som kombinerer grunnkonseptene i Z39.50 med webteknologi. SRW anvendes blant annet i The European Library *[TEL]*, og beskrives i *[VEE03]* som et alternativ til OAI-PMH.

2.5 Metadatahøsting

Ideen om metadatahøsting som interoperabilitetsmetode oppstod tidlig på 1990-tallet i prosjektet Harvest *[BOW94]*, men fikk liten utbredelse da den ble lansert. Konseptet om å høste inn metadata og lagre disse i en sentral indeks ble imidlertid ikke glemt, og i 1998/1999 ble metadatahøsting igjen introdusert i prosjektet Universal Preprint Server. Siden 2000 har OAI-PMH (se kap. 2.6) vært den dominerende teknologien på området.

I dette prosjektet er en av forutsetningene at metadatahøsting ved hjelp av OAI-PMH skal benyttes som metode for å oppnå interoperabilitet. Det er derfor interessant å undersøke metadatahøsting som interoperabilitetsmetode nærmere, med fokus på:

- Overganger til felles metadataformat
- Prinsipp
- Potensielle bruksområder

2.5.1 Overganger til felles metadataformat

En forutsetning for å kunne gjennomføre metadatahøsting, er at metadata fra de digitale samlingene som man ønsker å inkludere i en samlet tjeneste gjøres tilgjengelig i et felles utvekslingsformat. Dette metadataformatet må være av en så generell karakter at det kan romme metadatabeskrivelser for forskjellige typer informasjonsobjekter, hvor forskjellige metadataegenskaper er vektlagt ulikt.

På grunn av at de forskjellige datakildene/samlingene har ulike interne strukturer, kreves det individuelle overganger for hver enkelt datakilde fra deres interne struktur til det felles metadataformatet. Som nevnt i *kap. 2.2, s. 6*, består et metadataformat blant annet av et vokabular av termer som har en betydning (semantikk). Og ifølge *[WOO00]* er hensikten med en metadataovergang (crosswalk) å etablere en semantisk avbildning (mapping) som bringer samsvar mellom de ulike informasjonsfeltene i databasene og tilsvarende termer i fellesformatet som uttrykker det samme eller lignende meningsinnholdet.

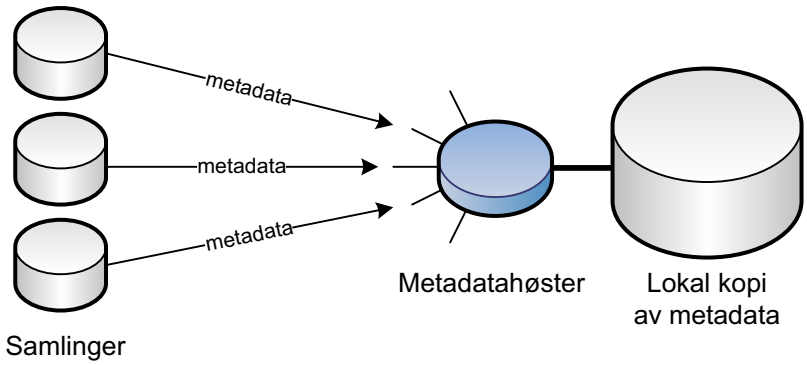
Blant potensielle problemstillinger som kan oppstå i forbindelse med semantisk avbildning nevner *[WOO00]*:

- Det eksisterer ingen 1:1 overgang mellom original databasestruktur og fellesformatet
- Det eksisterer ingen korresponderende felt i fellesformatet for enkelte typer informasjon
- Det er bare delvis samsvar mellom original struktur og fellesformatet
- En korrekt avbildning av strukturen garanterer ikke at meningsinnholdet blir korrekt avbildet

Hvis slike problemstillinger oppstår, resulterer det som oftest i en forringelse av metadatakvaliteten i fellesformatet sammenliknet med de originale metadataene.

2.5.2 Prinsipp

En betingelse for at metadatahøsting skal kunne finne sted, er at hver samling har en mekanisme som gjør det mulig for en høster å kommunisere med samlingen. Selve høstingen gjennomføres ved at metadatahøsteren henvender seg direkte til hver enkelt samling og innhenter en kopi av de metadataene som er tilgjengelige i utvekslingsformatet. Alle de innhøstede metadataene blir lagret på ett sentralt sted.



Figur 2-1. *Metadatahøsting*

Målet er å bruke disse innhøstede metadataene til å bygge forskjellige typer tjenester som øker metadataenes verdi for brukerne.

2.5.3 Potensielle bruksområder

Hvilke tjenester kan så metadatahøsting benyttes til? Dette er avhengig av flere faktorer, blant annet hva slags felles metadataformat som er benyttet og hvilken kvalitet de innhøstede metadataene har. I *[RMHP]* skisseres flere potensielle bruksområder:

- En "superkatalog" hvor alle typer materialer gjøres tilgjengelig for medlemmer av et konsortium
- Enhetlig tilgang til store mengder digtialiserte materialer innenfor bestemte emneområder
- En enhetlig måte for en institusjon for å dele sine egne metadata slik at den ikke behøver å utvikle spesielle løsninger for å støtte individuelle henvendelser
- En måte å gjøre innholdet i bibliotekers MARC-kataloger tilgjengelig for mer generelle søkemotorer
- En måte for å forbedre et biblioteks katalog ved å legge til lenker til digitale reproduksjoner av verk
- En "universell katalog" som tilbyr tilgang til biblioteksressurser, for eksempel bøker, tidsskrifter og andre materialer, gjennom ett enhetlig grensesnitt.

Ofte realiseres flere av disse bruksområdene samtidig, som en indirekte konsekvens av at en organisasjon har valgt å ta i bruk metadatahøsting. Hovedmålsetningen med dette prosjektet er å utvikle et enhetlig søkegrensesnitt for metadata fra heterogene samlinger - à la en "universell katalog" - slik at brukere kan få enklere tilgang til informasjon. Samtidig oppnår Nasjonalbiblioteket også en enhetlig måte for å dele metadata fra sine samlinger, hvis dette er av interesse.

2.6 Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH)

Open Archives Initative Protocol for Metadata Harvesting (heretter forkortet OAI-PMH) har siden lanseringen i 2001 etablert seg som den rådende standarden for metadatahøsting. Modellen tar utgangspunkt i at det eksisterer to typer aktører; datatilbydere og tjenestetilbydere. Og protokollen definerer en mekanisme for hvordan tjenestetilbydere kan høste poster som inneholder metadata fra datatilbydere.

Det er i denne sammenhengen viktig å understreke at OAI-PMH kun definerer mekanismen for å samle metadata på ett sted. Egenskaper knyttet til selve metadataene, som for eksempel hvilke lenker som eksisterer mellom metadataene og ressursene, eller hvilke tjenester de innhøstede metadataene benyttes til, er med andre ord utenfor protokollens definisjonsområde.

Ettersom OAI-PMH utgjør en av forutsetningene i dette prosjektet, beskrives protokollen i nærmere detalj i dette kapittelet. Fokus rettes mot:

- Bakgrunn for protokollen
- Hva OAI-PMH ikke er
- Definisjoner og konsepter
- Teknisk oppbygging og virkemåte
- OAI-PMH og metadata
- Framtidsplaner

For å unngå tvetydigheter, er den tekniske beskrivelsen av protokollen i hovedsak basert på *[OAI-PMH]*.

2.6.1 Bakgrunn

Organisasjonen som står bak OAI-PMH, Open Archives Initiative *[OAI]*, har som hovedmål å utvikle og fremme interoperabilitetsstandarder, slik at ressurser kan spres på en mer effektiv måte. Organisasjon ble dannet i 1999 etter et møte i det amerikanske brukerfellesskapet for e-printing. Alle detaljene omkring dette møtet, kjent som møtet i Santa Fe, er beskrevet i *[VAN00a]*.

Bakgrunnen for Open Archives Initiative, og den opprinnelige hensikten bak OAI-PMH, var ifølge *[LAG01]* å etablere et effektivt elektronisk alternativ til den tradisjonelle måten å publisere vitenskapelig materiale på. Enkeltstående e-print-arkiver[3] var allerede etablert innen forskjellige fagområder i løpet av 1990-tallet, og målet med OAI var å etablere et veldefinert, enkelt rammeverk for å knytte disse arkivene sammen gjennom internett. På denne måten håpet man å oppnå mer enhetlige søketjenester og en hurtigere spredning av vitenskapelige forskningsresultater.

For å oppnå størst mulig oppslutning, ble det vedtatt å etablere en enkel og lite ressurskrevende løsning basert på metadatahøsting. Den tekniske løsningen skulle ta utgangspunkt i den bredere

3. En *e-print* defineres i denne sammenhengen som et dokument som forfatter selv har arkivert i et datalager/arkiv.

Dienst-protokollen *[LAG95]*, i tillegg til erfaringene som ble gjort i forbindelse med prototypen og forgjengeren til OAI-PMH, Universal Preprint Service *[VAN00b]*.

Interessen for OAI-løsningen ble så stor at Open Archives Initiative i løpet av det første året ble utvidet til også å omfatte forleggere, bibliotek og museer. På grunn av denne økende interessen fra forskjellige typer organisasjoner, har fokusområdet gradvis blitt generalisert fra e-print via dokumentlignende objekter til ressurser underveis i utviklingsprosessen.

OAI-PMH versjon 1.0, ble lansert i januar 2001. Denne versjonen av protokollen befant seg i en eksperimentell fase i halvannet år, før OAI-PMH versjon 2.0 ble lansert i juni 2002. Dette er den foreløpig nyeste og første stabile versjonen av protokollen.

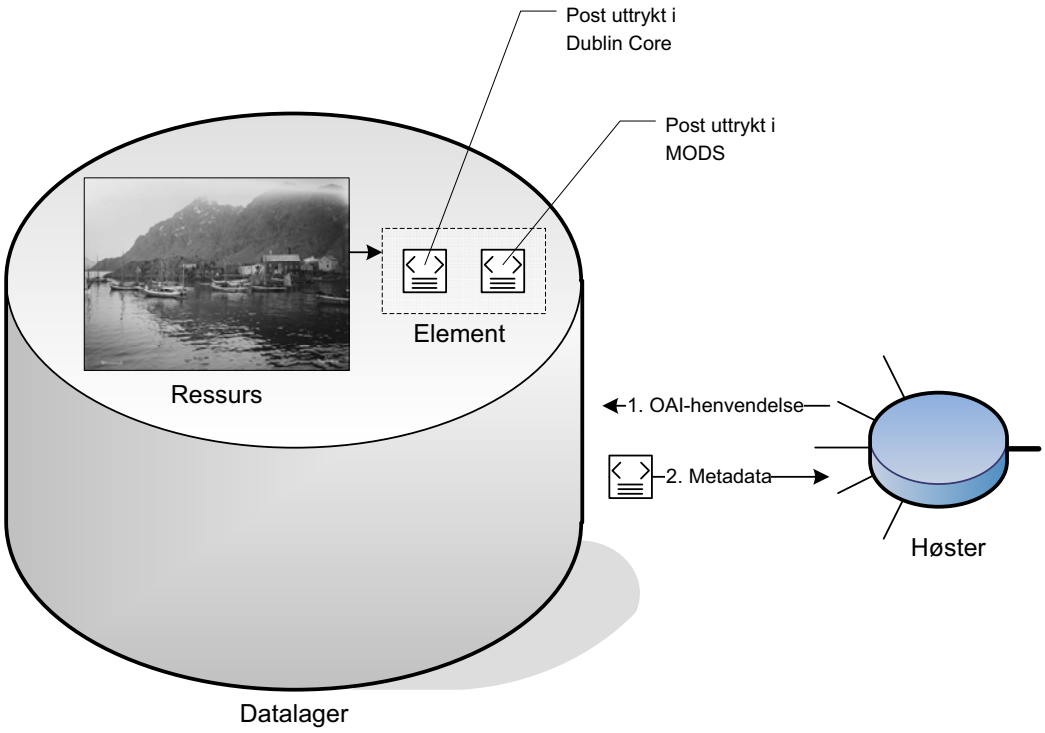
2.6.2 Hva OAI-PMH ikke er

Før OAI-PMH beskrives i nærmere detalj, kan det være nyttig å informere kort om protokollens begrensninger. I *[WAR01]* klargjøres noen vanlige misoppfatninger knyttet til OAI-PMH:

- OAI-PMH definerer ingen mekanismer for å tilby og høste selve informasjonsobjektene (dokumenter, bilder, etc.). OAI-PMH er kun en protokoll for utveksling av metadata.
- OAI-PMH dreier seg ikke om direkte interoperabilitet mellom arkiver. I stedet er protokollen basert på et tydelig skille mellom de to aktørtypene datatilbydere og tjenestetilbydere. Dette poenget understrekes i *[OAFOT]*: "OAI-PMH is not a search protocol, but its use can underpin search-based services; it is a base layer on which to build other services."
- Selv om modellen har et tydelig skille mellom datatilbydere og tjenestetilbydere, er det ingenting i veien for at en organisasjon ikke kan fylle begge rollene.
- OAI-PMH er ikke begrenset til metadataformatet Dublin Core, selv om dette er påkrevd som minste felles multiplum. Ytterligere informasjon om dette finnes i *kap. 2.6.5, s. 31*.

2.6.3 Definisjoner og konsepter

For å kunne beskrive OAI-protokollens tekniske oppbygning og virkemåte, må noen grunnleggende begreper defineres først. Forståelsen av disse begrepene er viktig for å forstå protokollens virkemåte.



Figur 2-2. *Grunnleggende konsepter i OAI-PMH. Høster initierer og sender OAI-henvendelse til datatilbyderen, som returnerer metadatapost(er) eller informasjon om datalageret.*

Høster (harvester)

En høster er en klientapplikasjon hos tjenestetilbyderen som sender ut OAI-PMH henvendelser for å samle inn metadataposter fra datatilbydernes datalager (repository).

Datalager (repository)

Et datalager er en nettverkstilgjengelig server hos datatilbyderen som kan prosessere de seks OAI-PMH henvendelsene (*tabell 2-8*) i henhold til beskrivelsene i *[OAI-PMH]*. Et datalager brukes dermed som lagringsplass for metadataene som kan høstes av en høster. OAI-PMH skiller mellom tre forskjellige entiteter knyttet til metadataene som gjøres tilgjengelig av OAI-PMH:

- **Ressurs** (resource) - er selve informasjonsobjektet som metadataene beskriver. Informasjon om ressurser er utenfor OAI-PMH sitt definisjonsområde.
- **Element** (item) - se nedenfor.
- **Post** (record) - se nedenfor.

Element (item)

Et element i et datalager er en container som inneholder eller dynamisk genererer metadata i ett eller flere formater for en enkel ressurs. Disse metadataene kan høstes som poster via OAI-PMH. Hvert element har en identifikator som er unik innenfor det betraktede datalageret.

Unik identifikator

En unik identifikator identifiserer et element i et datalager på en utvetydig måte, slik at metadata kan hentes ut fra ønsket element ved en OAI-PMH-henvendelse. Dersom et element inneholder metadata i mange formater, vil alle postene dele den samme unike identifikatoren. Et eksempel på en identifikator er `oai:galnor.nb.no:59000`.

Post (record)

En post er metadata for en ressurs uttrykt i et spesifikt format. En post returneres som en xml-fil som respons på en protokollhenvendelse om å hente metadata fra et element. En post identifiseres utvetydig av kombinasjonen av elementets unike identifikator, et metadataprefiks som identifiserer metadataformatet i posten, samt datomerkingen av posten.

Alle metadataposter har følgende generelle xml-struktur:

```
<header>
     <!-- Unik identifikator + tidsstempel -->
</header>
<metadata>
     <!-- Metadatabeskrivelse (xml), f.eks. i Dublin Core -->
</metadata>
<about>
     <!-- Valgfri tilleggsinformasjon om metadatabeskrivelsen -->
</about>
```

Sett

Et sett er en valgfri konstruksjon for å gjøre en logisk inndeling av datalagrene. Hensikten er å gruppere elementer med tanke på selektiv høsting.

Selektiv høsting

Selektiv høsting tillater en høster å begrense høstehenvendelsene til deler av metadataene som er tilgjengelige i et datalager. Dette kan enten gjøres ved hjelp av datomerking eller settmedlemsskap.

2.6.4 Teknisk oppbygning og virkemåte

Den tekniske oppbygningen i OAI-PMH er relativt enkel, ettersom et av målene med protokollen er at den skal være enkel å ta i bruk. Som nevnt innledningsvis er rammeverket oppbygd rundt to typer aktører:

- Datatilbydere (også kalt datalagre eller åpne arkiver) administrerer systemer/samlinger, og tilbyr fri tilgang til metadata gjennom OAI-PMH.
- Tjenestetilbydere benytter OAI-grensesnittene hos datatilbyderne til å høste inn og lagre metadata. De innhøstede metadataene brukes til å bygge tjenester for brukere, for eksempel søk.

Et viktig prinsipp i OAI-PMH er at det kun er tjenestetilbydere som kan henvende seg til datatilbydere, og ikke omvendt. Det er med andre ord ikke støtte for datatilbyder-drevet interaksjon.

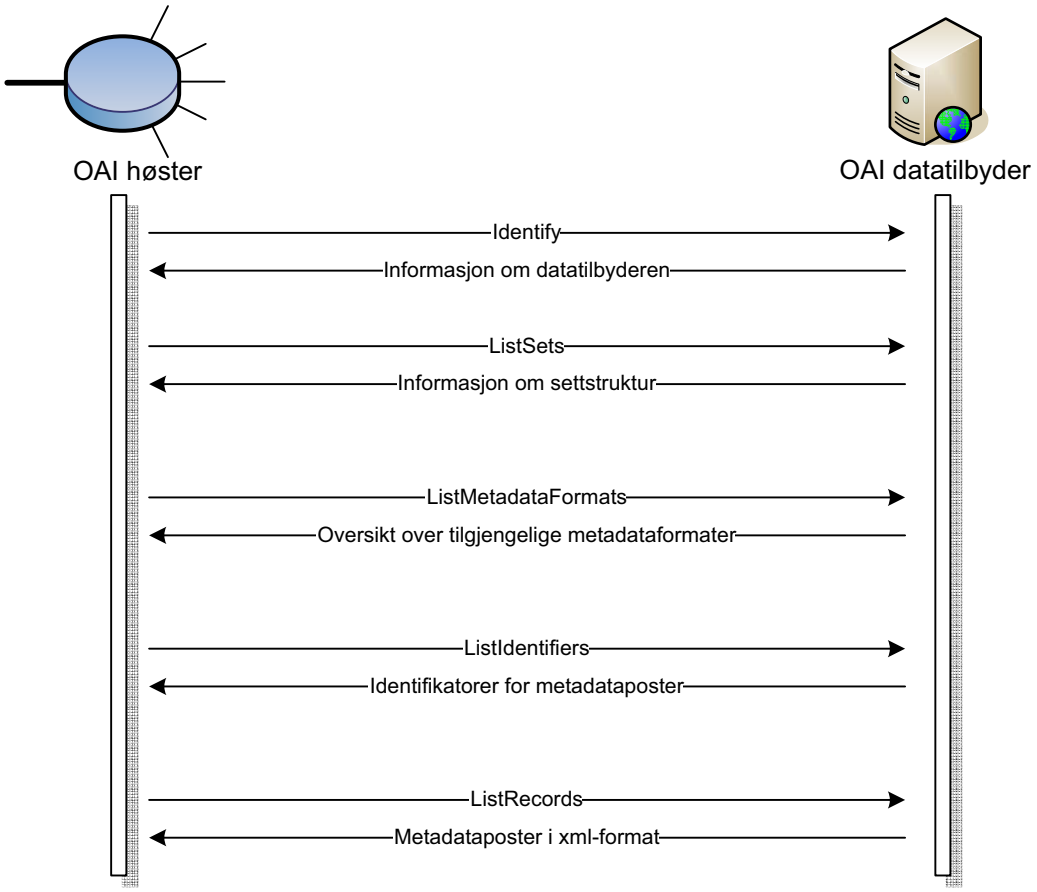
Protokollhenvendelser og –svar

OAI-PMH definerer seks forskjellige måter for en tjenestetilbyder å henvende seg til en datatilbyder. Alle disse seks henvendelsene er uttrykt i form av verb, hvorav halvparten gjelder høsting, mens den andre halvparten gjelder informasjon om datatilbyderen.

Tabell 2-8. *OAI-PMH-henvendelser og deres funksjon*

Henvendelse (verb)	Funksjon
Identify	Hente beskrivelse av datalager, dvs. implementerte standarder og protokoller
ListMetadataFormats	Hente liste over metadataformater som er tilgjengelige fra et datalager. Ved å bruke det valgfrie argumentet identifier, begrenser man henvendelsen til bare å gjelde metadataformatene for ett bestemt element.
ListSets	Hente liste over settstrukturen (sett og subsett) i datalageret. Kan være nyttig i selektiv høsting
ListIdentifiers	Høste identifikatorer for metadataposter, valgfritt om de må samsvare med et bestemt sett og/eller en tidsperiode. Responsen er i praksis en forkortet form av det ListRecords gir
GetRecord	Høste en individuell metadatapost som samsvarer med en spesifisert identifikator og i et bestemt metadataformat (angitt av metadataPrefix)
ListRecords	Høste metadataposter som samsvarer med et bestemt metadataformat (angitt av metadataPrefix), samt eventuelt et bestemt sett og/eller en bestemt tidsperiode

En tjenestetilbyder (representert ved en høster) henvender seg til vanligvis til datatilbydere etter et bestemt mønster. Figuren nedenfor illustrerer hvordan denne kommunikasjonssekvensen mellom en OAI-høster og en datatilbyder tradisjonelt foregår.



Figur 2-3. *Eksempel på kommunikasjonssesjon mellom høster og datatilbyder*

Alle kommunikasjon mellom datatilbyder og tjenestetilbyder foregår ved hjelp av HTTP (hypertext transfer protocol). Dette innebærer at en tjenestetilbyders henvendelser i OAI-PMH uttrykkes som HTTP-henvendelser med tilhørende OAI-argumenter.

```
http://oai.nb.no/OAIscript?verb=GetRecord&identifier=oai:gal-
nor.nb.no:59000&metadataPrefix=mods
```

I dette eksempelet henvender tjenestetilbyderen seg til datatilbyderens basis-URL (`http://oai.nb.no/OAIscript`), som er adressen til en HTTP-server som fungerer som et datalager. I tillegg til basis-URL, består alle henvendelser av ett eller flere argumenter på formen key=value. Dersom det er flere slike argumenter, skilles disse ved hjelp av &-tegnet. I eksempelet er det tre argumenter:

- Verb: `GetRecord`
- Identifier: `oai:galnor.nb.no:59000`
- Metadataformat: `mods`

Datatilbyderens respons på denne typen henvendelser er kodet i xml-syntaks.

Format for XML-respons

Alle svar på OAI-PMH-henvendelser må være velformulerte xml-dokumenter som samsvarer med xml-skjemaet definert i *[OAI-PMH]*.

For det nevnte eksempelet, vil dette i praksis si at xml-strukturen i respons er:

```
<?xml version=”1.0” encoding=”UTF-8” ?>
<OAI-PMH>
     <responseDate>2004-02-24T09:59:05Z</responseDate>
     <request verb="GetRecord" identifier="oai:galnor.nb.no:59000"
     metadataPrefix="mods">http://oai.nb.no/OAIscript></request>
     <GetRecord>
          <record>
               <header></header>
               <metadata>
                    <mods>
                         <!-- Metadatabeskrivelse i MODS -->
                    </mods>
               </metadata>
          </record>
     </GetRecord>
</OAI-PMH>
```

2.6.5 OAI-PMH og metadata

Open Archives Initiative har vedtatt at Dublin Core skal være det obligatoriske formatet for metadata i OAI-PMH[4]: “At a minimum, repositories must be able to return records with metadata expressed in the Dublin Core format, without any qualification.” *[OAI-PMH]* Av den grunn er metadataprefixet oai_dc reservert for ukvalifisert Dublin Core i OAI-PMH.

Selv om det er påkrevd at alle datalagre skal eksportere metadata i Dublin Core, er det ingenting i veien for at et datalager parallelt eksporterer metadata i flere formater. Som eksempel kan Library of Congress-prosjektet American Memory nevnes, hvor OAI-datalageret tilbyr metadataposter i formatene Dublin Core, MODS og MARCXML. Et element i datalagaret kan, som nevnt i *kap. 2.6.3, s. 26*, inneholde eller generere metadata i mange formater for den samme ressursen. Det eneste kravet som stilles av OAI-protokollen, er at metadatapostene må være strukturerte som xml-data i henhold til et tilhørende xml-skjema.

Motivet for å eksportere metadata parallelt i flere formater, er hovedsakelig at institusjoner ønsker å inngå i flere forskjellige tjenester hvor det stilles ulike krav til detaljnivået i metadataene. Ved

4. På den offisielle e-postlista til Open Archives Initiative har det vært diskutert om dette kravet bør frafalles. Ingen beslutninger er foreløpig fattet (mai 2004).

både å tilby relativt enkle metadata til allmenne tjenester, og samtidig tilby mer detaljerte metadata for spesifikke applikasjoner og/eller domener, åpner man for flere bruksområder for metadataene.

2.6.6 Framtidsplaner

Foreløpig nyeste versjon av OAI-PMH (v2.0) ble lansert i juli 2002, og siden har det ikke skjedd oppdateringer av protokollen. I *[LAG03]* hevder lederne av Open Archives Initiative, Carl Lagoze og Herbert Van de Sompel, at det i hovedsak er to åpne spørsmål som blir viktige i tiden som kommer:

1. Hvorvidt en formell standardisering av protokollen hos en organisasjon som NISO, IETF eller W3C kan være nyttig, vurdert i forhold til hvor krevende det er å gjennomføre en slik prosess. Tendensen er foreløpig å fortsette en de facto utvikling inntil et virkelig standardiseringsbehov melder seg.
2. Utformingen av protokollens teknisk fremtid, hvor lederne ønsker forsiktighet i forhold til utvidelser av kjernefunksjonaliteten i protokollen. I denne sammenhengen bør det nevnes at det på Open Archives Initative sine offisielle epost-lister stadig blir diskutert potensielle forbedringer/utvidelser av protokollen. Ett av forslagene er å videreutvikle OAI-PMH slik at protokollen også kan fungere som en SOAP-basert webtjeneste.

KAPITTEL 3

Systemer basert på OAI-PMH

Et system basert på OAI-PMH kan mer eksplisitt beskrives som *et system basert på metadata som er innhøstet ved hjelp av OAI-PMH*. Protokollen utgjør med andre ord et basislag som det er mulig å bygge forskjellige typer tjenester oppå, for eksempel søketjenester. Denne typen systemer beskrives i OAI-terminologi som tjenestetilbydere.

I dette kapittelet beskrives et utvalg av eksisterende tjenestetilbydere som er basert på OAI-PMH. Ettersom dette prosjektet går ut på å utvikle et felles søkesystem basert på OAI-PMH, er det naturlig å undersøke lignende eksisterende løsninger med tanke på å få oversikt over erfaringer og systemarkitekturer. De mest interessante løsningene i denne sammenhengen er

- Arc
- OAIster
- Scirus

Før disse systemene beskrives nærmere, introduseres imidlertid først generelle arkitekturprinsipper for tjenestetilbydere basert på OAI-PMH. Mot slutten av kapittelet beskrives to tjenestetilbydere hvor søking ikke står i fokus, i tillegg til hvordan OAI-PMH anvendes hos Library of Congress.

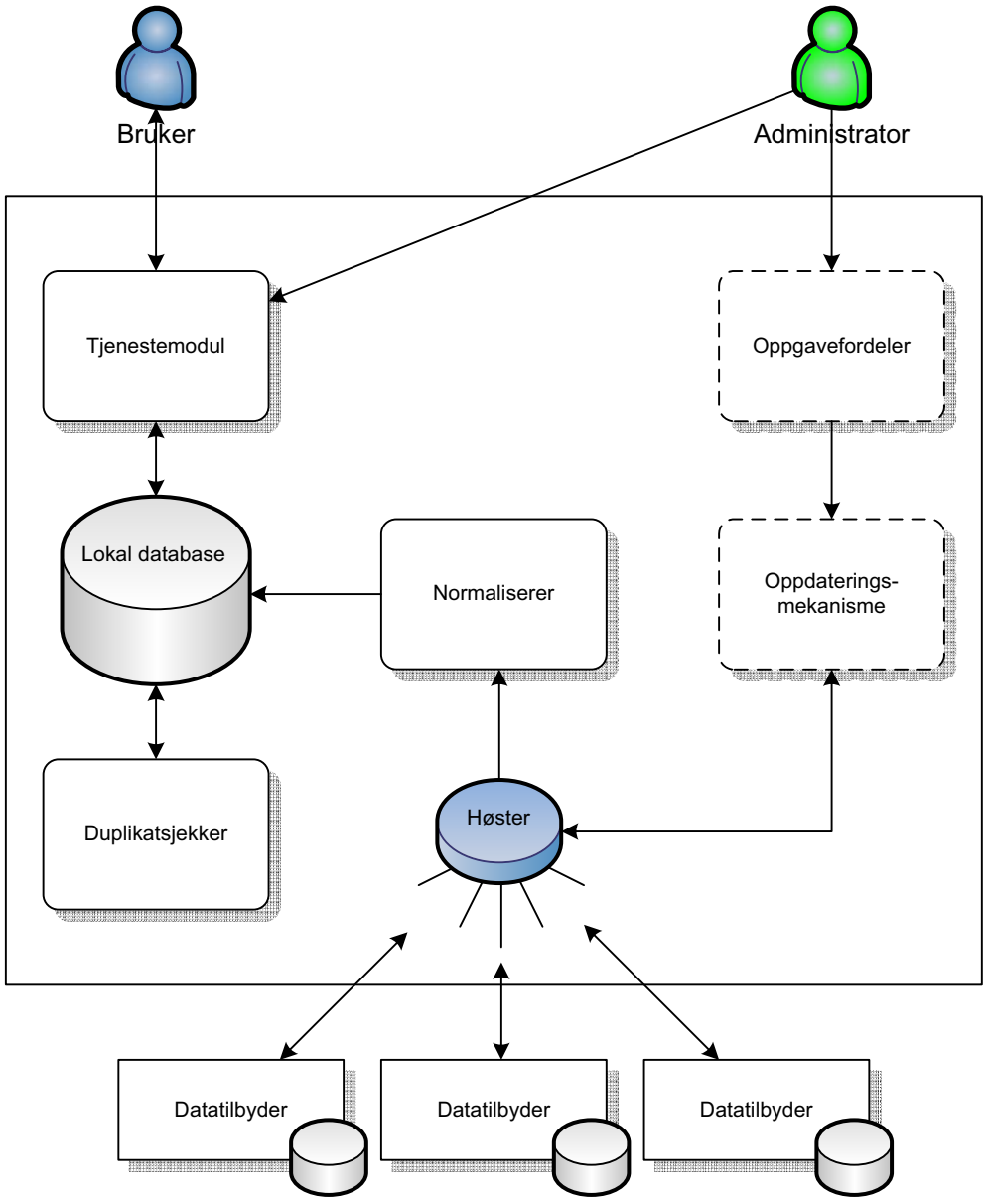
3.1 Introduksjon til implementasjon av OAI-PMH tjenestetilbydere

I Open Archives Forum sin OAI-veiledning, *[OAFOT]*, beskrives prinsipper og arkitekturforslag for implementasjon av både data- og tjenestetilbydere. Ettersom fokuset i denne oppgaven er å implementere en søketjeneste basert på metadata innhøstet ved hjelp av OAI-PMH, er prinsippene for et tjenestetilbydersystem mest interessante.

3.1.1 Komponenter og arkitektur hos tjenestetilbyder

Selv om OAI-PMH er en applikasjonsuavhengig spesifikasjon for metadatahøsting, er det likevel mulig å lage en generell arkitektur for oppbygningen av en tjenestetilbyder. De fleste av kompo-

nentene i denne arkitekturen finnes også i eksemplene på eksisterende løsninger. Illustrasjonen er basert på arkitekturforslaget i *[OAFOT]*.



Figur 3-1. *Generell arkitektur i OAI-PMH-basert tjenestetilbyder*

Høster

En høster er tidligere definert som en klientapplikasjon hos tjenestetilbyderen som sender ut OAI-PMH henvendelser for å samle inn metadataposter fra datatilbydernes datalager (*kap. 2.6.3, s. 26*). Høsteren består logisk av tre hovedkomponenter:

- **Henvendelseskomponent.** Lager HTTP-henvendelser for de tillatte OAI-PMH-verbene, og sender disse til datatilbyderne.
- **XML-parser.** Analyserer xml-responsen fra datalagrene, validerer metadataene ved hjelp av XML-skjema, og omformer metadataene fra den opprinnelige xml-strukturen til systemets interne datastruktur.
- **Flytkontroll.** Gjør det mulig å motta en lang resultatliste, for eksempel etter en ListRecords-henvendelse, i flere mindre deler. Implementeres ved hjelp av resumption tokens.

Normaliserer

En normaliserer rydder opp i de innhøstede metadataene ved å omforme forskjellige formater til en homogen struktur. I normaliseringsprosessen inngår for eksempel homogenisering av dato og språkkode. I tillegg er det mulig å oversette mellom forskjellige språk i denne fasen.

Dersom de innhøstede metadataene skal omformes til et internt dataformat, kan denne konverteringen eventuelt gjøres her.

Duplikatsjekker

En duplikatsjekker, som fjerner identiske metadatabeskrivelser, er en nyttig komponent i systemer som høster metadata fra mange forskjellige datatilbydere. Siden enkelte datatilbydere opptrer som aggregatorer, kan det oppstå tilfeller hvor de samme metadataene blir høstet fra flere kilder. I mindre systemer, med en viss grad av kildekontroll, er duplikatsjekkeren mindre nødvendig.

Tjenestemodul

Tjenestemodulen tilbyr systemets tjenester til brukere. Det relativt vide begrepet tjenestemodul omfatter alle typer tjenester et OAI-basert system kan tilby, som for eksempel søketjenester. Grunnlaget for alle typer tjenester er de innhøstede metadataene som er lagret i den lokale databasen.

Oppgavefordeler

En oppgavefordeler (scheduler) er en administrasjonskomponent som igangsetter både tidsbestemte og vanlige høstinger av arkiver. Igangsettingen kan enten være manuell eller automatisert.

Oppdateringsmekanisme

Oppdateringsmekanismen er en administrasjonskomponent som både sørger for å høste inn oppdaterte utgaver av metadata som allerede er lagret i den lokale databasen og å høste inn helt nye metadata.

Lokal database

I systemets lokale database lagres de normaliserte metadataene. Dersom det benyttes en relasjonsdatabase, må metadataene omformes fra xml-struktur til en tabellstruktur. Alternativt kan man benytte en xml-database.

3.2 Arc

Søketjenesten Arc *[ARC]*, som ble lansert i år 2001, er en av pionerene innen utviklingen av systemer for felles søkegrensesnitt basert på OAI-protokollen. I en foreløpig oppsummering av OAI-PMH, *[LAG03]*, nevnes Arc som et viktig bidrag til protokollens suksess.

Hovedideen bak Arc er at ett separat søkegrensesnitt skal gjøre det mulig for brukere å søke i metadata som er høstet fra mange forskjellige OAI datakilder. Bakgrunnen for prosjektet var - og er fortsatt - at brukbarheten ved mange av dagens internett-tidsskrifter og vitenskapelige digitale bibliotek er begrenset av mangelen på en felles/samlende tjeneste som gir dem et enhetlig brukergrensesnitt.

3.2.1 Datatilbydere og metadataformat

Arc er et generelt søkesystem som høster metadata fra forskjellige datatilbydere med forskjellige typer innhold, blant annet museumssamlinger og e-printsamlinger. På nåværende tidspunkt inneholder Arc mer enn én million metadataposter fra om lag 160 forskjellige datatilbydere. OAI Data Provider Registry *[OAIDPR]*, som er Open Archives Initiative sin offisielle oversikt over datatilbydere, utgjør utgangspunktet for hvilke datakilder Arc høster metadata fra.

Siden Arc-systemet høster metadata fra forskjellige typer datatilbydere, benyttes kun ukvalifisert Dublin Core som metadataformat. Et viktig poeng med Arc-systemet er ideen om å utnytte eventuelle metadatakontroller som eksisterer hos datatilbyderne. Dersom en datatilbyder benytter kontrollerte vokabularer eller kontrollerte verdier, fører dette til høyere kvalitet i metadataene, som igjen kan resultere i høyere kvalitet i en søketjeneste.

Ifølge *[LIU02]* tilsier erfaringer fra metadatahøstingen i Arc-prosjektet at det er stor sannsynlighet for at datatilbyderne benytter metadatakontroll i de tilfellene hvor antall records er mye større enn antall unike verdier. Spesielt feltene for type, format, spåk og dato utsettes ofte for kontroll. Det er imidlertid viktig å være klar over at hvert datalager kan anvende forskjellig semantikk for de samme feltene.

3.2.2 Systemarkitektur

Arkitekturen i Arc består av tre hovedkomponenter:

- OAI-høster
- Relasjonsdatabase
- Søketjeneste

Den egenutviklede høsteren sørger for å hente inn metadata fra datatilbyderne som er registrert i det interne administrasjonssystemet. Arc har både intern oppgavefordeler og oppdateringsmekanisme, slik at høstingen er mest mulig automatisert. Det generelle høstemønsteret er at det først foregår en massehøsting av alle metadataene hos en datatilbyder, og at det deretter utføres periodiske høstinger av enkeltelementer for å holde databasen oppdatert.

Alle innhøstede metadata gjennomgår en normaliseringsprosess før de lagres i systemets relasjonsdatabase, som er fulltekstindeksert for høyest mulig søkeytelse. Hver enkelt Dublin Core-element blir lagret som et selvstendig felt i databasen. Arc har støtte både for Oracle og MySQL som relasjonsdatabase.

Søketjenesten består av et webbasert søkegrensesnitt, hvor brukeren skriver inn sine spørringer, og en søkemotor som kommuniserer med databasen. For å oppnå best mulig søkeytelse, anvendes mellomlagring (caching) av søkeresultater.

I et system som Arc, hvor det skjer jevnlige oppdateringer av både datakilder og metadata, må søkegrensesnittet være tilpasningsdyktig til disse endringene. Dette for å sikre at eventuelt nye søkemuligheter også reflekteres i søkegrensesnittet. I Arc er denne utfordringen løst ved hjelp av en egen komponent for grensesnittbygging som dynamisk bygger et nytt, oppdatert grensesnitt på serversiden når nye datakilder og metadataposter legges til Arc-databasen. Grensesnittbyggeren endrer ikke layouten til søkegrensesnittet, men oppdaterer verdiene en bruker kan velge mellom når et søkefelt velges.

3.2.3 Søkegrensesnitt

Søkegrensesnittet i Arc tilbyr brukeren tre forskjellige søkemetoder *[LIU02]*:

1. **Enkelt søk.** Brukeren kan lage fritekstspørringer, og har mulighet for å benytte de boolske operatorene AND, OR og NOT. Systemet søker i alle metadatafeltene.
2. **Avansert søk.** Brukeren kan enten søke i bestemte metadatafelt (forfatter, tittel og abstract), eller søke/browse spesifikke datakilder. Når brukeren velger en bestemt datakilde, bygger systemet en rekke valgmuligheter for metadatafelt med kontrollert vokabular (arkiv, type, språk, emneord). Deretter kan brukeren velge en eksakt, kontrollert verdi, slik at systemet kan utføre et presist søk. Denne søkemetoden skalerer imidlertid dårlig.
3. **Interaktivt søk.** Prinsippet i denne søkemetoden er at brukeren formidler spørringen sin gjennom en interaksjon med systemet. I første fase av interaksjonen beskriver brukeren sin spørring ved hjelp av en rekke emneord, noe som resulterer i at systemet presenterer metadata fra de arkivene som har relevante metadataposter. I andre fase av interaksjonen benytter brukeren resultatene fra første fase til å velge interessante emnekategorier som raffinerer søket, slik at systemet gir enda mer detaljert respons.

Søkeresultatene presenteres i et todelt vindu: Den ene delen viser alle datakildene og antall treff for hver av disse, den andre delen viser oppsummeringsinformasjon om hvert enkelt søketreff for en valgt datakilde. I tillegg kan brukeren velge å se en detaljert metadatabeskrivelse for en ressurs, og eventuelt følge lenker til selve informasjonsressursen.

3.2.4 Erfaringer

I *[LIU01]* og *[LIU02]* beskrives forskjellige problemstillinger og erfaringer fra utviklingsprosessen av Arc:

- Kvaliteten på en felles tjeneste er sterkt avhengig av kvaliteteten på de enkelte datatilbyderne. Årsaker til variabel datakvalitet hos datatilbyderne kan være:
  - Feilstaving og/eller fravær av autoritetskontroll i de lokale systemene
  - Datatilbyderne bruker forskjellige klassifiseringsmetoder
- Ulike arkiver har ulike måter å formatere/navngi spesifikke metadatafelt, for eksempel dato. Dette nødvendiggjør datanormalisering.
- Bruk av kontrollert vokabular i søkegrensesnittet kan være et nyttig hjelpemiddel for å definere tvetydige metadatafelt og for å hjelpe brukerne med å formulere mer presise og pålitelige spørringer.
- Feil i xml-syntaks og tegnkodingsproblemer ble ofte påtruffet, noe som kan ugyldiggjøre store datasett.
- Ikke alle datatilbydere følger OAI-protokollen strengt nok, og enkelte datatilbydere har gjort semantiske feil (feiltolkning av Dublin Core-felt) i implementasjonen av OAI-protokollen.
- Høsting av metadata fra en ny datatilbyder krever andre teknikker enn periodisk høsting med tanke på å holde dataene oppdaterte. Hovedårsaken er at de store datamengdene som må overføres, kan belaste datatilbyderen unødig.
- Høstingsfrekvensen bør vurderes grundig, blant annet i forhold til oppdateringsfrekvensen hos datatilbyderne. Altfor hyppige høstinger overbelaster både søketjenesten og datatilbyderne, mens for få høstinger kan føre til synkroniseringsproblemer, det vil si at dataene hos tjenestetilbyderen blir foreldet i forhold til datatilbyderens.
- Hvis datatilbyderen benytter sikkerhetsbeskyttelse, for eksempel brannmur, kan dette blokkere høsteren.

En oppsummering av hvilke av disse erfaringene som er nyttige i forhold til systemet som skal utvikles i dette prosjektet, finnes i *kap. 3.7, s. 44*.

3.3 OAIster

OAIster *[OAISTER]* er - i likhet med Arc - et av de mest profilerte og omtalte OAI-baserte søkesystemene. OAIster ble lansert våren 2002, og er designet for å teste hvor godt OAI-PMH er egnet til høsting av metadata for digitale objekter fra mange og varierte digitale objekt-datalagre. OAIster skal dessuten tilby en tjeneste som gjør det mulig for sluttbrukere å få tilgang til innhøstede metadata.

Sammenliknet med andre tjenestetilbydere skiller OAIster seg ut på grunn av at tjenesten kun aggregererer metadata som har lenker til faktiske digitale ressurser/representasjoner. Dette kommer også fram i målsetningene for OAIster, beskrevet i *[HAG03]*:

- Ingen "blindgater": Sluttbrukere skal ha tilgang både til metadata og online-representasjoner av ressursene, ikke bare selve metadatabeskrivelsen.
- Avsløre digitale ressurser i den "skjulte webben", det vil si ressurser som befinner seg i databaser og som dermed ikke er tilgjengelige gjennom vanlige internettsøkemotorer.
- Tjenesten skal være lett gjenfinnbar og synlig.

3.3.1 Datatilbydere og metadataformat

Ifølge *[OAISTER]* inneholder OAIster-systemet 3,273,233 metadataposter fra 301 forskjellige datatilbydere per 7. juni 2004. Dette er en markant økning fra 1,538,431 metadataposter fra 197 datakilder, som var tallet da den samme nettsiden ble aksessert 8. september 2003.

Open Archives Initiative sin offisielle oversikt over datatilbydere, OAI data provider registry *[OAIDPR]*, er OAIster sitt utgangspunktet for hvilke datakilder det skal høstes metadata fra. I tillegg kan interesserte institusjoner kontakte OAIster for å bli lagt til på høstelisten.

På grunn av det store antallet datakilder med metadata som beskriver forskjellige typer ressurser, blir kun metadata i Dublin Core-formatet høstet inn av OAIster.

3.3.2 Systemarkitektur

Hovedkomponentene i OAIsters arkitektur er:

- Høster
- Mellomvare, Digital Library eXtension Service *[DLXS]*
- Søkemotor

Høsteren som OAIster benytter, Java OAI Harvester, er en java-applikasjon utviklet av et samarbeidende OAI-prosjekt *[UIUC]*. Fordelene ved å benytte en ferdigutviklet høster, framfor å måtte utvikle den selv, uttrykkes i *[HAG03]*: "We avoided most of the sticky issues associated with the use of the protocol, as we did not develop the harvester tool."

OAIster-systemet skiller seg fra Arc ved at det er basert på et eksisterende mellomvaresystem, Digital Library eXtension Service *[DLXS]*, istedenfor at alle komponentene er selvutviklet fra grunnen av. Blant egenskapene i DLXS finner vi et eget format for bibliografisk informasjon, *DLXS bibliographic class*, som alle de innhøstede metadataene blir omformet til. Denne operasjonen utføres av et xslt-basert omformingsverktøy, som også filtrerer vekk metadataposter uten lenke/URL til selve informasjonsressursen - i tråd med målsetningen om å unngå "blindgater". Omformingsverktøyet gjør også en enkel normalisering av innholdet i metadataelementet *dc:type*, for eksempel omgjøres både "illustration" og "picture" til "image".

Metadataene indekseres deretter i systemets database, og gjøres tilgjengelig for sluttbrukere via et grensesnitt som benytter DLXS-søkemotoren XPAT, *[XPAT]*.

3.3.3 Søkegrensesnitt

I OAIster brukes det samme søkevinduet både til å gjøre enkle søk og til å gjøre mer avanserte søk. I praksis innebærer dette at brukeren kan velge mellom tre alternativer:

- Ett søkefelt (à la Google) som fører til søk i alle felt i alle samlinger
- Søk i bestemte felt (tittel, forfatter, emneord, datoer og format)
- Søk innen bestemte samlinger.

OAIster er utviklet med tanke på å skulle brukes i operasjonelle systemer (production environments), noe som har preget utviklingsprosessen av søkegrensesnittet. Foreløpig siste versjon av OAIster ble lansert sommeren 2002 etter at utviklerne hadde gjennomført omfattende brukbarhetstester *[HAG03]*. For å lage et best mulig søkegrensesnitt ble det foretatt undersøkelser av brukernes behov gjennom spørreundersøkelser på Internett, og ut fra de innkomne resultatene ble det designet prototyper på søkegrensesnitt, som deretter ble testet på reelle brukere i direkte intervjuer. Denne prototypingen viste at brukerne ikke var i stand til å skjelne mellom et sammendrag av en metadatapost og den fullstendige metadataposten ved presentasjon av søkeresultater. OAIster viser derfor fullstendige metadatabeskrivelser direkte til brukeren i søkeresultatene.

3.3.4 Erfaringer

I likhet med Arc-prosjektet har det i løpet av OAIster-utviklingen også blitt gjort mange generelle erfaringer som er av allmenn interesse:

- I mer enn 10 prosent av repositoriene som ble høstet, ble det oppdaget xml valideringsfeil.
- Planlegging av høsting kan være utfordrende ettersom høstingsforsøk kan overlappe hverandre, og dermed forårsake problemer for minneintensive, samtidige prosesser.
- Innholdet i de forskjellige datalagrene er svært varierende, med tanke på emneområder, typer informasjonsobjekter, samt ulike akademiske nivåer i innholdet.
- Metadatabeskrivelsene har varierende kvalitet. Enkelte elementer, for eksempel subject, repeteres ofte mange ganger i samme metadatabeskrivelse, mens andre elementer, for eksempel rights svært ofte blir ignorert.
- Enkel normalisering av de innhøstede metadataene kan utføres ved hjelp av en normaliseringstabell som inneholder en oversikt over grunnleggende omforminger. En ordentlig normalisering vil imidlertid kreve thesaurus/kontrollert vokabular.
- Duplisering av innhøstede poster krever spesielle tiltak, for eksempel en duplikatsjekker. Duplisering kan oppstå når:
  - Metadataposter høstes både fra original datatilbyder og fra en aggregator-tilbyder som allerede har høstet hos original datatilbyder.
  - En organisasjon, som både er data- og tjenestetilbyder, høster sine egne metadataposter.
- Rettigheter for de digitale ressursene kan være problematisk. Selv om en organisasjon gir tilgang til metadata, betyr ikke dette at det er åpen tilgang til selve ressursen. Rettighetsinformasjonen (dc:rights) har ofte svært varierende kvalitet.
- Det eksisterer foreløpig ingen måter å uttrykke at man har med en offisiell utgave av et digitalt objekt å gjøre.

En oppsummering av hvilke av disse erfaringene som er nyttige i forhold til systemet som skal utvikles i dette prosjektet, finnes i *kap. 3.7, s. 44*.

3.4 Scirus

Scirus *[SCIRUS]* er en kommersiell vitenskapelig websøkemotor som gjør det mulig å søke i blant annet vitenskapelig websider, vitenskapelige tidsskrifter, upublisert forskning og konferanseagendaer. Søketjenesten ble lansert av forlaget Elsevier våren 2001, og har siden hatt en stadig økning i mengden indeksert materiale. Våren 2004 var offisielt 167 millioner vitenskapelige websider indeksert.

Scirus benytter tre forskjellige metoder til å samle inn vitenskapelige data:

1. **Webcrawler.** Den viktigste datainnsamlingsmetoden er webcrawleren, som traverserer verdensveven for å lokalisere vitenskapelige websider.
2. **OAI-PMH.** Metadatahøsting ved hjelp av OAI-PMH gir tilgang til vitenskapelig materiale som ellers ikke kan finnes ved å søke i en vanlig websøkemotor som for eksempel Google.
3. **Samarbeidsavtaler.** Scirus har direkte samarbeidsavtaler med flere kilder om indeksering av deres materiale, blant annet ScienceDirect, MEDLINE på BioMedNet, BioMed Central og patentkontoret i USA.

I denne sammenhengen er det spesielt interessant å undersøke Scirus sin datainnsamling ved hjelp av OAI-PMH.

3.4.1 Datatilbydere og metadataformat

Siden Scirus er konsentrert rundt vitenskapelige materiale, er det datakilder med denne typen innhold som er mest interessante. I motsetning til Arc og OAIster, høster Scirus vitenskapelige metadata fra bare et lite utvalg datakilder *[SCI03]*. Til gjengjeld er dette anerkjente OAI datatilbydere:

- arXiv.org
- NASA (inkludert NACA og LTRS)
- CogPrints
- Chemistry Preprint Server
- Computer Science Preprint Server
- Mathematics Preprint Server

Metadataformatet er Dublin Core.

3.4.2 Systemarkitektur

Det faktum at Scirus er en kommersielt utviklet søkeløsning, fører til at få detaljer om systemets arkitektur og indre oppbygning er tilgjengelige. Sammenlignet med rendyrkede OAI-systemer som Arc og OAIster, spiller OAI-PMH en mindre dominerende rolle i Scirus, hvor denne teknologien bare utgjør én av mange brikker. Likevel kreves de samme grunnleggende systemkomponentene; høster, lokalt datalager/indeks og søketjeneste.

Ifølge *[SCI04]* benytter Scirus en tilpasset utgave av en OAI-høster med åpen kildekode til å høste inn metadata fra de utvalgte datatilbyderne.

I Scirus anvendes Fast Enterprise Search Platform (ESP) fra *[FAST]* som indekseringsløsning og søkemotor. Dette er en komplett indekserings- og søkeløsning for alle typer ustrukturerte og strukturerte data, inkludert metadata som er innhøstet ved hjelp av OAI-PMH. Ifølge *[SCI04]* konverteres alle de innhøstede metadataene til et felles internt dataformat, før disse dataene indekseres i Fast Data Search gjennom Fasts API.

3.4.3 Søkegrensesnitt

Scirus har, i likhet med majoriteten av søketjenestene på web, både et enkelt og et avansert søkegrensesnitt. I det enkle søkegrensesnittet kan brukeren skrive inn én eller flere søketermer, og velge om det skal vises resultater fra webkilder og/eller tidsskriftkilder. Ut fra søkeresultatene genereres det en liste med nøkkelord som brukeren kan benytte til å raffinere søket.

I det avanserte søkegrensesnittet kan brukeren tilpasse søket sitt på forskjellige måter:

- Velge blant 20 forskjellige emneområder
- Begrense søket til bestemte tidsperioder
- Søke etter bestemte informasjonstyper, for eksempel artikler, bøker og websider
- Begrense søket til bare å gjelde bestemte datakilder
- Søke i ett eller flere av metadatafeltene tittel, navn, emneord, ISSN og URL

3.4.4 Erfaringer

Siden oppstarten i april 2001 har Scirus utviklet seg til å bli et viktig redskap for søking etter vitenskapelig materiale på web. Anvendelsen av OAI-PMH har vært et viktig bidrag i denne utviklingen, og Scirus kan muligens betraktes som en aksept av OAI-teknologien og -prinsippene - også kommersielt.

Selv om det finnes minimalt med vitenskapelig litteratur om erfaringer i forbindelse med Scirus, er systemet likevel interessant i forhold til dette prosjektet av to hovedgrunner:

1. Scirus illustrerer, i likhet med OAIster, at det er mulig å ta utgangspunkt i en allerede utviklet OAI-høster.
2. Scirus er et fungerende eksempel på at søketeknologien fra Fast kan anvendes som indekserings- og søkeløsning for metadata som blir høstet ved hjelp av OAI-PMH. I dette prosjektet er det forutsatt at begge disse teknologiene skal anvendes i prototypen.

3.5 Andre typer tjenestetilbydere

Selv om de aller fleste systemene som er basert på metadata høstet ved hjelp av OAI-PMH tilbyr en eller annen form for søketjeneste, eksisterer det også andre systemer som fokuserer på andre typer tjenester. To alternative tjenestetilbydere som illustrerer dette er DP9 og OAI Repository Explorer.

3.5.1 DP9

DP9 *[DP9]* er en OAI tjenestetilbyder som henvender seg til webcrawlere. Hensikten er å være en inngangstjeneste (gateway service) som gjør det mulig for generelle søkemotorer, for eksempel Google, å indeksere innholdet i arkiv med OAI-støtte. DP9 fungerer i praksis som et mellomledd mellom søkemotorens crawler og arkivet etter følgende prinsipp:

- DP9 tilordner en vedvarende URL til hver enkelt metadatapost i et datalager
- Når en crawler henvender seg til en URL som representerer en metadatapost i OAI-arkivet, oppfanger DP9 henvendelsen fra crawleren og oversetter den til en OAI-PMH-henvendelse
- OAI-arkivet returnerer den aktuelle metadataposten i xml-format.
- DP9 oversetter metadataposten fra xml til html, som crawleren forstår.

På denne måten kan også generelle søkemotorer indeksere opplysninger som ellers ville forblitt skjult.

3.5.2 OAI Repository Explorer

OAI Repository Explorer *[ORE]* er en tjenestetilbyder som i første rekke henvender seg til administratorer av OAI datalagre. Repository Explorer er et interaktivt webbasert verktøy for å teste om et datalager oppfyller kravene som OAI-datatilbyder, og fungerer i praksis som en OAI-høster med grafisk brukergrensesnitt. Mer konkret vil dette si at Repository Explorer gir brukeren mulighet til å teste alle OAI-PMH-henvendelsene (se *kap. 2.6.4, s. 28*) for et hvilket som helst OAI-datalager, og at eventuelt innhøstede metadata vises direkte i nettleseren.

3.6 OAI-PMH hos Library of Congress

Alle systemene som er beskrevet så langt i dette kapittelet, er tjenestetilbydere hvor tjenestene er basert på metadata innhøstet ved hjelp av OAI-PMH. Disse søketjenestene gir imidlertid begrenset oversikt over hvordan OAI-PMH passer inn i helhetsbildet i et digitalt bibliotek. I forhold til dette prosjektet er det derfor interessant å undersøke hvordan USAs nasjonalbibliotek, Library of Congress *[LC]*, anvender OAI-PMH, og hvordan det fungerer i rollen som OAI datatilbyder. Dette kan blant annet gi innspill om tekniske løsninger og praktiske problemer, samt hvordan disse kan unngås.

3.6.1 Oversikt

Library of Congress i USA er en av institusjonene som har lengst erfaring med bruk av OAI-PMH, ettersom protokollen tidlig ble tatt i bruk for å gjøre digitalisert materiale tilgjengelig for integrasjon i andre tjenester. Allerede i november 2000, før første offisielle utgave av OAI-PMH var lansert, hadde Library of Congress, ifølge *[ARC03]*, etablert en OAI datatilbyder for utvalgte digitaliserte samlinger fra prosjektet American Memory. Dette prosjektet er et forsøk på å samle historisk amerikansk materiale, og inneholder i juni 2004 mer enn hundre multimediesamlinger med over 7 millioner digitaliserte dokumenter, fotografier, lydopptak, film og tekst fra Library of Congress sine Americana-samlinger.

Bakgrunnen for at Library of Congress så raskt tok i bruk OAI-PMH var at protokollen gav mulighet til å gjøre ressursene mer tilgjengelige og nyttige for utenforstående organisasjoner gjennom høsting av metadataene. Da enkelte samlinger fra American Memory ble offentlig tilgjengelig på web i 1994, mottok Library of Congress etter hvert henvendelser fra flere organisasjoner som ønsket å integrere en hel samling eller deler av en samling i sine egne nettressurser.

Library of Congress hadde imidlertid ikke kapasitet til å velge ut og oversende deler av samlingene sine, samtidig som organisasjonene som ønsket å inkludere ressurser ikke hadde kapasitet til å motta en hel samling og gjøre utvelgelsen på egen hånd. I denne sammenhengen ble OAI-PMH et nyttig redskap:

> *"Indeed, one of the key benefits to LC from the harvesting protocol is that no special arrangements have to be made when others want to get the records and use them in another service." [ARC03], s. 135.*

I *[ARC03]* betraktes dessuten metadatahøsting som en alternativ måte å utvide tilgangen til innholdet i samlinger, ettersom lenkene fra metadatapostene fører brukerene tilbake til Library of Congress hvor ressursene befinner seg.

Antall metadataposter som er tilgjengelige for høsting har økt fra 3000 til mer enn 120 000 (mot slutten av 2002), og metadataene tilbys i tre formater:

- Dublin Core (se *kap. 2.2.1, s. 7*)
- MARCXML (se *kap. 2.2.3, s. 11*)
- MODS (se *kap. 2.2.4, s. 13*)

Både Arc og OAIster er blant tjenestetilbyderne som høster metadataposter fra samlingene i American Memory.

3.6.2 Erfaringer

Hovedmålet med OAI-PMH-prosjektet ved Library of Congress er å gjøre samlingene tilgjengelige for flere organisasjoner på en mer fleksibel måte. Selv om dette er noe forskjellig fra målet med dette prosjektet - å gjøre informasjon lettere tilgjengelig for brukere - finnes det likevel et viktig likhetstrekk: Begge prosjektene foregår i en bibliotekskontekst, det vil si American Memory er tilknyttet Library of Congress, mens dette prosjektet er tilknyttet Nasjonalbiblioteket.

Library of Congress sitt prosjekt er dessuten et praktisk eksempel på at MODS kan anvendes som metadataformat hos OAI datatilbydere.

3.7 Oppsummering

Gjennomgangen av søkesystemer basert på metadata innhøstet ved hjelp av OAI-PMH, har avdekket flere viktige prinsipper, problemstillinger og erfaringer både når det gjelder selve søketjenesten og når det gjelder datakildene.

Når det gjelder systemarkitektur, er det to erfaringer som er spesielt nyttige for det videre arbeidet i dette prosjektet:

1. Eksisterende høsterapplikasjoner kan problemfritt benyttes i et OAI-system.
2. Søketeknologi fra Fast kan integreres med OAI-PMH i en søketjeneste.

Enkelte av problemerfaringene er også interessante ettersom de kan gi nyttig informasjon om hvilke problemer som bør og kan unngås, og hva som eventuelt bør vektlegges for å oppnå et best mulig resultat i dette prosjektet. I det videre arbeidet med dette prosjektet er det viktig å være klar over at:

- Ikke alle datatilbydere følger OAI-protokollen strengt nok
- Metadatabeskrivelsene har varierende kvalitet, blant annet på grunn av:
    - Feilstaving og/eller fravær av autoritetskontroll i de lokale systemene
    - Forskjellige datatilbyderne bruker forskjellige klassifiseringsmetoder
- Feil xml-syntaks i metadatapostene forekommer ofte
- Ulike datakilder har ulike måter å formatere/navngi spesifikke metadatafelt, for eksempel dato. Dette nødvendiggjør datanormalisering til en enhetlig form.
- En enkel normalisering av de innhøstede metadataene kan utføres ved hjelp av en normaliseringstabell som inneholder en oversikt over grunnleggende omforminger. En ordentlig normalisering vil imidlertid kreve thesaurus/kontrollert vokabular.

Flere av problemene knyttet til utviklingen av en OAI-PMH-basert tjenestetilbyder skyldes altså feil eller mangler hos datatilbyderne. Siden dette prosjektet skal utvikles internt hos Nasjonalbiblioteket, har man imidlertid kontroll over alle datatilbyderne, og kan dermed løse eventuelle problemstillinger som skulle oppstå.

KAPITTEL 4

Systemarkitektur

Et hovedmål i dette prosjektet er å utvikle en prototyp på et søkesystem basert på metadata som er innhøstet ved hjelp av OAI-PMH.

Med grunnlag i erfaringene fra eksisterende systemer, skisseres i denne delen av oppgaven en løsningsarkitektur for prototypen som skal utvikles. Mer konkret innebærer dette:

- En beskrivelse av forutsetningene for systemet
- En analyse av hvilke krav systemet skal oppfylle, samt hvilke valg som er gjort.
- En formell kravspesifikasjon for systemet

4.1 Forutsetninger

Hele denne oppgaven, inkludert systemet som skal utvikles, er basert på tre grunnleggende betingelser etter ønske fra Nasjonalbiblioteket:

- Bruk av samlinger fra Nasjonalbiblioteket
- Bruk av OAI-PMH
- Bruk av Fast Data Search

4.1.1 Samlinger fra Nasjonalbiblioteket

Metadata fra flere av Nasjonalbibliotekets samlinger/databaser skal utgjøre datagrunnlaget for søketjenesten som skal utvikles. Det er på forhånd ikke sagt noe verken om antall samlinger eller hva slags type innhold (tekst, lyd, bilde, etc.) de bør ha.

En nærmere vurdering av hvilke samlinger som er interessante og hvilke valg som er gjort finnes i *kap. 4.2.2, s. 51*.

Implementasjon av OAI datatilbyderfunksjonalitet for de samlingene som velges skal gjøres av Nasjonalbiblioteket, siden dette ikke er hovedfokus i dette prosjektet. Applikasjonen OAICat

[OAIC] anvendes som utgangspunkt for å gi de utvalgte samlingene funksjonalitet som OAI data-tilbydere.

4.1.2 OAI-PMH

Systemet skal baseres på metadatahøsting som interoperabilitetsmetode, og OAI-PMH (se *kap. 2.6, s. 25*) skal anvendes for å realisere høstingen. Dette er en svært grunnleggende forutsetning som legger føringer på oppbygningen av hele systemet.

Konkret fører denne forutsetningen til at det må gjøres vurdering og valg av en OAI høster (se *kap. 4.2.4, s. 54*). Dessuten må det velges et felles metadataformat for alle metadata som skal høstes (se *kap. 4.2.3, s. 52*).

4.1.3 Fast Data Search

Alle søketjenester på Internett er basert på en søkemotor som søker i data lagret i en lokal indeks. Dette prinsippet gjelder også i dette prosjektet, hvor det forutsettes at den kommersielle søke- og indekseringsløsningen Fast Data Search (versjon 3.2) skal anvendes til formålet. Som eksempelet Scirus viser (se *kap. 3.4, s. 41*), er det fullt mulig for et Fast-system å indeksere metadatabeskrivelser som er innhøstet ved hjelp av OAI-PMH. Ytterligere informasjon om anvendelse av Fast Data Search i digitale bibliotek finnes i *[LER04]*.

Siden Fast Data Search spiller en såvidt viktig rolle i systemet som skal utvikles, introduseres noen grunnleggende definisjoner og prinsipper i systemets virkemåte. Beskrivelsen av Fast Data Search er basert på *[FDS03]*.

Terminologi

Fast Data Search er basert på tre grunnleggende begreper:

- **Innhold.** Alle typer data som er eksterne i forhold til Fast Data Search, og som ennå ikke har blitt formidlet til Fast-systemet.
- **Dokument.** Data inni Fast Data Search, det vil si en intern representasjon av en innholdsentitet som har blitt konvertert til formatet FastXML.
- **Samling.** Intern gruppering av dokumenter som er søkbare.

Fast Data Search benytter en FileTraverser-applikasjon til å hente innholdsfiler inn i systemet.

FastXML

Alle innhøstede metadata (innhold) må først konverteres til FastXML-dokumenter før de kan indekseres i Fast Data Search. FastXML, som er det interne formatet for xml-dokumenter, har en relativt enkelt struktur basert på begrepene dokumenter og elementer, hvor hvert dokument kan bestå av flere elementer.

```
<documents>
     <document>
          <element></element>
          ...
          <element></element>
     </document
     ...
     <document>
          <element></element>
          ...
          <element></element>
     </document
</documents>
```

Innholdet i disse elementene er informasjon fra den originale innholdsentiteten, i dette tilfellet en metadatabeskrivelse. Siden metadatabeskrivelser er en strukturert oppbygning av elementer, er det mulig å lage en 1:1-avbildning mellom elementer i metadatabeskrivelsene og elementer i FastXML-dokumentet.

Indeksprofil

For å kunne indeksere innhøstede metadata, og senere søke i disse indekserte dataene, må det utvikles en indeksprofil for Fast Data Search. Dette er en xml-basert konfigurasjonsfil som definerer måten de indekserte dokumentene skal være søkbare på. Indeksprofilen definerer med andre ord layouten til den søkbare indeksen, og spesifiserer på hvilken måte de ulike feltene skal behandles under spørring og resultatprosessering. Følgende egenskaper beskrives i en indeksprofil:

- Hvilke dokumentelementer som skal være søkbare felt, det vil si en avbildning fra elementer i FastXML-dokumenter (mods.titleInfo) til søkbare felt i indeksen (modstitle)
- Hvilke dokumentelementer som skal returneres som en del av resultatsettet
- Hvordan verdier som brukes til sortering og rangering skal beregnes
- Hvilke enkeltfelt som skal inngå i sammensatte felt. Et sammensatt felt er en gruppering av enkle felt som kan refereres ved hjelp av ett unikt navn i spørringer
- Én eller flere resultatpresentasjoner (result views) som definerer hvilke felt i indeksen som skal returneres og vises i resultatlisten etter et treff for en spørring

Hovedelementene i en standard indeksprofil er strukturert på følgende måte:

```
<index-profile name=”ip”>
     <field-list>
     <composite-field>
     <result-specification>
</index-profile>
```

Fast Data Search har en medfølgende indeksprofil som er utviklet med tanke på indeksering av webdokumenter, men siden det benyttes et strukturert metadataformat i dette prosjektet, må det utvikles en egen indeksprofil tilpasset strukturen i dette metadataformatet.

Søkeegenskaper

Når de innhøstede metadatabeskrivelsene har blitt indeksert i Fast Data Search, kan det søkes i disse beskrivelsene. Fast Data Search har følgende egenskaper for søking:

- Spørringer (queries) til Fast Data Search Index formidles ved hjelp av `http get`, og må samsvare med en bestemt syntaks og grammatikk
- Spørrespråket "Fast Data Search Query Language" benyttes til å utføre eksakte søk, til å begrense definisjonsområdet for et søk til verdier som tilhører en bestemt samling i Fast Data Search, og til å begrense søket til en bestemt indeks innenfor en angitt samling
- Søkeresultater skal formateres i henhold til mønsteret i en angitt output template, for eksempel ren tekst, html eller xml

Oppsummering

Fast Data Search er et kraftig indekserings- og søkesystem som kan anvendes på mange forskjellige typer data, også metadata. For å kunne indeksere innhøstede metadata kreves:

- En konverteringsrutine som omformer metadata fra det opprinnelige metadataformatet til FastXML
- En indeksprofil som definerer hvilke metadatafelt som skal være søkbare

4.2 Analyse og valg

For å kunne lage en formell kravspesifikasjon for systemet, er det nødvendig med en analyse av de forskjellige delene av systemet og eksterne faktorer som påvirker systemet. Valgene som gjøres her kompletterer systemforutsetningene.

4.2.1 Brukere

I den generelle oversikten over systemarkitektur for en OAI-PMH-basert tjenestetilbyder (se *kap. 3.1.1, s. 33*), forekommer to kategorier brukere:

- **Administrator.** Person som jobber med konfigurasjon og drift av systemet.
- **Bruker.** Alminnelige sluttbrukere som benytter tjenestene systemet tilbyr.

Enhver tjeneste er utviklet med tanke på å tilfredsstille et behov hos sluttbrukerne. I dette prosjektet er brukerne informasjonssøkere med behov for enklere tilgang til informasjon, og hovedmålet blir dermed å utvikle en søketjeneste som dekker dette behovet. Av den grunn blir de delene av systemet som er viktige for sluttbrukerne prioritert i den videre planleggingen og utviklingen av systemet.

Administratorenes behov, som er av en helt annen karakter enn informasjonsøkernes, vil ikke bli vektlagt i fortsettelsen. En medvirkende årsak til dette er at administratorer utgjør en mye mindre brukergruppe enn sluttbrukerne. Konkret fører dette til at komponentene som er knyttet til administrasjon av systemet, oppgavefordeler og oppdateringsmekanisme, ikke blir prioritert. I et operasjonelt system er det imidlertid viktig å ta hensyn til både til sluttbrukere og administratorer.

4.2.2 Samlinger

Selv om det er forutsatt at samlinger fra Nasjonalbiblioteket skal brukes som datakilder i systemet, er det ikke gitt noen rammer verken for hvor mange samlinger som skal inngå i systemet, eller hva slags type innhold samlingene skal ha.

Ett av målene med dette prosjektet er å utvikle en prototyp på en OAI-PMH-basert søketjeneste. En slik prototyp skal illustrere at det er mulig å lage et integrert søkegrensesnitt basert på metadata fra forskjellige samlinger med forskjellige typer innhold.

I enkelte systemer, som for eksempel OAIster (se *kap. 3.3, s. 38*), blir det høstet metadataposter fra flere hundre forskjellige datakilder. I dette forskningsprosjektet, hvor selve prinsippet skal illustreres, er det imidlertid tilstrekkelig med 3-5 datakilder. Det er likevel viktig at de samlingene som velges inneholder beskrivelser av forskjellige typer informasjonsobjekter, for eksempel tekstdokumenter, bilder, lyd eller video. Dette fordi det vil kunne gi en pekepinn på hvilke utfordringer som er knyttet til felles metadataformater, metadatanormalisering og integrering av forskjellige typer metadata i én indeks.

Som et kompriss mellom hvilke samlinger det er praktisk mulig for Nasjonalbiblioteket å utvikle til OAI datatilbydere og hvilke samlinger som er interessante å jobbe med i dette prosjektet, er det valgt tre (3) konkrete samlinger som inneholder forskjellige typer informasjonsobjekter.

Tabell 4-1. *Samlinger som skal benyttes i dette prosjektet.*

Samling	Innhold	Metadata beskriver	Totalt antall beskrivelser	Antall som anvendes i prototypen
Galleri NOR	Metadata og bildefiler	Fotografier	ca. 71 000	1000
Digitalt Radioarkiv (DRA)	Metadata og lenker til lydfiler	Radiosendinger	ca. 32 000	281
Mavis	Metadata og lenker til digitale objekter	Multimedia	ca. 203 000[a]	122

a. Mange av disse postene er helt enkle registreringer som kun brukes for fysisk kontroll av materialet

Galleri NOR

Galleri NOR *[GNOR]*, som er Nasjonalbibliotekets fotodatabase, inneholder mer enn 71000 bilder som er tilgjengelige på internett. Fotodatabasen har vært tilgjengelig på internett siden 1996, og ifølge Nasjonalbiblioteket er hensikten med galleriet å presentere materiale fra flere sentrale aktø-

rer i norsk fotohistorie. Fotografiene i Galleri NOR er tatt i tidsperioden 1880 - 1970. Galleriet inneholder både selve bildene og metadatabeskrivelser av bildene i et format basert på en revidert utgave av feltkatalogen *[NKKM]*.

I dette prosjektet skal det anvendes et utdrag på 1000 metadatabeskrivelser fra Galleri NOR.

Digitalt radioarkiv (DRA)

Digitalt radioarkiv (DRA) er et samarbeidsprosjekt mellom Nasjonalbiblioteket og NRK som ble startet i år 2000. Hovedideen med arkivet er digitalisering og metadatabeskriving av historiske lydbånd, samt metadatabeskriving av moderne digitale opptak. Målet er å gi publikum tilgang til materialet gjennom internett, men foreløpig er ikke dette oppnådd på grunn av uløste rettighetsspørsmål. Radioarkivet inneholder både metadatabeskrivelser for lydopptakene og lenker til lydfilene, og totalt er det digitalisert ca. 32 000 bånd med til sammen 24 000 timer lydopptak i arkivet.

I dette prosjektet skal det anvendes et utdrag på 281 metadatabeskrivelser fra Digitalt Radioarkiv.

Mavis

Mavis, Merged Audio Visual Information System, er et integrert system for lagring, administrasjon og gjenfinning av multimedieressurser og tilhørende metadata ("media asset management software"). Systemet håndterer alle medietyper, og gir mulighet for å beskrive både verk og informasjonsbærer. Nasjonalbiblioteket har benyttet Mavis siden midten av 1990-tallet, og systemet brukes til å administrere metadata for lydmateriale, filmmateriale og pliktavlevert kringkastingsmateriale. Mavis er ikke tilgjengelig for søking på web, med unntak av spesialdatabasen *Jazzbasen*. Systemet benyttes i tillegg til Nasjonalbiblioteket av Library of Congress i USA og Bundesarchiv i Tyskland.

I dette prosjektet skal det anvendes et lite utdrag på 122 metadatabeskrivelser fra Mavis. Alle disse beskrivelsene gjelder videofiler.

4.2.3 Metadataformat

Metadataformatet spiller en viktig rolle i systemet, ettersom dette utgjør et felles bindeledd mellom de forskjellige datakildene og søketjenesten basert på OAI-PMH.

Tidligere har de to formatene Dublin Core og MODS blitt sammenlignet med hverandre (*kap. 2.2.5, s. 17*). Kort oppsummert kan man si at MODS gir rom for relativt detaljerte strukturerte beskrivelser, mens Dublin Core er et enklere format som kan brukes til å uttrykke grunnleggende metadata.

I de aller fleste systemene som er basert på OAI-PMH er de innhøstede metadatapostene beskrevet i Dublin Core, hovedsakelig fordi det er obligatorisk for OAI datatilbydere å tilby metadata i dette formatet. Dette systemet skal utvikles i en bibliotekskontekst hvor det tradisjon for å bruke rike

metadataformater (for eksempel MARC), og hvor Dublin Core foreløpig ikke er veletablert som metadataformat. Ifølge *[DIL03]* eksisterer det tre hovedgrunner for at Dublin Core har blitt lite akseptert i biblioteksverdenen:

1. Ufullstendighet - i form av manglende kvalifikatorer, spesielt for elementene Creator, Contributor og Publisher.
2. Manglende retningslinjer for bruk av kvalifikatorer, noe som blant annet fører til nyoppfinnelse av lokale instruksjoner i hvert nytt brukerfellesskap. Dette er en hovedårsak til at Dublin Core Library Application Profile (se *kap. 2.2.2, s. 9*) er under utarbeidelse.
3. Treg innføring av Dublin Core, spesielt for biblioteksapplikasjoner.

MODS muliggjør rikere ressursbeskrivelser enn Dublin Core kan tilby, også i kvalifisert form. I tillegg er mulighetene for hierarkiske beskrivelser i MODS en egenskap som ikke eksisterer i Dublin Core. I *[DIL03]* hevdes det at MODS er å foretrekke primært på grunn av disse ulempene med Dublin Core.

Én av utviklerene av MODS, Rebecca Guenther, har også vært ansvarlig for utviklingen av Dublin Core Library Application Profile (DC-Lib). På direkte spørsmål om å vurdere de to formatene mot hverandre, svarer hun:

> *DC-Lib imposes some contraints on Dublin Core so that you know what you can expect. Especially with complex digital library objects that require more complex descriptions that show the relationship between objects, I would suggest MODS. Also, if you are trying to integrate MARC records with other metadata, MODS is closer to MARC and allows more complex descriptions, so is useful. We expose our OAI harvested records for some of our collections with MODS as one of the alternatives. The reason is that OAI knows simple Dublin Core and full MARCXML and MODS gives a more user-friendly output than full MARC but it retains more of the original MARC data in MODS than Dublin Core could. [GUE03c]*

Siden samlinger fra Nasjonalbiblioteket utgjør utgangspunktet for dette prosjektet, er det sannsynlig at det kan forekomme komplekse digitale biblioteksobjekter (complex digital library objects). Med tanke på at systemet etter hvert også kan bli utvidet til å dekke samlinger med MARC-metadata, synes det fornuftig å velge et rikere metadataformat enn Dublin Core.

Et annet poeng er at MODS er et såvidt nytt metadataformat at det er ikke har blitt anvendt som fellesformat i andre OAI-PMH-prosjekter. Prototypen som skal utvikles vil derfor være en anledning til å undersøke hvor godt egnet MODS er i denne typen systemer.

På bakgrunn av disse vurderingene, synes det klart at MODS har flere fortrinn i forhold til Dublin Core i bibliotekskonteksten som dette prosjektet skal utvikles i. MODS velges derfor som felles metadataformat.

4.2.4 OAI-høster

OAI-høsteren, som sørger for å samle inn metadata, utgjør også en viktig komponent i systemet som skal utvikles. Et avgjørende spørsmål er på hvilken måte høsteren skal utvikles, og i praksis eksisterer det to alternativer:

1. Utvikle en helt ny høsterapplikasjon, og dermed "finne opp hjulet på nytt".
2. Ta utgangspunkt i en allerede utviklet høsterapplikasjon med åpen kildekode, og tilpasse den til dette systemet.

De positive erfaringen fra prosjektene OAIster (se *kap. 3.3, s. 38*) og Scirus (se *kap. 3.4, s. 41*), tilsier at det er både fornuftig og arbeidsbesparende å ta utgangspunkt i en allerede utviklet applikasjon.

På Open Archive Initiative sin liste over OAI-verktøy *[OAIT]* finnes det flere OAI-høstere med åpen kildekode. Ut fra positiv omtale i *[LAG03]* velges OCLC-applikasjonen OAIHarvester2 *[OAIH]* som utgangspunkt i dette systemet. OAIHarvester2 har følgende egenskaper:

- Åpen kildekode
- Støtter OAI-PMH versjon 1.1 og 2.0
- Kommandolinjebasert Java-applikasjon
- Lagrer innhøstede metadata som filer eller lister dem opp i konsollvinduet

På grunn av at kildekoden til OAIHarvester2 er åpen, er det relativt enkelt å gjøre eventuelle utvidelser og/eller lokale tilpasninger.

4.2.5 Søkegrensesnitt

Søkegrensesnittet, som er systemets kontaktpunkt med brukeren, skal muliggjøre søking i de indekserte, innhøstede metadataene, og i tillegg presentere søkeresultatene for brukeren. Søkegrensesnittet skal være webbasert, mest av hensyn til tilgjengelighet, men også på grunn av at dette er en velkjent utviklingsmetode.

I *[ROS02]* nevnes flere variabler som påvirker design av søkegrensesnittet:

- Brukerens kunnskapsnivå om søking: Novise i forhold til ekspert
- Typen resultater brukeren ønsker: Oversikt i forhold til detaljer
- Typen informasjon som det søkes i: Strukturerte felt i forhold til fulltekst
- Mengden informasjon som det søkes i

I brukeranalysen (se *kap. 4.2.1, s. 50*) ble det slått fast at målgruppen for søketjenesten som skal utvikles er informasjonssøkere. Dette er imidlertid en uensartet gruppe som består av mange forskjellige typer brukere med forskjellige søkebehov og ulike kunnskaper om søking. Søkegrensesnittet som skal utvikles må derfor ha støtte både for enkel og avansert søking, og søkeresultatene må presenteres på en oversiktlig måte. Det bør her legges til at det avanserte søkegrensesnittet ikke bør gjøres for avansert, siden dette kan føre til at det blir problematisk for brukeren å forstå.

Selv om alle metadataelementer kan gjøres søkbare, er det bare enkelte elementer som er egnet til søking og som er interessante for brukerne å søke i. Eksempler på slike elementer er:

- Tittel
- Navn
- Emneord
- Abstract (tekstlig ressurssammendrag)

Denne typen elementer ble i *kap. 2.1.3, s. 5* omtalt som *felles aksesspunkter* for alle typer metadatabeskrivelser, og de utgjør derfor basisen for søkegrensesnittet som skal utvikles i dette prosjektet.

4.3 Systemspesifikasjon

På bakgrunn av forutsetningene og analysen av ulike faktorer som påvirker oppbygningen av systemet, er det mulig å lage en formell systemspesifikasjon. Her beskrives hensikten med og oppførselen til programvaresystemet som skal utvikles.

Systemspesifikasjonen består av følgende deler:

- Funksjonelle krav
- Begrensninger og antakelser
- Krav til dokumentasjon

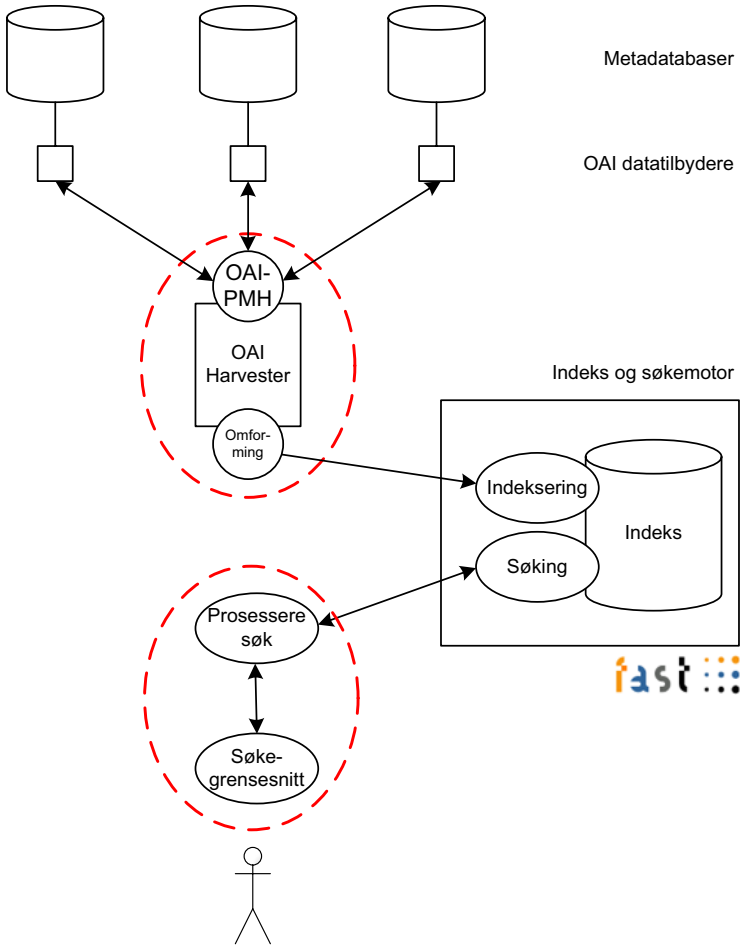
I en kravspesifikasjon for et operasjonelt system ville det også vært naturlig å fokusere på forhold knyttet til iverksettelse og drift av systemet. Ettersom dette er et forskningsprosjekt som skal resultere i en prototyp, er det imidlertid ikke lagt vekt på:

- **Ikke-funksjonelle krav.** Krav som ikke gjelder funksjonaliteten ved et system, men som er viktige for helheten når et system tas i bruk. Eksempler på ikke-funksjonelle krav er brukbarhet, ytelse, pålitelighet, sikkerhet og maskinvarekrav.
- **Valideringskriterier.** Detaljert beskrivelse av forskjellige tester som må gjennomføres for å verifisere at programvaren fungerer slik den skal.

Systemet som skal utvikles består av to forskjellige hoveddeler:

1. Systemet skal ved hjelp av OAI-PMH høste metadata fra forskjellige metadatabaser med heterogent innhold, og deretter omforme de innhøstede metadataene til et indekserbart format. De indekserbare metadataene skal deretter lagres i søkeindeksen.
2. Systemet skal tilby et enhetlig søkegrensesnitt som gjør det mulig å søke i de indekserte metadataene. Søkeindeksen vil fungere som bindeledd mellom de to hoveddelene.

I denne sammenhengen er det viktig å klargjøre at den første delen - *metadatahøsting* - representerer en prosess som kun forekommer en sjelden gang, mens den andre delen - *søking* - representerer en prosess som gjentas stadig.



Figur 4-1. *Konseptuell modell*

4.3.1 Funksjonelle krav

Funksjonelle krav beskriver hendelser systemet skal utføre, det vil si hvilken input som initierer en hendelse (stimulus), hva som skjer når systemet utfører hendelsen, samt hvilken output hendelsen resulterer i (respons). Nedenfor følger en beskrivelse av de funksjonelle kravene for de forskjellige komponentene i systemet.

OAI datatilbydere

Ettersom Nasjonalbiblioteket skal utvikle funksjonaliteten som gjør de utvalgte samlingene til OAI datatilbydere, ligger funksjonelle krav knyttet til dette utenfor definisjonsområdet til denne kravspesifikasjonen. Det eksisterer imidlertid ett unntak for dette systemet: Formatet for metadatabeskrivelsene må være kjent for å kunne utvikle en avbildning fra den interne strukturen i metadatabasene.

- I dette systemet skal MODS, som er et metadataformat for bibliografiske beskrivelser, benyttes

OAI-høster

OAI-høsteren i dette systemet skal baseres på applikasjonen OAIHarvester2 *[OAIH]*. For at høsterapplikasjonen skal passe inn i dette systemet, må det imidlertid gjøres en utvidelse:

- Høsteren skal utvides slik at de innhøstede metadataene - ved hjelp av xslt - kan omformes til andre formater og lagres i disse formatene
- I dette systemet skal metadataene omformes fra MODS til Fast Data Search sitt interne dataformat FastXML

OAI-høsteren skal være en kommandolinjebasert selvstendig applikasjon, det vil si at den skal ikke integreres med resten søkesystemet i et eget administrasjonsgrensesnitt.

Metadataomforming og normalisering

Kravene til metadataomformingen er:

- Malene for omforming av metadata fra MODS-struktur til FastXML-struktur skal defineres i et xslt-stilark.
- Det skal gjøres en enkel normalisering (homogenisering) av alle innhøstede metadata, slik at forskjellige datotyper omformes til et felles format.

Indeks og søkemotor

Fast Data Search versjon 3.2 (se *kap. 4.1.3, s. 48*) skal benyttes som indekserings- og søkesystem.

Innledningsvis i kravspesifikasjonen ble det nevnt at søkeindeksen skal fungere som et bindeledd mellom systemets to hoveddeler; metadatahøstingen og søkegrensesnittet. Det må derfor gjøres tilpasninger både på indekserings- og søkesiden.

Krav til indeksering:

- Følgende metadatafelt skal indekseres slik at de blir søkbare:
  - Title
  - Name
  - TypeOfResource
  - Publisher
  - Abstract
  - DateCreated
  - Subject
- Følgende felt skal ikke være søkbare. De skal imidlertid vises i søkeresultatlisten:
  - NameType
  - NameRole
  - Genre
  - Extent
  - InternetMediaType
  - Identifier
  - AccessConditions
- Det skal opprettes en ny samling i Fast Data Search som skal inneholde alle de innhøstede metadatabeskrivelsene

- Det skal utvikles en indeksprofil som definerer indeksstrukturen for Fast Data Search
- Fast-applikasjonen FileTraverser skal brukes for å hente filene som inneholder de innhøstede metadatabeskrivelsene inn i Fast Data Search

Søkeprosessering

Systemet skal ha et mellomvarelag som styrer kommuniksjonen mellom brukergrensesnitt og Fast Data Search. Dette mellomvarelaget skal:

- Motta input fra brukergrensesnittet.
- Bruke inputen til å formulere spørringer i spørrespråket til Fast Data Search.
- Videreformidle spørringene til Fast Data Search.
- Motta søkeresultater fra Fast Data Search.
- Presentere søkeresultatene for brukeren.

Mellomvarelaget skal ikke:

- Behandle søkeuttrykk med informasjonsgjenfinningsfunksjoner, for eksempel stoppordliste og stemming.
- Behandle inputen fra bruker med f.eks. konvertering av store og små bokstaver, strengtrimming, boolske parametre og lignende.

For å korte ned utviklingstiden, skal mellomvarelaget baseres på eksisterende programkode som følger med Fast Data Search.

Søkegrensesnitt

Generelle krav:

- Søkegrensesnittet skal være webbasert
- Søkegrensesnittet skal støtte både enkel og avansert søking

Krav til enkelt søk:

- Skal gjøre det mulig for brukeren å skrive inn ett eller flere søkeord i ett søkefelt (à la Google).
- Systemet skal søke i følgende metadatafelt i alle indekserte metadataposter
  - Abstract
  - Utgiver
  - Navn
  - Emneord (subject)
  - Tittel
  - Undertittel
- AND skal være standard boolsk operator mellom søkeordene dersom bruker oppgir flere søkeord.
- Brukeren skal kunne oppgi fraser på følgende måte *"dette er en frase"*.

I enkelt søk vektlegges ikke:

- Bruk av boolske operatorer, dvs. hvis brukeren eksplisitt benytter operatorene AND, OR og NOT, skal systemet ikke ta hensyn til disse.

Krav til avansert søk:

- Skal gjøre det mulig for erfarne brukere å detaljspesifisere hva det skal søkes etter og hvor søket skal foregå. Det skal med andre ord være mulig å søke i et bestemt felt eller kombinasjoner av flere angitte felt.
- Brukeren skal kunne oppgi ett eller flere søkeord i ett eller flere av søkefeltene:
  - Tittel
  - Navn
  - Utgiver
  - Emneord (subject)
  - Abstract
- Brukeren skal kunne angi hvilken ressurstype han/hun søker etter:
  - Tekst
  - Lyd
  - Bilde
  - Video
- Brukeren skal kunne angi hvilken datakilde metadatabeskrivelsene opprinnelig kommer fra
- Brukeren skal kunne gjøre datosøk:
  - Alle datoer
  - Eksakt dato
  - Tidsintervall (mellom to datoer)
- I søkefelt hvor det er naturlig at brukeren velger verdier fra et kontrollert vokabular (for eksempel ressurstype, dato o.l.), skal dette benyttes
- Brukeren skal kunne nullstille alle søkeparameterene på en enkel måte, siden avansert søk kan inneholde mange forskjellige søkeparametere

Når brukeren igangsetter både enkelt og avansert søk, skal alle søkeparametrene sendes videre til mellomvarelaget for videre prosessering.

Krav til presentasjon av søkeresultater:

- Systemet skal liste opp antall søketreff
- Søkeresultatene skal presenteres i rangert rekkefølge (i henhold til relevansrangering gjort av Fast Data Search)
- Søkeresultatene skal inneholde beskrivelse av hvilken datakilde metadatabeskrivelsen opprinnelig kommer fra, siden metadatabeskrivelsene i et søkeresultat kan ha sin opprinnelse i forskjellige datakilder
- Søkeuttrykket skal repeteres i en søkeboks sammen med søkeresultatene, slik at brukeren har mulighet til å revidere søket etter å ha sett søkeresultatene.

Resultatlistas design og brukbarhet skal ikke vektlegges nevneverdig.

4.3.2 Begrensninger og antakelser

Selv om alle kravene til systemet er beskrevet i detalj, finnes det eksterne faktorer som kan påvirke utviklingsprosessen, og som derfor må tas hensyn til.

Interaksjon med ekstern programvare

Som tidligere nevnt, er det forutsatt at Fast Data Search skal benyttes som indekserings- og søkesystem. Dette er et kraftig system som trolig vil komme til stor nytte, forutsatt at utviklingsprosessen planlegges slik at integrasjonen med dette eksisterende systemet foregår uten problemer. Fast Data Search har et klart definert grensesnitt som gjør integrasjonen med andre systemer enkel.

Programmeringsspråk

Systemet som skal utvikles består av flere forskjellige komponenter som er basert på forskjellige programmeringsspråk:

- Videreutviklingen av OAIHarvester2 skal gjøres i Java, siden dette er programmeringsspråket som er benyttet til å utvikle applikasjonen
- Stilarkene som omformer og normaliserer metadata skal utvikles i xslt
- Utviklingen av mellomvarelaget for søkeprosessering skal gjøres i PHP, siden basisskriptene som følger med Fast Data Search er utviklet i dette skriptspråket

I tillegg krever alle utviklingsfaser god forståelse av xml.

4.3.3 Krav til dokumentasjon

I ethvert utviklingsprosjekt skal det produseres dokumentasjon som beskriver det utviklede systemet. Selv om dette er et forskningsprosjekt, hvor dokumentasjon kanskje ikke er den delen som bør vektlegges mest, skal følgende dokumenter produseres:

- Brukerveiledning, det vil si en beskrivelse av hvordan metadatahøsteren og søkegrensesnittet skal brukes. Brukerveiledningen for søkegrensesnittet skal lages både for papir og web
- Dokumentasjon av systemets tekniske oppbygning. All programkode skal kommenteres, slik at det for eksempel kan genereres Javadoc
- UML-diagrammer som illustrerer designet av de viktigste systemkomponentene

4.4 Oppsummering

4.4.1 Forutsetninger og valg

- Metadata skal høstes fra Nasjonalbibliotekets samlinger. I dette prosjektet benyttes tre utvalgte samlinger:
  - Galleri NOR
  - Digitalt Radioarkiv (DRA)
  - Mavis
- Funksjonalitet som OAI datatilbydere for disse datakildene skal utvikles av Nasjonalbiblioteket
- Fast Data Search v3.2 skal brukes som indekserings- og søkesystem

4.4.2 Krav

- Fokus på brukerne av søkesystemet, ikke administratorer
- MODS skal benyttes som felles metadataformat hos OAI datatilbyderne
- OAI-høster:
  - OAI-høsteren skal baseres på applikasjonen OAIHarvester2
  - OAI-høsteren skal utvides slik at den kan omforme innhøstede metadata til andre formater
  - Innhøstede metadata skal i dette prosjektet omformes fra MODS til Fast Data Search sitt indekseringsformat FastXML
- Omformingsmalen skal defineres i et xslt-stilark
- En indeksprofil skal definere indeksstrukturen i Fast Data Search
- Et mellomvarelag skal styre kommuniksjonen mellom brukergrensesnitt og Fast Data Search
- Søkegrensesnitt:
  - Et webbasert søkegrensesnitt skal muliggjøre søking i de innhøstede, indekserte metadataene
  - Enkelt søk skal gjøre det mulig å skrive inn ett eller flere søkeord i ett søkefelt
  - Avansert søk skal gjøre det mulig å søke i bestemte felt eller kombinasjoner av flere angitte felt
  - Søkeresultatene skal presenteres på en oversiktlig måte i listeform

KAPITTEL 5

Prototyp

En prototyp er et systemforslag som viser at det er mulig å realisere en løsningsarkitektur i praksis, og som dermed gjør det mulig å evaluere sterke og svake sider ved en løsning. Hensikten med prototypen i dette prosjektet er hovedsakelig tredelt:

- Å vise at det går an å lage en søketjeneste basert på metadata innhøstet ved hjelp av OAI-PMH
- Å undersøke hvordan Fast Data Search fungerer som indekserings- og søkeløsning når konteksten er digitale bibliotek.
- Å undersøke hvordan MODS er egnet som felles metadataformat

For å kunne implementere en prototyp, kreves først en detaljert systemdesign basert på spesifikasjonen i kap. 4. Utviklingsprosessen i dette prosjektet har foregått trinnvis i henhold til spiralmodellen, det vil si med en vekselvirkning mellom utviklingsfasene design, implementasjon og utprøving/testing. I dette kapittelet beskrives:

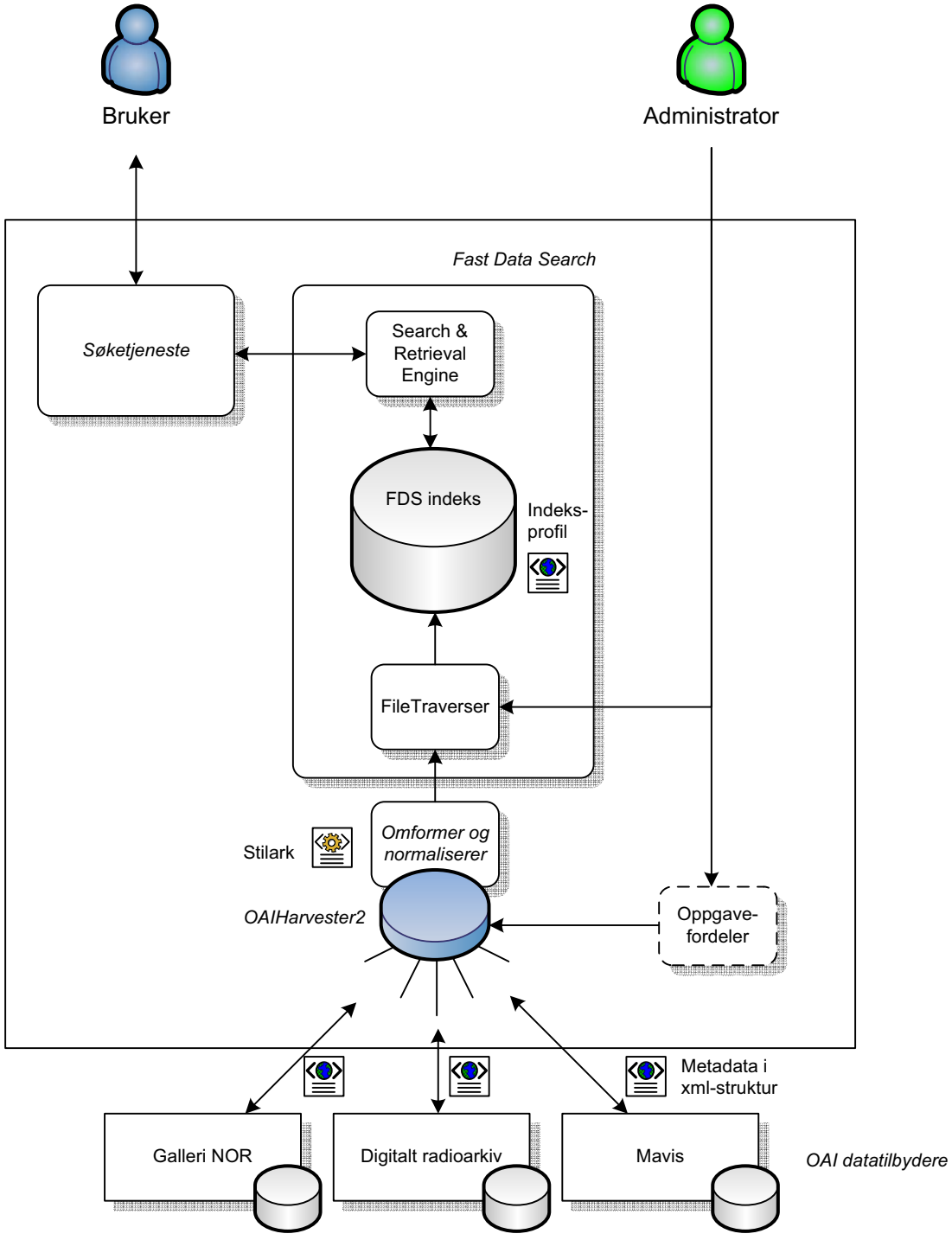
- Hovedtrekkene i systemets design
- Prototypen på systemet

5.1 Systemoversikt

Systemspesifikasjonen (se *kap. 4.3, s. 55*) danner utgangspunktet for systemets design. Dette er en arkitekttegning av de ulike komponentene som til sammen utgjør et helhetlig system.

De viktigste delene av systemet er:

- OAI datatilbydere
- OAIHarvester2
- Metadataomforming
- Fast Data Search
- Søketjeneste



Figur 5-1. *Systemoversikt*

I dette kapittelet beskrives de ulike komponentene i denne rekkefølgen siden den representerer den logiske hendelseskjeden de inngår i systemet på. Det er i denne sammenhengen viktig å klargjøre at det foregår to hovedprosesser i systemet, slik det ble nevnt i systemspesifikasjonen (se *kap. 4.3, s. 55*):

1. Metadatahøsting for å etablere et datagrunnlag i den sentrale indeksen
2. Søking i de indekserte metadataene

Den første prosessen, som innebærer innhøsting og indeksering av metadata, kan i dette prosjektet betraktes som en engangsforeteelse som igangsettes manuelt av en systemadministrator. Gjennomføringen av denne prosessen er en forutsetning for at den andre prosessen skal kunne finne sted.

Den andre prosessen, søking i de indekserte metadataene, skjer gjentatte ganger og igangsettes av brukerne av systemet. Sett fra et brukerperspektiv er det kun denne søkeprosessen som er synlig, og den forutsetter ingen forståelse av hvilke komponenter og prosesser som ligger til grunn for søketjenesten. Dette understreker påstanden i *[NEL02]* om at OAI-PMH er mellomvare som ikke skal synes for brukerne når den anvendes ordentlig.

Sammenlignet med den generelle arkitekturen for en OAI-PMH-basert tjenestetilbyder, (se *kap. 3.1.1, s. 33*) er følgende systemkomponenter utelatt i dette systemet:

- Duplikatsjekker: Ettersom systemet skal høste metadata fra et begrenset utvalg av datakilder som Nasjonalbiblioteket selv har kontroll over, er det på forhånd mulig å fastslå at problemet med duplikate metadatabeskrivelser ikke vil oppstå i dette systemet
- Oppdateringsmekanisme

I tillegg er det verdt å legge merke til at normaliseringskomponenten er inkludert i OAI-høsteren, og at anvendelsen av Fast Data Search fører til en litt annerledes intern struktur enn ved bruk av en lokal database.

5.2 OAI datatilbydere: Metadataavbildning

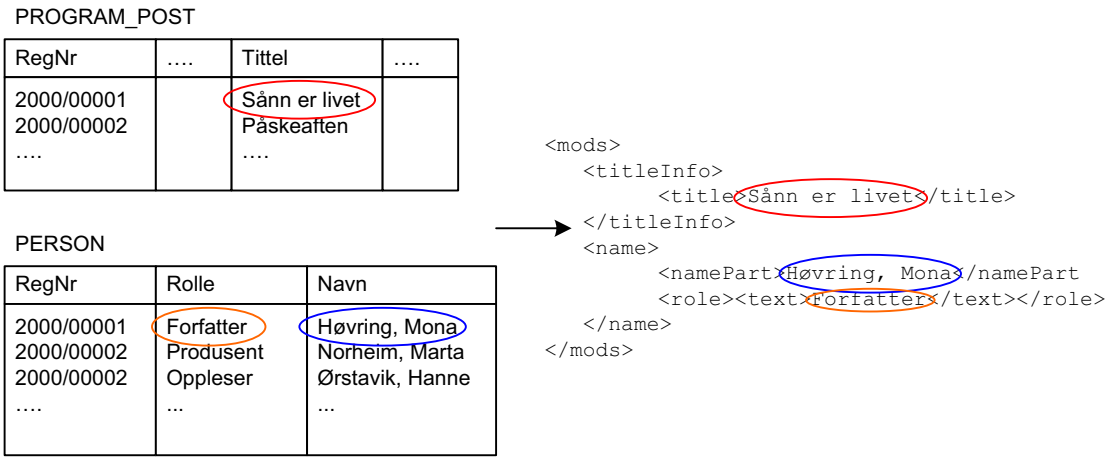
I prototypen høstes metadata fra følgende metadatabaser:

- Digitalt Radioarkiv
- Galleri NOR
- Mavis

I henhold til systemspesifikasjonen inngår ikke design og implementasjon av OAI datatilbydere som en direkte del av dette prosjektet. Unntaket er spesifiseringen av et felles metadataformat - MODS - for alle datatilbyderne, samt design av avbildninger (mappinger) fra det interne formatet i hver enkelt datakilde til fellesformatet. Slike semantiske avbildninger ble i *kap. 2.5.1, s. 23* beskrevet som en forutsetning for å kunne gjennomføre metadatahøsting.

Alle metadatabasene som fungerer som OAI datatilbydere i dette prosjektet, er i realiteten relasjonsdatabaser med individuelle interne strukturer, hvor metadataene logisk sett er lagret i tabeller. Ved hjelp av en OAI datatilbyderapplikasjon fungerer imidlertid hver enkelt metadatabase også som et OAI datalager som kan tilby strukturerte metadata. Forutsetningen for denne funksjonaliteten er at det er designet individuelle metadataoverganger fra databasenes tabellstrukturer til MODS-formatets xml-struktur.

I praksis gjennomføres selve avbildningen i sanntid ("on-the-fly") internt i hver enkelt OAI datatilbyderapplikasjon ved innkommende OAI-henvendelser fra en OAI-høster. Eksempelet illustrerer hvordan feltene *Tittel*, *Rolle* og *Navn* avbildes fra den interne tabellstrukturen i databasen for Digitalt Radioarkiv (DRA) til MODS-elementer i xml-struktur.



Figur 5-2. *OAI datatilbyderne konverterer metadata fra intern tabellstruktur i relasjonsdatabase til xml-strukturen i MODS ved OAI-høsting*

En fullstendig oversikt over avbildningene mellom de utvalgte metadatabasenes interne struktur og MODS finnes i *appendiks A.4, s. 117*

5.3 OAIHarvester2: Utvidelse for omforming av innhøstede metadata

I behovsanalysen av OAI-høstere i *kap. 4.2.4, s. 54* ble det bestemt at OAIHarvester2 *[OAIH]* skal være utgangspunkt for OAI-høsteren i denne prototypen. OAIHarvester2 inkluderer både en fungerende høsterapplikasjon og en samling Java-klasser som gjør at høsterfunksjonalitet kan bygges inn andre applikasjoner.

Den eksisterende høsterapplikasjonen i OAIHarvester2 inneholder all den grunnleggende funksjonaliteten som behøves for å kunne høste metadata fra OAI datatilbydere. I denne prototypen er derfor den viktigste oppgaven å designe og implementere en utvidelse, slik at høsteren ved hjelp

av stilark (xslt) kan omforme de innhøstede metadataene til formatet FastXML. Dette formatet er nødvendig for å kunne indeksere metadataene i Fast Data Search. Utvidelsen av høsteren gjør det dessuten mulig å gjøre en enkel normalisering av metadataene som en del av omformingsfasen.

5.3.1 Omvendt utvikling (reverse engineering)

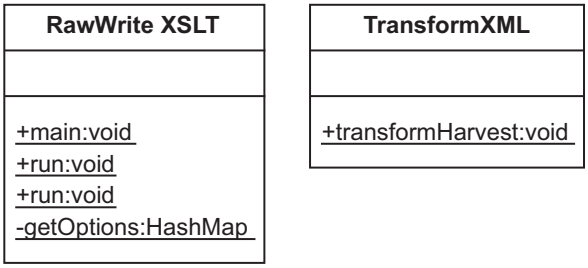
For å få oversikt over det eksisterende designet til applikasjonen OAIHarvester2, ble det gjennomført en omvendt utvikling hvor det ble generert en UML-modell ut fra den eksisterende kildekoden. Hele UML-modellen er beskrevet i *appendiks B, s. 123*. Hovedtrekkene i det eksisterende designet av OAIHarvester2 er:

- Applikasjonsklassen RawWrite er selve OAI-høsteren
- Pakken verb inneholder en generell superklasse, HarvesterVerb, som inneholder funksjonalitet som er felles for alle verbene, samt egne klasser for hvert av de seks OAI-verbene:
  - Identify
  - GetRecord
  - ListSets
  - ListRecords
  - ListIdentifiers
  - ListMetadataFormats
- RawWrite benytter de ulike verb-klassene til å gjøre de ønskede OAI-henvendelsene til datatilbydere

5.3.2 Redesign

Selv om OAIHarvester2 fungerer som en selvstendig høsterapplikasjon, har den imidlertid ikke innebygd støtte for omforming av de innhøstede metadataene til andre formater. I dette prosjektet er det, som nevnt, behov for å omforme de innhøstede metadataene til Fast Data Search sitt interne dataformat FastXML.

For å oppnå den ønskede funksjonaliteten, utvides OAIHarvester2 med en ny klasse, `TransformXML`, som plasseres sammen med applikasjonsklassen `RawWrite`.



Figur 5-3. *UML klassediagram for utvidet OAIHarvester2*

Prinsipper i den nye klassen:

- **Teknologi.** XSLT (Extensible Stylesheet Language Transformations)
- **Utgangspunkt.** Innhøstede metadata som er lagret i en xml-fil
- **Metode.** Eksternt stilark (xslt-fil) definerer maler for hvordan omformingen skal foregå
- **Resultat.** De omformede metadataene lagres i en ny xml-fil

I tillegg utvides applikasjonsklassen `RawWrite` slik at den ut fra inputparameterene sjekker hvorvidt en omforming skal gjøres. Hvis dette er tilfelle, skal omformingen av de innhøstede dataene som er lagret i en xml-fil utføres ved hjelp av metodene i den nye klassen `TransformXML`. Den utvidete høsterapplikasjonen har fått navnet `RawWrite_XSLT`.

Selv om utvidelsen av OAIHarvester2 i dette prosjektet gjøres med tanke på å omforme innhøstede metadata til et format som Fast Data Search kan forstå, er utvidelsen av generell karakter. Dette vil med andre ord si at innhøstede metadata kan omformes til et hvilket som helst annet xml-format ved å anvende andre stilark.

5.3.3 Anvendelse

Høsterapplikasjonen OAIHarvester2 utgjør en grunnleggende del av systemet, og er en forutsetning for å etableringen av et datagrunnlag i den sentrale indeksen, som igjen er en forutsetning for et felles søkegrensesnitt.

OAIHarvester2 er implementert i Java, og startes manuelt av systemadministrator fra kommandolinjen.:

```
java -classpath "log4j.jar;harvester2.jar"
     harvester2.app.RawWrite_XSLT
     -out dra.xml
     -metadataPrefix mods
     -xslt FastXML.xsl
     http://nwa1.nb.no:8080/drarepository/servlet/OAIHandler
```

De forskjellige inputparametrene spesifiserer høsterens virkemåte:

- out: navn på filen som inneholder de innhøstede metadataene
- metadataPrefix: angir hvilket format de innhøstede metadatapostene skal være i
- xslt: navn på filen som inneholder et stilark som definerer omformingen av de innhøstede metadataene
- URL: hvilken OAI datatilbyder det skal høste metadata fra, i dette tilfellet DRA.

Endringen av OAIHarvester2, som gjør at den kan omforme innhøstede metadata, innebærer en mulighet for å oppgi parameteret `-xslt` med en tilhørende fil som definerer metadataomformingen.

Når høsteren startes opp på denne måten, kontakter den angitt datatilbyder med OAI-verbet *Identify*. Hvis den får svar, foregår det deretter en kommunikasjon mellom høster og datatilbyder hvor høsteren kontakter datatilbyderen med OAI-verbene *ListMetadataFormats*, *ListSets* og *ListRecords*. Denne kommunikasjonen er illustrert i *figur 2-3, s. 30*.

Resultatet etter at høstingen er ferdig er en xml-fil som i tillegg til selve metadataene også inneholder alle detaljene om datatilbyderen. På grunn av utvidelsen av OAIHarvester2 som er gjort i dette prosjektet, omformer i tillegg høsteren de innhøstede metadataene til formatet FastXML. De omformede metadataene lagres i en egen xml-fil.

Når høsteren for eksempel høster metadata fra Digitalt Radioarkiv (DRA), blir de innhøstede metadataene først lagret i xml-filen `dra.xml`. Deretter starter omformingen av denne filen i henhold til omformingsreglene beskrevet i stilarket `FastXML.xslt`. Resultatet er en ny xml-fil med navnet `dra_FastXML.xml`, som inneholder de omformede metadataene.

Underveis gis det tilbakemeldinger på skjerm om hvilken operasjon som pågår.

Siden det blir høstet metadata fra flere OAI datatilbydere, er det laget et skript som definerer høsterens oppførsel for de forskjellige datatilbyderne. På denne måten kan alle metadataene høstes ved å starte dette ene skriptet istedenfor å måtte starte høsteren manuelt for hver enkelt datatilbyder. Flere detaljer om høsterskriptet finnes i *appendiks B.5, s. 131*.

5.4 Metadataomforming

Utvidelsen av OAI-høsteren gjøres, som tidligere nevnt, for å kunne omforme de innhøstede metadataene til et format som kan indekseres i Fast Data Search. En forutsetning for at en slik omforming skal kunne finne sted, er at det eksisterer et stilark som definerer reglene for omforming av metadataene. Nedenfor beskrives designprinsippene for hvordan et stilark kan brukes til å:

- Omforme metadata fra én struktur til en annen
- Normalisere (homogenisere) innholdsvariasjoner i bestemte metadataelementer

Alle detaljer i tilknytning til omforming av metadata er beskrevet i *appendiks C, s. 133*.

5.4.1 Strukturomforming fra MODS til FastXML

Prinsippet i ethvert xslt-stilark er at man definerer maler som beskriver hvordan én xml-datastruktur kan omformes til en annen. I dette prosjektet anvendes denne teknikken til å omforme de innhøstede metadataene fra formatet MODS (se *kap. 2.2.4, s. 13*) til det interne formatet i Fast Data Search, FastXML (se *kap. 4.1.3, s. 48*).

Den overordnede strukturen i stilarket er:

- Én hovedmal identifiserer strukturen i hele det opprinnelige metadatadokumentet, som kan inneholde mange metadatabeskrivelser
- Én metadatamal idenfiserer hver enkelt metadatabeskrivelse, som inneholder mange metadataelementer
- For hvert enkelt metadataelement, for eksempel *titleInfo* eller *abstract*, eksisterer det en egen elementmal som definerer hvordan det aktuelle elementet skal omformes
- For metadataelementer som inneholder subelementer, for eksempel *subject* med subelementer som *topic, geographic* og *temporal*, må det utformes elementmaler som identifiserer hvert av subelementene, og som definerer hvordan disse skal omformes

Metadataomformingen, som utføres av OAIHarvester2, skjer i henhold til omformingsmalene som er beskrevet i xslt-stilarket FastXML.xsl (se *appendiks C, s. 133*). Her defineres både strukturomforming fra MODS til FastXML og innholdsnormalisering av datoer.

Eksempelet nedenfor er et lite utdrag av en metadatabeskrivelse som illustrerer strukturforandringen fra MODS til FastXML:

```
<mods>
     <abstract>Prot: Hamar - Vangs kirke 6. Sep. 1902</abstract>
</mods>
```

omformes til

```
<document>
     <element name="mods.abstract">
          <value>Prot: Hamar - Vangs kirke 6. Sep. 1902</value>
     </element>
</document>
```

5.4.2 Innholdsomforming: Normalisering av datoer

I oppsummeringen av OAI-PMH-baserte søketjenester (se *kap. 3.7, s. 44*) kommer det fram at ulike datakilder har ulike måter å formatere spesifikke metadatafelt, for eksempel dato. En forundersøkelse av metadata fra de tre datatilbyderne i dette prosjektet gir det samme resultatet: Det finnes forskjellige typer datoer både internt i hver enkelt datakilde og mellom de forskjellige datakildene. Dette nødvendiggjør datanormalisering til en enhetlig form.

Tabell 5-1. *Ulike datoformater i de originale metadatabasene*

Datoformat	Antall tegn	Metadatabase	Tolkning
1980	4	DRA, Mavis	1980
194006	6	DRA	juni 1940[a]
000000	6	DRA	ingen registrert dato
1098911	7	DRA	1980-9-11[b]
1980-?	6	Mavis	fra 1980 til ukjent dato
1984-01-01	10	DRA, Galleri NOR	1. januar 1984
Ca.	4	Mavis	ingen registrert dato
Ca. 1980	8	Mavis	ca. 1980

a. Usikker.
b. Usikker. Den opprinnelige registreringen 1098911 ser ut til å inneholde feil.

På samme måte som stilark brukes på det strukturelle planet til å omforme metadata fra én struktur (MODS) til en annen (FastXML), benyttes også teknikken på det innholdsmessige planet til å omforme forskjellige typer datoer til ett felles heltallsformat. Målet er at alle former for datoer skal konverteres til det indekserbare tallformatet *ååååmmdd*, slik at for eksempel *16. februar 2004* blir omformet til *20040216*. Hensikten er å gjøre de ulike datoene fra de forskjellige metadatabasene søkbare. Den indekserte datoen i heltallsform benyttes imidlertid bare internt i Fast Data Search under selve søkeprosessen. Når søkeresultatene presenteres for brukeren, blir den opprinnelige datoen i tekstform benyttet, for eksempel *Ca. 1980* eller *1995-01-01*.

Prinsippet i konverteringsrutinen er at lengden på den opprinnelige verdien i datofeltet kan brukes til å avgjøre hvilket format den originale datoen har, og dermed bestemme hvilken konverteringsrutine som skal brukes for å konvertere den til fellesformatet *ååååmmdd*. Dersom det eksisterer forskjellige typer datoer med likt antall tegn, utføres det en tegnsjekk for å skille mellom de ulike datotypene. Datoen *2003* inneholder for eksempel fire tegn, mens *1997-03-15* inneholder ti tegn. Etter konvertering blir disse datoene henholdsvis *20030000* og *19970315*, som begge kan indekseres som heltall.

Eksempelet som følger illustrerer omformingen av en dato:

```
<mods>
     <originInfo>
          <dateCreated>1995-01-01</dateCreated>
     </originInfo>
</mods>
```

omformes til fellesformatet *ååååmmdd*:

```
<document>
     <element name="dateinteger">
          <value>19950101</value>
     </element>
</document>
```

Et flytskjema som illustrerer hele prosessen for omforming av datoer finnes i *appendiks C, s. 133*.

5.5 Fast Data Search

Fast Data Search (se *kap. 4.1.3, s. 48*) er et komplett indekserings- og søkesystem som kun krever enkelte lokale tilpasninger for å kunne integreres i prototypen. Før metadataene i FastXML-format kan indekseres i Fast Data Search, må indekseringsprosessen først konfigureres i det webbaserte brukergrensesnittet til Fast Data Search. Konfigurasjonen består av følgende to trinn:

1. Oppdatere indeksprofil, slik at indekseringen skjer i henhold til indeksprofilen som er definert for denne prototypen.
2. Opprette ny samling som alle metadataene skal indekseres i.

Når disse trinnene er utført, starter systemadministrator applikasjonen FileTraverser, som gjennomgår alle metadatafilene i FastXML-format som skal indekseres, og videresender innholdet til indeksering i angitt samling (collection) i Fast Data Search. I denne prototypen eksisterer det én FastXML-fil for hver metadatabase, det vil si totalt tre filer. Etter at metadataene er indeksert i Fast Data Search, kan de søkes i ved hjelp av den innebygde Search and Retrieval Engine.

I henhold til kravspesifikasjonen (se *kap. 4.3, s. 55*) er det bare enkelte metadatafelt som er interessante å indeksere med tanke på søking i denne prototypen. Spesifiseringen av hvilke felt som skal være mulig å søke i, og hvilke felt som bare skal vises i søkeresultatene gjøres i en indeksprofil. Dette er en xml-fil som uttrykker sammenhengen mellom metadataelementer og søkbare felt. Eksempel på en indeksprofil-definisjon:

```
<field
     name="modstitle"
     element-name="mods.title"
     index="yes"
     wildcard="yes"
/>
```

Dette betyr at det lages et indeksfelt som heter `modstitle`, og at innholdet i ethvert element av typen `mods.title` blir indeksert som et søkbart felt i Fast Data Search. Dette er en forutsetning for at en bruker skal kunne søke etter bestemte titler. Attributtet `wildcard` innebærer at det er støtte for trunkering (*) i søkeuttrykk som sendes til det aktuelle feltet.

Indeksprofilen brukes også til å spesifisere sammensatte felt, som gjør det mulig å sende en spørring til ett sammensatt felt istedenfor flere enkeltfelt. I denne prototypen defineres det for eksempel et sammensatt felt, *simplesearch*, som inneholder referanse til alle de enkle feltene *tittel, navn, utgiver, subject* og *abstract*.

Alle detaljer omkring indeksprofilen i dette prosjektet er beskrevet i *appendiks D, s. 143*.

5.6 Søketjenesten: Søkegrensesnitt og mellomvare

Systemets søketjeneste består av to hoveddeler:

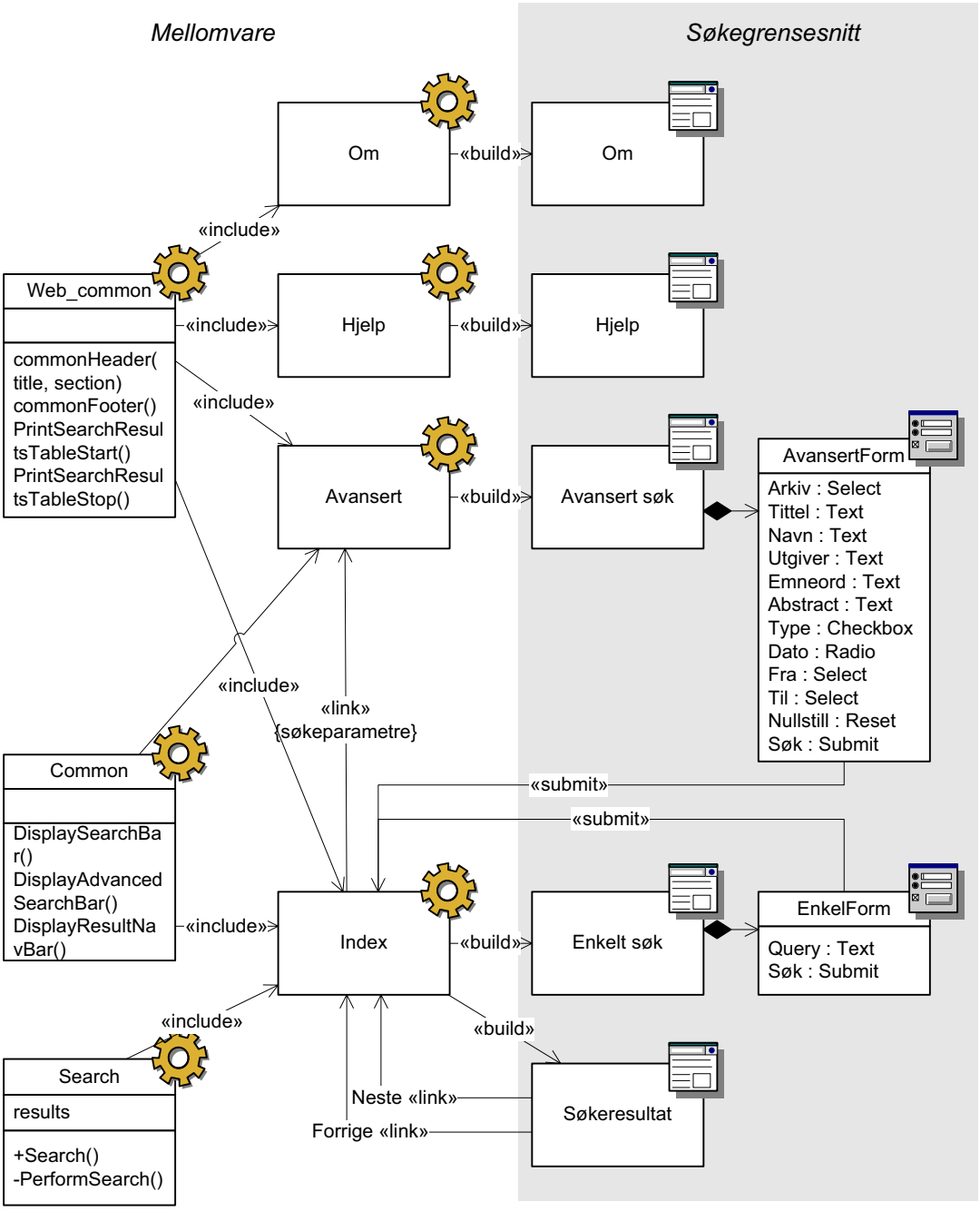
1. Søkegrensesnitt
2. Mellomvarelag som sørger for samhandling mellom søkegrensesnittet og Fast Data Search.

Det webbaserte søkegrensesnittet utgjør kontaktpunktet mellom brukeren og systemet. Søkegrensesnittet bygges over et mellomvarelag som tar seg av kommunikasjon med søkemotoren i Fast Data Search. Designet i modellen er uttrykt i henhold til Web Application Extension for UML *[CON02]*, en utvidelse av UML som gjør det mulig å modellere både statiske og dynamiske websider i UML.

Mellomvarelaget er utvilket i skriptspråket php, mens søkegrensesnittet er en kombinasjon av php, html og css.

Tabell 5-2. *Sentrale begreper i UML Web Application Extension*

Begrep	Symbol	Forklaring
Serverside		Dynamisk webside med skriptkode
Klientside		HTML-side
HTML-skjema		Skjema i klientside



Figur 5-4. *Logisk UML-modell av søketjenestens søkegrensesnitt og mellomvarelag*

Søkegrensesnittet består av følgende websider (klientsider):

- Om
- Hjelp
- Avansert søk (inneholder avansert søkeskjema)
- Enkelt søk (inneholder enkelt søkeskjema)
- Søkeresultat (vises etter at brukeren har gjennomført et søk)

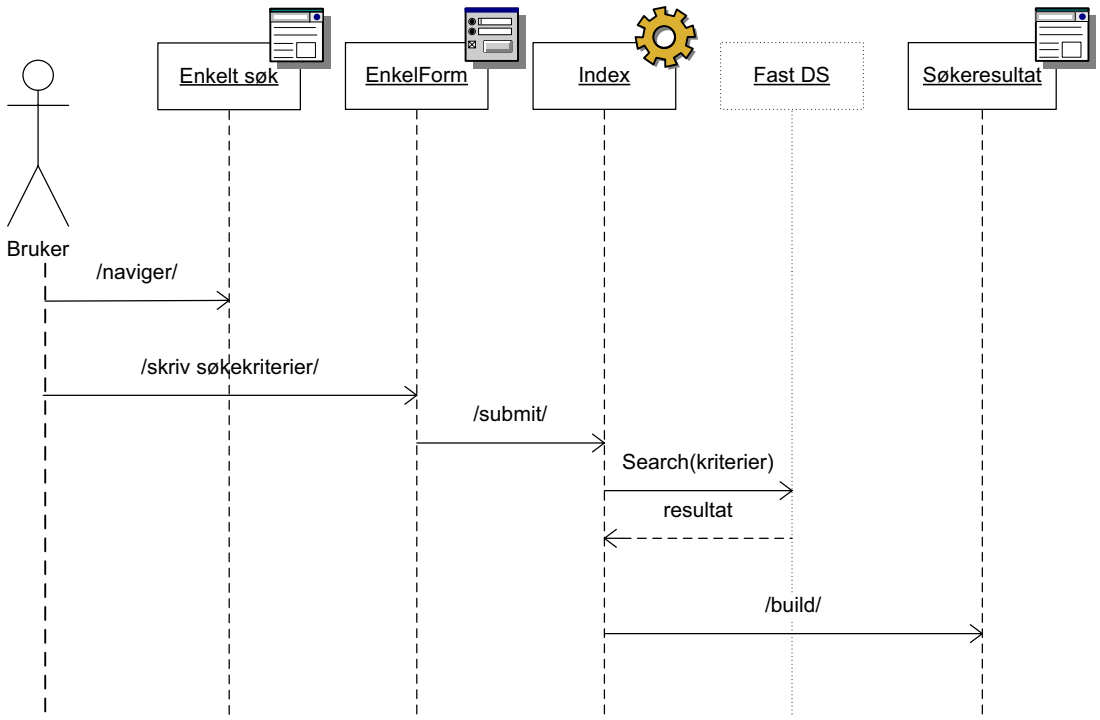
Som modellen viser, genereres hver av websidene i søkegrensesnittet dynamisk (stereotype "build") ut fra en tilhørende serverside i mellomvarelaget. Det er her verdt å legge merke til at serversiden *Index* både genererer websiden for enkelt søk og eventuelle websider som inneholder søkeresultater.

Serversidene *Web_common*, *Common* og *Search* innehoder metoder som er felles for de serversidene som genererer søkegrensesnittet. Ved å skille disse metodene ut i egne serversider, kan de gjenbrukes på en enkel måte. For å kunne nyttiggjøre disse felles metodene, inkluderes (stereotype "include") disse serversidene der de er nødvendige.

Siden det er et søkesystem som er utviklet, er det viktig å beskrive virkemåten til enkelt og avansert søk nærmere. Websidene for enkelt og avansert søk inneholder henholdsvis et enkelt og et avansert søkeskjema. Når en bruker skriver inn et søkeuttrykk - enten i det enkle eller i det avanserte søkeskjemaet - og starter søkeprosessen, formidles søkekriteriene fra det respektive søkeskjemaet til serversiden *Index*. Index bygger opp en spørring på bakgrunn av kriteriene i søkeskjemaet, og igangsetter søket i Fast Data Search (ikke tatt med i figur) ved hjelp av den inkluderte metoden *Search()*.

Search() er en sammensatt metode som både utfører selve søket og som genererer websidene med søkeresultater. Hvis søkeresultatet inneholder mer enn et visst antall treff, fordeles søkeresultatene over flere websider. Dette er årsaken til at det er pekere (stereotype "link") til neste og forrige søkeresultatside.

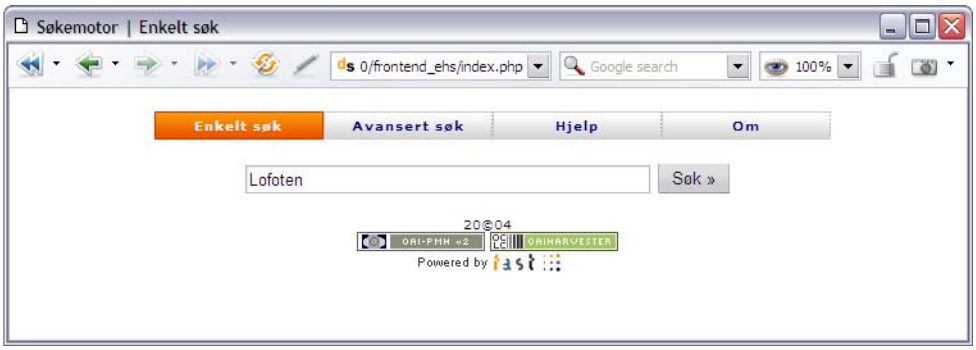
Rekkefølgen i søkeprosessen er illustrert i *figur 5-5, s. 76*.



Figur 5-5. *UML sekvensdiagram for søkeprosessen.*

5.6.1 Enkelt søk

Enkelt søk består kun av ett søkefelt hvor brukeren kan skrive inn sine søkeord (én eller flere søketermer). En søk-knapp brukes for å starte søket. Søkegrensesnittet minner om enkelt søk i de generelle websøkemotorene Google og AllTheWeb.



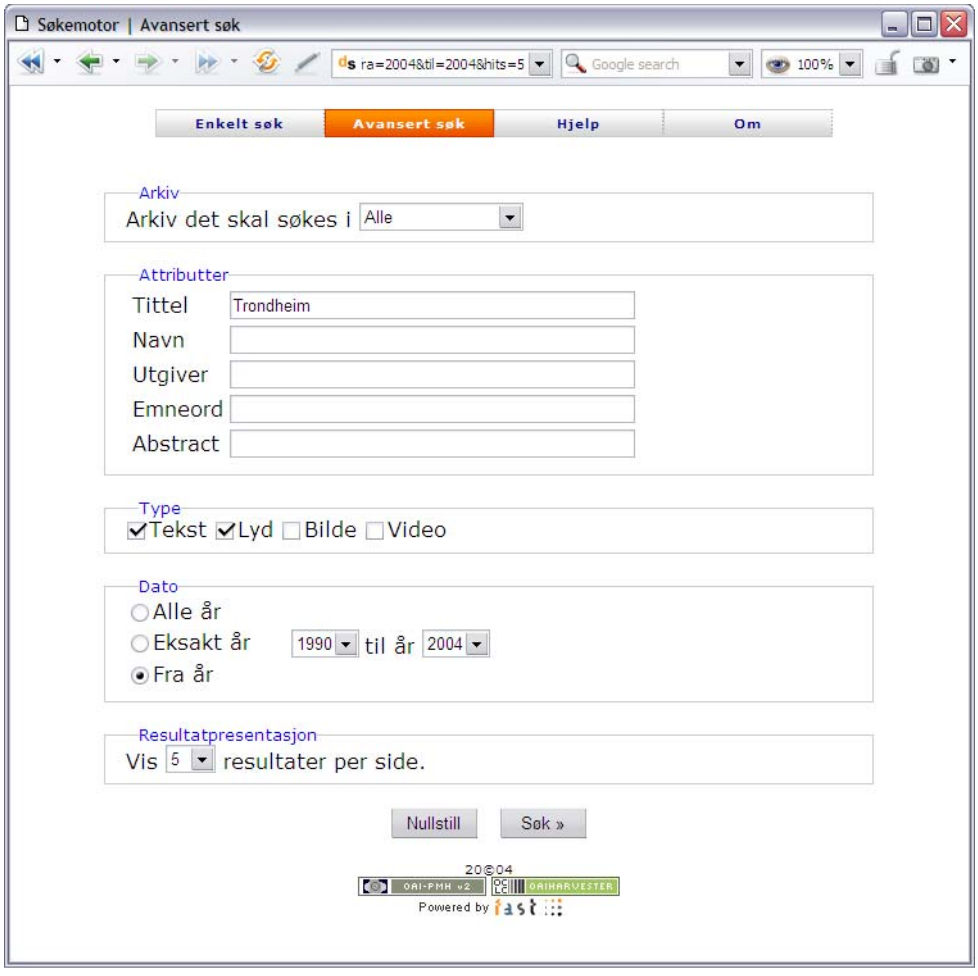
Figur 5-6. *Enkelt søkegrensesnitt.*

Hensikten med enkelt søk er at søkeren på en ukomplisert måte skal kunne søke etter det han/hun leter etter. Et enkelt søkegrensesnitt stiller minimale krav til erfaring hos brukeren, og gjør det derfor mulig for alle typer brukere å søke etter informasjon.

Som nevnt i *kap. 5.5, s. 72* definerer indeksprofilen et sammensatt felt – simplesearch – som består av feltene tittel, navn, utgiver, emneord og abstract. Når brukeren formulerer en spørring i enkelt søk, sender mellomvarelaget denne spørringen direkte til dette sammensatte feltet i Fast Data Search. Dette resulterer i praksis i at det søkes i alle feltene som inngår i det sammensatte feltet.

5.6.2 Avansert søk

Avansert søk reflekterer strukturen i søkeindeksen i langt større grad enn hva enkelt søk gjør, og gjør det dermed mulig for brukeren å spesifisere mer detaljert hva han/hun ønsker å søke etter.



Figur 5-7. *Avansert søkegrensesnitt.*

I henhold til kravspesifikasjonen er følgende felt er tatt med i det avanserte søkegrensesnittet:

- Tittel
- Navn
- Utgiver
- Emneord
- Abstract

I tillegg kan brukeren velge hvilken type ressurser han/hun søker etter:

- Tekst
- Lyd
- Bilde
- Video

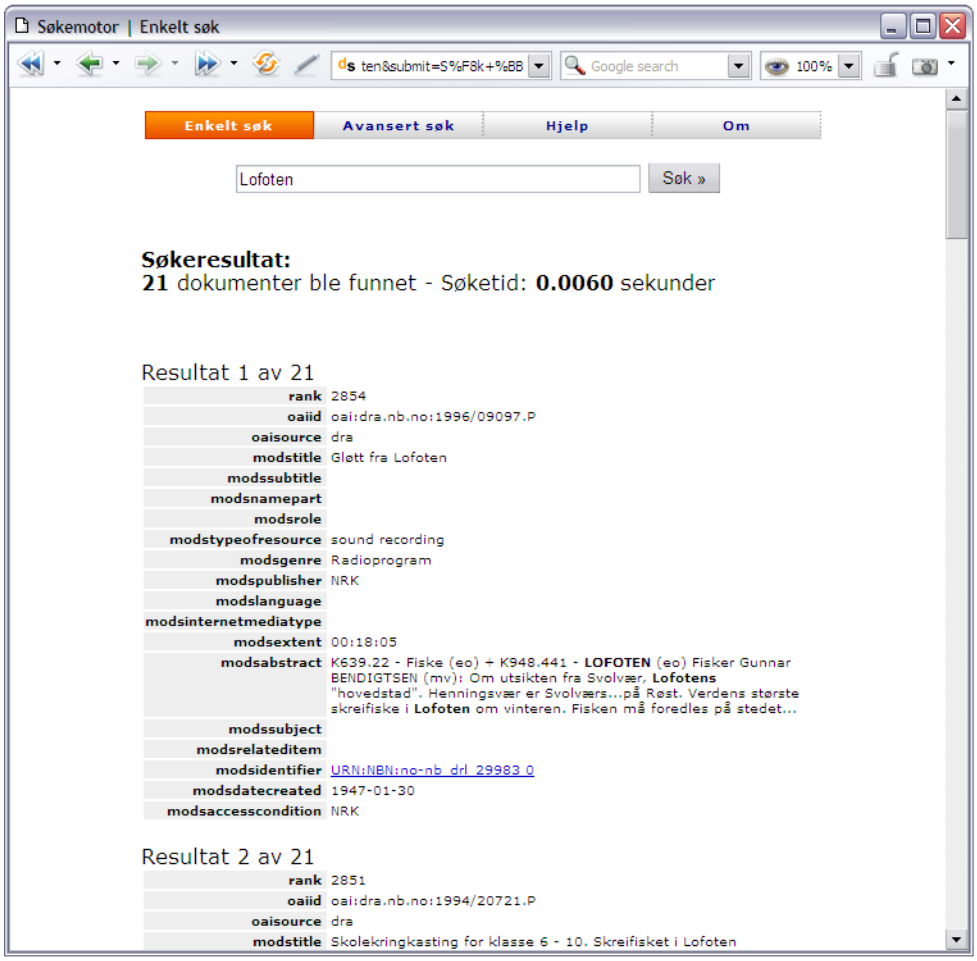
Brukeren har dessuten muligheter til å angi hvilket arkiv han/hun ønsker å søke i, og/eller søke etter bestemte årstall eller tidsperioder.

Når brukeren igangsetter et avansert søk, formulerer mellomvarelaget en spørring direkte til hvert enkelt av de aktuelle feltene som er definert i Fast Data Search. Dette innebærer at det for søketermer i søkefeltet *utgiver* søkes direkte i indeksfeltet `modspublisher`, mens det for søketermer i feltet *emneord* kun søkes i indeksfeltet `modssubject`.

I tillegg er det viktig å legge merke til at søkefeltene *arkiv* og *type* fungerer som avgrensninger for hvilke resultater som skal returneres, i motsetning til tradisjonelle søkefelt.

5.6.3 Søkeresultat

Resultatene presenteres i listeform, og hvert treff er en tabell med oversikt over indekserte metadataelementer og deres tilhørende verdi. Metadataelementer uten verdi vises som tomme tabellceller.



Figur 5-8. *Søkeresultatene presenteres i listeform.*

5.7 Oppsummering

Som beskrevet i *kap. 4.2.1, s. 50* kan prototypen som er utviklet i dette prosjektet betraktes fra to perspektiver:

1. Brukerperspektiv
2. Administratorperspektiv

Ettersom prototypen er utviklet med tanke på å gjøre informasjon lettere tilgjengelig for brukere, er det mest naturlig å betrakte den ut fra et brukerperspektiv. I dette perspektivet er det søketjenesten som utgjør systemet, siden denne gjør det mulig for brukerne å søke etter informasjon fra mange kilder på ett sted. Brukerne behøver med andre ord ikke vite noe om hva slags system som ligger til grunn for søketjenesten og/eller hvordan dette systemet er oppbygd.

Sett fra et administratorperspektiv utgjør derimot søketjenesten bare toppen av isfjellet. Under overflaten består systemet av forskjellige typer komponenter som jobber mot ett mål: Å gi brukeren mulighet til å søke i metadata fra mange forskjellige kilder. Hele systemet kan oppsummeres ved å illustrere de forskjellige overgangene metadataene gjennomgår, fra de befinner seg i de opprinnelige metadatabasene til de presenteres som et søkeresultat for brukeren.

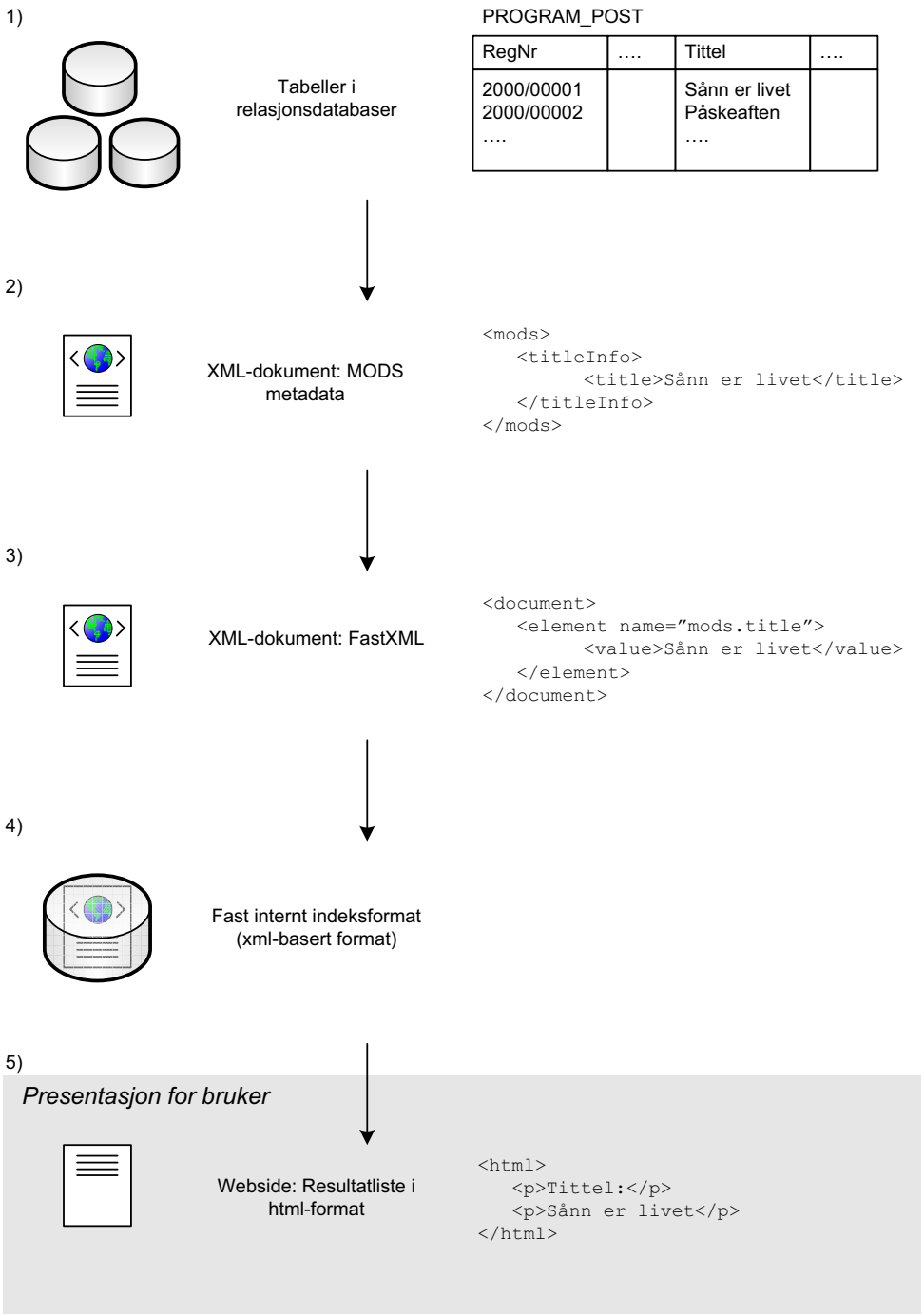
I trinn 1 eksisterer metadataene som felt i tabellene i de opprinnelige relasjonsdatabasene. Hver enkelt database har sin individuelle datastruktur.

I trinn 2 er metadataene avbildet til xml-strukturen i metadataformatet MODS. Denne avbildningen gjøres internt i hver enkelt OAI datatilbyderapplikasjon i henhold til forhåndsdefinerte metadataoverganger.

I trinn 3 omformes metadataene fra én xml-struktur, MODS, til en annen xml-struktur, FastXML. Denne omformingen gjøres av OAI-høsteren etter at metadataene er høstet inn. Omformingen utføres i henhold til regler definert i et stilark.

I trinn 4 omformes metadataene fra formatet FastXML til Fast Data Search sitt interne dokumentformat. Denne omformingen utføres av Fast Data Search som en del av indekseringsprosessen.

I trinn 5 presenteres metadataene som en del av søkeresultatene. Denne omformingen gjøres av mellomvarelaget som mottar søkeresultatene fra Fast Data Search.



Figur 5-9. *Metadataenes gjennomgår flere omforminger i systemet*

KAPITTEL 6

Testing og evaluering av søketjenesten

Hovedmålsetningen med dette prosjektet er å gjøre informasjon fra flere forskjellige samlinger lettere tilgjengelig for brukerne gjennom et enhetlig søkegrensesnitt (se *kap. 1.1, s. 1*). For å kunne avgjøre hvorvidt prototypen på systemet oppfyller denne målsetningen og de opprinnelige kravene som ble stilt, må det gjøres en testing og evaluering av søketjenesten. Den viktigste indikatoren i denne sammenhengen er hvorvidt søkeresultatene inneholder treff fra flere forskjellige datakilder.

Underveis i utviklingsprosessen har det blitt gjennomført testing av de ulike komponentene, men løsningen som helhet - slik den framstår for brukeren - er ikke testet tidligere. Dette kapittelet er derfor i første rekke en testing og evaluering av selve søketjenesten, gjort ut fra et brukerperspektiv. En nærmere evaluering av komponentene i det underliggende systemet finnes i *kapittel 7*.

I prototypen er totalt 1403 metadatabeskrivelser indeksert i Fast Data Search, og i dette kapittelet benyttes disse søkbare beskrivelsene som grunnlag for å teste og evaluere:

- Enkelt søk
- Avansert søk

Ettersom Fast Data Search benyttes som indekserings- og søkesystem, er søkefunksjonalitet som trunkeringsstøtte, frasesøk o.l. gitt på forhånd. Hovedpoenget med eksemplene i dette kapittelet er imidlertid å vise et utvalg søkesituasjoner som illustrerer målsetningen med prototypen - ikke å demonstrere alle tilgjengelige søkeegenskaper.

6.1 Enkelt søk

Hensikten med enkelt søk er at det skal være en lettfattelig måte for alle typer brukere å søke etter informasjon på. Selv om enkelt søk innebærer relativt begrensede valgmuligheter for brukeren, søkes det likevel i de metadatafeltene som normalt inneholder søkbare opplysninger. Som nevnt i

kap. 5.6.1, s. 76 innebærer enkelt søk at det søkes i metadatafeltene *tittel*, *navn*, *utgiver*, *emneord* og *abstract*.

Eksempel 1 (én søketerm)

Søketermen i det første eksempelet er et stedsnavn, *Eidsvold.*

Søket gir 17 treff som presenteres i resultatlisten (se *figur 6-1, s. 85*), og en nærmere undersøkelse av søketreffene viser at disse stammer fra forskjellige kilder:

- DRA: 4 treff
- Galleri NOR: 13 treff
- Mavis: 0 treff

Dette betyr at søketermen *Eidsvold* er funnet i metadatabeskrivelser både for radioinnslag (fra DRA) og bilder (fra Galleri NOR), og at termen dermed er en fellesnevner for alle søketreffene, som presenteres i én resultatliste. Siden søkeresultatene kun har denne ene fellesnevneren, er det verdt å legge merke til variasjonene mellom de to kildene. De fire søketreffene fra DRA gjelder krigsskipet Eidsvold under andre verdenskrig, mens søketreffene fra Galleri NOR gjelder forskjellige bygninger i Eidsvold.

Eksempel 2 (én søketerm)

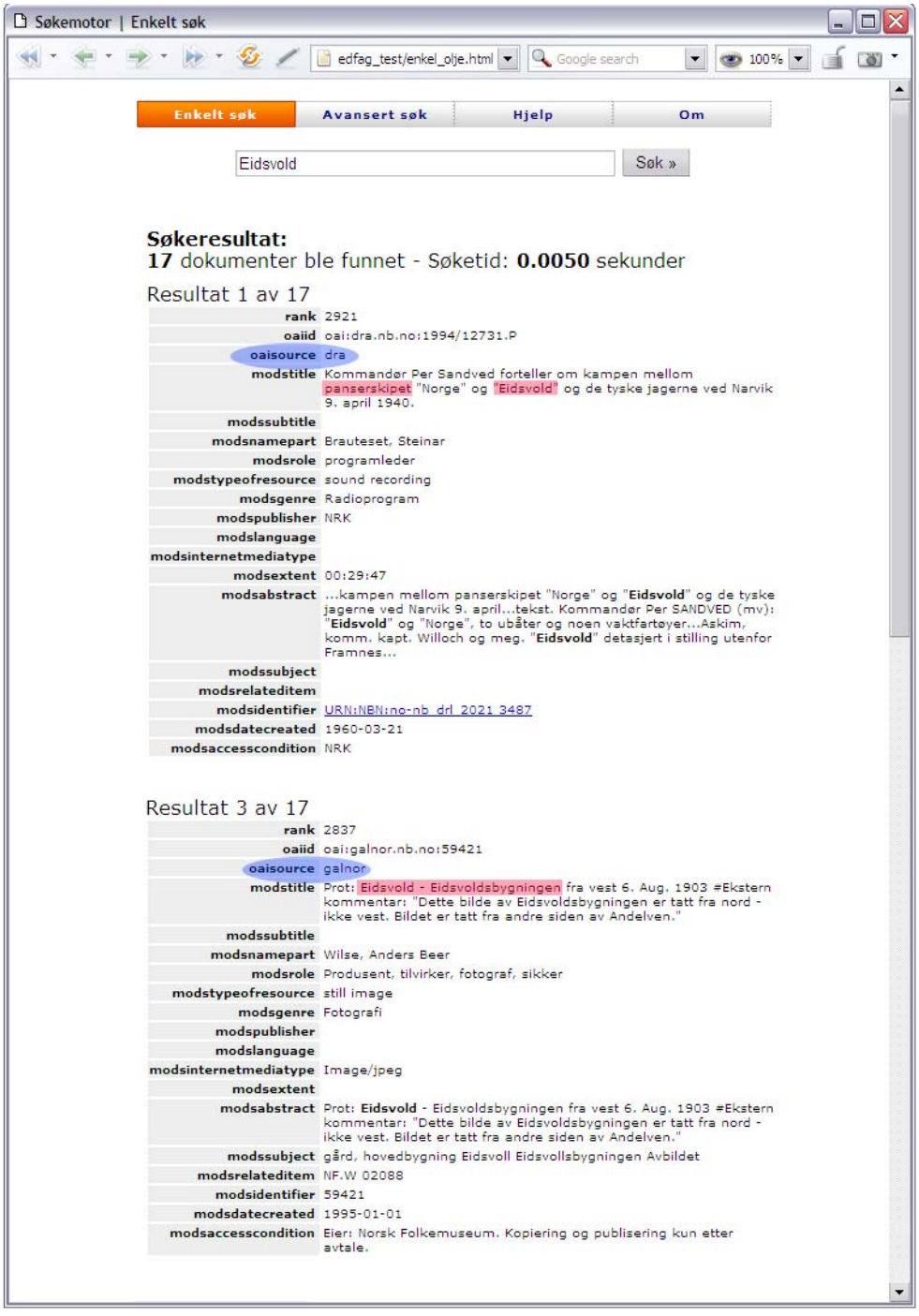
Søketermen i det andre eksempelet er et emneord, *olje*

Søket gir 3 treff som presenteres i resultatlisten (se *figur 6-2, s. 86*, opprinnelig datakilde er markert med blå farge), og en nærmere undersøkelse av søketreffene viser at disse stammer fra forskjellige kilder:

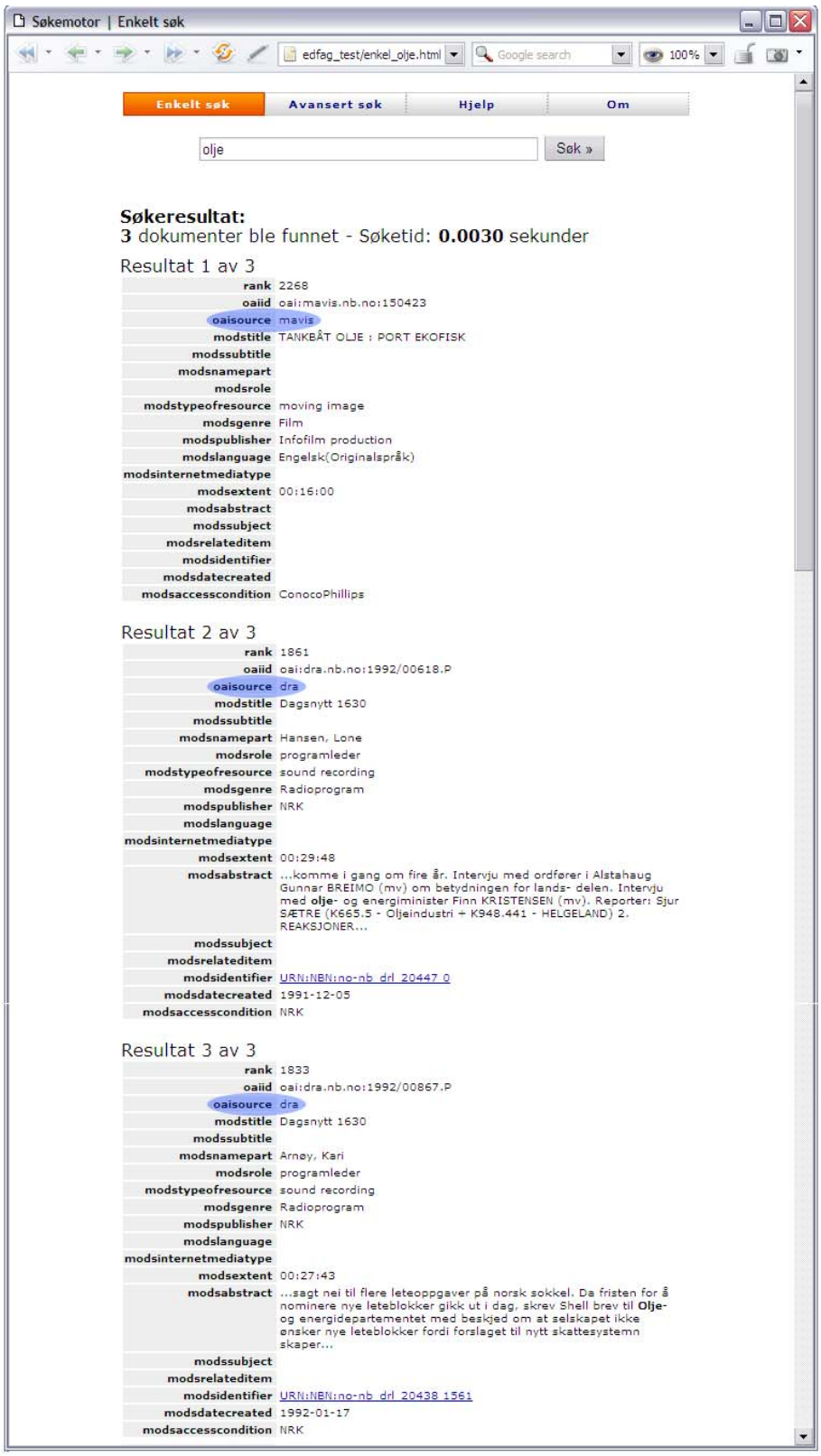
- DRA: 2 treff
- Galleri NOR: 0 treff
- Mavis: 1 treff

Som det kommer fram av resultatlisten på figuren, er det funnet metadatabeskrivelser både for film (fra Mavis) og radioinnslag (fra DRA). I metadatabeskrivelsen fra Mavis er søketermen olje funnet i tittelfeltet, mens metadatabeskrivelsene fra DRA inneholder søketermen olje i abstract-feltet. Dette er en bekreftelse på at det i et enkelt søk i realiteten søkes i flere enkeltfelt samtidig.

Sammenlignet med det første eksempelet, illustrerer dette eksempelet at det er mulig å benytte forskjellige kategorier søketermer, blant annet emneord og navn.



Figur 6-1. *Utdrag av søkeresultater i eksempel 1.*



Figur 6-2. *Søkeresultater eksempel 2*

Eksempel 3 (sammensatte søketermer)

Et eksempel på et søkeuttrykk med flere søketermer er *Eidsvold jernbane**

I dette søkeuttrykket er det verdt å legge merke til trunkeringstegnet (*), som gjør at alle ord som begynner med jernbane blir funnet, for eksempel jernban*en* eller jernbane*stasjon*. Støtte for trunkering inngår som en del av søkefunksjonaliteten i Fast Data Search.

Dette søket gir 4 treff som presenteres i resultatlisten, og en nærmere undersøkelse av søketreffene viser at disse stammer fra forskjellige kilder:

- DRA: 1 treff
- Galleri NOR: 3 treff
- Mavis: 0 treff

Sammenlignet med det første eksempelet med én søketerm, Eidsvold, er resultatlisten betydelig redusert. Årsaken er at søkeuttrykket inneholder flere termer, det vil si det eksisterer flere fellesnevnere mellom metadatabeskrivelsene som blir gjenfunnet. En undersøkelse av de fire søketreffene viser at disse er en delmengde av resultatene i det første eksempelet, og at de både inneholder termen Eidsvold og en eller annen form av jernbane.

Eksempel 4 (frasesøk)

I det enkle søkegrensesnittet er det også muligheter for frasesøk av typen *"Ski-VM i Trondheim 1997"*. Ettersom bare 1400 metadatabeskrivelser er indeksert i systemet, eksisterer det et begrenset datagrunnlag å søke i. Dette gjør det vanskelig å lage et eksempel på frasesøk som gir en resultatliste som illustrerer at et frasesøk kan gi søketreff i metadata som stammer fra forskjellige kilder. Med et større datagrunnlag ville det vært mulig å illustrere et enkelt frasesøk.

6.2 Avansert søk

Avansert søk gir brukeren langt mer detaljerte søkemuligheter enn enkelt søk, slik at han/hun i større grad enn ved enkelt søk kan spesifisere nøyaktig hva det skal søkes etter og i hvilke metadatafelt det skal søkes. I dette delkapittelet testes forskjellige måter å utføre avansert søk for å illustrere hvordan søkeresultatene påvirkes av de forskjellige søkeparameterene:

Søketermen *Eidsvold*, som i enkelt søk gav 17 treff, brukes også som utgangspunkt for testingen av det avanserte søkegrensesnittet. På denne måten er det mulig å illustrere både sammenhenger og forskjeller mellom enkelt og avansert søk.

Eksempel 1 (enkeltfelt)

Den enkleste testen av avansert søk går ut på å benytte ett av søkefeltene, *Tittel*, til én søketerm, *Eidsvold*.

Dette søket gir 15 treff som presenteres i resultatlisten, og i likhet med enkelt søk stammer søketreffene fra forskjellige kilder:

- DRA: 2 treff
- Galleri NOR: 13 treff
- Mavis: 0 treff

Med unntak av to færre søketreff, er resultatet identisk med søkeresultatet etter *Eidsvold* i enkelt søk. Resultatlisten inneholder med andre ord metadatabeskrivelser som opprinnelig kommer fra to forskjellige kilder. Årsaken til at to metadatabeskrivelser fra DRA ikke kommer med i søkeresultatene, er at de ikke inneholder termen Eidsvold i tittelfeltet.

Eksempel 2 (kombinasjoner av flere søkeparametere)

Dersom brukeren ønsker, kan han/hun kombinere så mange av de avanserte søkeparameterene som det er behov for. Jo flere søkeparametere som kombineres, jo mer presist resultatsett oppnår søkeren.

Kombinasjon av å søke etter *Eidsvold* i feltet *Abstract* med *DRA* som kilde gir 4 søketreff. Hvis dato i tillegg må være i intervallet *fra 1940 til 1960*, reduseres resultatet til 3 søketreff. Årsaken til dette er at én metadatabeskrivelse inneholder 1983 som registrert dato.

Datosøk er en type raffinering av søkeuttrykket som kan være nyttig for brukeren i enkelte sammenhenger, for eksempel hvis han/hun søker etter beskrivelser fra en bestemt tidsperiode eller ett bestemt år. Datosøk kan i denne prototypen enten brukes i kombinasjon med andre søkeparametere, eller det kan brukes som en selvstendig søkeparameter.

Hvis man søker etter *Eidsvold* i feltet *Abstract*, og datoavgrensningen er *fra 1945 til 1980*, det vil si at det kun er metadatabeskrivelser med registrerte datoer i dette tidsrommet som inngår i søkeresultatene.

Dette søket gir 2 treff - begge fra DRA - i forhold til 17 mulige søketreff for alle datoer. Alle de 13 potensielle søketreffene fra Galleri NOR filtreres vekk på grunn av at disses har samme dato, 1995-01-01, mens to potensielle treff fra DRA filtreres vekk på grunn av at de faller utenfor tidsintervallet 1945 - 1980. Dette eksempelet viser at datoavgrensningen fungerer, men er samtidig en illustrasjon på at datagrunnlaget kanskje blir litt for lite til å illustrere hovedpoenget med søketjenesten for avanserte søkeparametere.

Datosøk kan også brukes som en enkeltstående søkeparameter, for eksempel ved å søke etter alle metadatabeskrivelser fra et bestemt år. Året 1996 gir 5 søketreff, hvorav ett beskriver et radioinnslag fra DRA, mens de fire andre beskriver videofilmer fra Mavis.

Eksempel 3 (avgrensning v.h.a. datakilde og/eller ressurstype)

Det avanserte søkegrensesnittet gir også brukeren mulighet til å avgrense hvilke datakilder metadatabeskrivelsene opprinnelig skal komme fra. På denne måten kan brukeren i praksis søke i hver enkelt kilde fra et enhetlig søkegrensesnitt istedenfor å måtte oppsøke den enkelte datakilden.

Ved å søke etter *Eidsvold* i søkefeltet *Abstract*, samtidig som datakilde avgrenses til *Galleri NOR*, oppnår man 13 søketreff. Hadde kilden derimot vært *DRA*, ville resultatet blitt 4 søketreff. Totalt sett blir altså alle de 17 søketreffene funnet. Ved å søke i hver enkelt av disse kildene, ville brukeren ha oppnådd det samme resultatet. Da måtte imidlertid han/hun ha forholdt seg til forskjellige typer søkegrensesnitt, og dette kunne ha komplisert søkeprosessen.

I likhet med at brukeren kan avgrense opprinnelig datakilde, kan han/hun også angi hvilke ressurstyper det skal søkes etter. Ved for eksempel å søke etter *Eidsvold* i søkefeltet *Abstract*, samtidig som ressurstype avgrenses til *lyd*, oppnår man 4 søketreff. Alle disse kommer fra DRA, ettersom dette er den eneste kilden til lydopptak i dette systemet.

På grunn av at det eksisterer et 1:1 forhold mellom kilde og ressurstype, er det i denne prototypen vanskelig å teste funksjonaliteten for søk etter forskjellige ressurstyper i kombinasjon med filtrering for opprinnlig kilde. Dersom systemet hadde inneholdt metadatabeskrivelser fra flere kilder, for eksempel tre forskjellige bildedatabaser, ville denne funksjonen kommet mer til sin rett enn det den gjør i denne prototypen.

6.3 Oppsummering

Testeksemplene av søketjenesten viser at en arkitektur basert på OAI-PMH gjør det mulig å søke i metadata fra flere kilder på ett sted. Hovedmålsetningen med systemet er med andre ord nådd.

I alle eksemplene på enkelt søk inneholder resultatlisten søketreff fra to av de tre opprinnelige samlingene. Dette er en bekreftelse på at systemet fungerer slik det er tenkt, det vil si at det gjør det lettere for brukeren å finne informasjon. Eksemplene illustrerer også at forskjellige typer søketermer, for eksempel emneord og stedsnavn, kan brukes til søking, enten enkeltvis eller i kombinasjoner med hverandre.

Avansert søk gir brukeren langt flere muligheter til å spesifisere i detalj hvilke metadatafelt det skal søkes i, for på den måten å oppnå mer presise søkeresultater. Avansert søk gir dessuten mulighet til kombinasjoner, avgrensning og raffinering.

Ideelt sett burde det eksistert et eksempel med søketreff fra alle tre datakildene samtidig. På grunn av at metadataene fra Mavis er konsentrert rundt et begrenset emneområde, oljeutvinning på Ekofisk-feltet, har dette imidlertid vært vanskelig. Likeledes har et relativt lite datagrunnlag gjort det vanskelig å teste alle mulighetene som er tilgjengelige i det avanserte søkegrensesnittet.

KAPITTEL 7

Evaluering og konklusjon

I innledningen av denne avhandlingen (se *kapittel 1*) ble oppgavens problemstilling formulert ved hjelp av to spørsmål:

- Hvordan kan en arkitektur som muliggjør samtidig søk i flere heterogene metadatabaser utformes?
- Hvordan kan et søkegrensesnitt som nyttiggjør seg denne arkitekturen oppbygges, slik at man gjennom ett enhetlig søkegrensesnitt kan få tilgang til ressursbeskrivelser fra alle de heterogene metadatabasene?

I denne avhandlingen er det presentert et forslag til en systemarkitektur, og det er utviklet en prototyp som demonstrerer arkitekturens viktigste egenskaper i praksis. I forrige kapittel ble søketjenesten testet og evaluert ut fra et brukerperspektiv, mens i dette kapittelet evalueres både komponentene i den underliggende arkitekturen og løsningen i et helhetsperspektiv. Det vil si hele systemet evalueres både i forhold til de opprinnelige forutsetningene og i forhold til valgene som ble gjort i *kapittel 4* (løsningsarkitektur).

Konkret innebærer dette en nærmere vurdering av:

- Systemet som helhet
- Komponenter i systemet
- Forutsetninger for løsningen
- Erfaringer underveis i utviklingsprosessen
- Alternative løsningsmetoder
- Forslag til videre arbeid

7.1 Systemet som helhet

En evaluering av systemet som helhet krever et tilbakeblikk på oppgavens hovedmålsetning (se *kap. 1.1, s. 1*): *Å gjøre informasjon fra flere forskjellige samlinger lettere tilgjengelig for brukere.*

Evalueringen av søketjenesten (se *kap. 6.3, s. 89*) er en bekreftelse på at prototypen oppfyller denne opprinnelige målsetningen. Med andre ord kan man si at:

- Systemet demonstrerer en arkitektur basert på metadatahøsting ved hjelp av OAI-PMH som muliggjør søking i metadata fra flere heterogene kilder
- Systemet demonstrerer et webbasert søkegrensesnitt som anvender denne underliggende arkitekturen

Arbeidet med å utvikle en systemarkitektur og en prototyp har resultert i flere viktige funn. De viktigste funnene i denne avhandlingen er:

- Det er mulig å utvikle et søkesystem basert på metadata som blir innhøstet ved hjelp av OAI-PMH
- Metadatahøsting kan brukes til å oppnå lokal interoperabilitet internt i en stor organisasjon
- Metadataformatet MODS, som er et mer uttrykksfullt alternativ enn Dublin Core, kan brukes som felles metadataformat
- Fast Data Search kan integreres som søke- og indekseringsløsning

En ekstra bonus er at systemet også kan brukes til å gjøre metadata fra samlinger uten et eksisterende websøkegrensesnitt tilgjengelige for søking. Dette er tilfellet med metadataene fra datakilden Mavis i denne prototypen. Dersom en organisasjon, som for eksempel Nasjonalbiblioteket, på en ressursbesparende måte ønsker å tilby søkemuligheter for mange samlinger, kan en systemarkitektur basert på metadatahøsting benyttes.

7.1.1 Sammenligning med eksisterende løsninger

Sammenlignet med tilsvarende eksisterende løsninger (se *kapittel 3*) har systemet i dette prosjektet flere likheter, men også enkelte forskjeller.

Alle systemene har en felles grunnleggende systemarkitektur (se *kap. 3.1.1, s. 33*), det vil si de inneholder komponentene OAI-høster, normaliserer, lokalt datalager og søketjeneste. Selv om det eksisterer lokale forskjeller i implementasjonen av disse komponentene, fungerer de omtrent likt i praksis. Til forskjell fra Arc og OAIster inkluderer ikke prototypen i dette prosjektet komponenter for administrasjon av systemet, hovedsakelig fordi disse ikke er nødvendige for å illustrere prinsippene i søketjenesten. Alle systemene tilbyr brukeren både enkelt og avansert søkegrensesnitt.

En grunnleggende forskjell er at søketjenesten i dette prosjektet er utviklet internt i en organisasjon, det vil si at den samme entiteten er ansvarlig både for datakilder og søketjeneste. Dette er en alternativ og lite omtalt måte å anvende OAI-PMH på, sammenlignet med de globale søketjenestene Arc, OAIster og Scirus som høster metadata fra mange forskjellige organisasjoner. Prototypen demonstrerer imidlertid at OAI-PMH også egner seg til dette formålet.

Anvendelsen av MODS som felles metadataformat er også en viktig forskjell fra de andre søketjenestene, som baserer seg på Dublin Core. Og bortsett fra Scirus er det ingen andre OAI-PMH-baserte søketjenester som anvender Fast Data Search som intern søke- og indekseringsløsning.

7.2 Komponenter i systemet

Arkitekturen i systemet (se *figur 5-1, s. 64*) består av flere forskjellige komponenter som er basert på forskjellige teknologier. For å utdype den helhetlige vurderingen av systemet, er det hensiktsmessig å gjøre en enkeltvis evaluering av systemets komponenter.

7.2.1 Samlinger

Ettersom hensikten med dette prosjektet er å integrere forskjellige typer samlinger med forskjellige typer innhold i ett søkegrensesnitt, er det en forutsetning at et visst antall samlinger benyttes i prototypen. I *kap. 4.2.2, s. 51* ble tre til fem samlinger vurdert til å være et tilstrekkelig antall for å illustrere prinsippet i dette systemet. Et utdrag metadatabeskrivelser fra de tre samlingene Digitalt Radioarkiv, Galleri NOR og Mavis ble valgt til å utgjøre datagrunnlaget for systemet.

Disse tre samlingene inneholder metadata for tre forskjellige typer informasjonsobjekter, noe som er tilstrekkelig til å illustrere prinsippene i denne avhandlingen. Dersom en ønsker å utdype flere forhold knyttet til søking i heterogene metadata, bør antall samlinger som inngår i systemet utvides. Utvalgskriterier for ytterligere samlinger kan være:

- Å inkludere flere samlinger med metadatabeskrivelser for de samme typene informasjonsobjekter som allerede er benyttet i prototypen. Dette kunne gitt et bedre grunnlag for å evaluere søketjenesten, samt forskjeller mellom beskrivelser av samme ressurstype.
- Å inkludere samlinger som inneholder andre typer informasjonsobjekter, for eksempel tekstdokumenter.
- Å inkludere samlinger som inneholder rikere metadatabeskrivelser enn dem som er benyttet i denne prototypen. Dette kunne gitt et bedre grunnlag for evaluering av metadataformatet MODS, samt flere forhold knyttet til normalisering.

Metadatakvalitet

Systemet inneholder totalt et utvalg på 1403 metadatabeskrivelser fra de tre samlingene:

- 1000 beskrivelser av fotografier fra Galleri NOR
- 281 beskrivelser av radioprogrammer fra Digitalt Radioarkiv
- 122 beskrivelser av videofilmer fra Mavis

På bakgrunn av erfaringene fra lignende prosjekter (se *kap. 3.7, s. 44*), var det på forhånd ventet at det ville være variasjoner i de innhøstede metadatabeskrivelsene, både hva angår rikhet i metadataene og hvilke retningslinjer de originale metadataene er beskrevet i henhold til. Dette viste seg også å være tilfelle ved nærmere undersøkelse av metadataene fra de ulike kildene.

I løpet av utviklingsprosessen er det erfart både begrenset kvalitet og/eller rikdom (richness) i metadatabeskrivelsene for de forskjellige ressursene i de ulike databasene. Kvalitetsproblemene dreier seg i første rekke om inkonsistens i beskrivelsene innad i hver enkelt metadatabase, for eksempel flere forskjellige måter å registrere datoer på, eller variasjoner i registrering av navn. Når det gjelder metadatarikdom dreier dette seg om at bare noen få egenskaper knyttet til en ressurs er registrert.

Mens beskrivelsene fra GalleriNOR og Mavis inneholder relativt enkle opplysninger i et begrenset antall av metadataelementene, er beskrivelsene fra Digitalt Radioarkiv mer omfattende. For eksempel inneholder *abstract*-feltet i en DRA-beskrivelse et fulltekstsammendrag av innholdet i radioprogrammet, mens *abstract*-feltet i en GalleriNOR-beskrivelse er et duplikat av informasjonen i tittelfeltet. I Mavis-beskrivelser brukes ikke abstract-feltet overhodet.

Problemstillinger knyttet til kvaliteten i metadataene skal, i henhold til *[OAI-PMH]*, behandles av OAI tjenestetilbyderen før dataene indekseres. Ettersom Nasjonalbiblioteket både fyller rollen som datatilbyder og tjenestetilbyder, kan det imidlertid gjøres undersøkelser på forhånd av de forskjellige samlingene for å konstatere om metadataene har høy nok kvalitet til at de kan inkluderes i en felles søketjeneste.

En konsekvens av den manglende rikdommen er at bare et subsett av elementene i MODS-formatet utnyttes. Dette kan medføre at det blir vanskelig å bli klar over styrkene og svakhetene ved MODS.

Problemstillinger knyttet til OAI datatilbydere

Ettersom utviklingen av de forskjellige OAI datatilbyderne ikke har inngått direkte i dette prosjektet, har det dukket opp flere problemstillinger i tilknytning til dette.

Generelle problemstillinger for OAI datatilbydere:

- Uklarhet rundt hvilket metadataprefiks som skulle benyttes (oai_mods eller mods)
- Uklarhet rundt hvilken MODS-versjon (2 eller 3) som skulle benyttes

Problemstillinger relatert til OAI datalager:

- Usikkerhet rundt bruk av elementene name/role til å beskrive bygninger i subject
- Usikkerhet rundt rollebeskrivelser
- Attributtet type for elementet navn (personal | corporate)

Problemstillinger relatert til metadata i de opprinnelige samlingene:

- Parenteser og forkortelser i tittelinformasjon
- Opplysninger om navn/rolle i tittelfeltet
- Flere termer/beskrivelser/verdier i ett og samme tekstfelt (name, role, subject)
- Forskjellige måter å angi dato på

- Inkonsekvent formattering av tidsintervaller innad i et datalager

I *appendiks A, s. 111* finnes en nærmere beskrivelse av erfaringer knyttet til de opprinnelige samlingene og deres funksjonalitet som OAI-datatilbydere.

7.2.2 Felles metadataformat

Som nevnt i *kap. 2.5.1, s. 23* utgjør et felles format for metadatabeskrivelser fra alle samlingene en forutsetning for å kunne samle alle metadataene i en sentral indeks. I *kap. 4.2.3, s. 52* ble MODS foretrukket framfor Dublin Core som felles metadataformat i dette prosjektet, hovedsakelig på grunn av:

- Ufullstendighet og manglende retningslinjer for bruk av kvalifikatorer i Dublin Core
- Mulighet for rike, hierarkiske metadatabeskrivelser i MODS.

Prototypen bekrefter at det er fullt mulig å bruke MODS som fellesformat i denne typen prosjekter, og at mulighetene for hierarkiske metadatabeskrivelser gjør det mulig å uttrykke sammenhenger mellom metadataelementer på en måte som ikke er mulig ved hjelp av Dublin Core. Effekten av disse hierarkiske metadatabeskrivelsene forsvinner imidlertid ved indeksering. Årsaken er at de innhøstede metadataene i MODS-formatet må konverteres til det ikke-hierarkiske dataformatet FastXML, som Fast Data Search krever ved indeksering.

De relativt enkle opprinnelige metadatabeskrivelsene som benyttes i dette systemet fører imidlertid til at potensialet i MODS-formatet bare utnyttes delvis. Dersom en ønsker å teste flere av egenskapene til MODS, bør det anvendes samlinger med rikere metadata, for eksempel i MARC-format.

Et interessant spørsmål i denne sammenhengen er hvordan Dublin Core ville fungert som felles metadataformat i dette prosjektet. Som allerede nevnt utnyttes ikke potensialet i MODS-formatet til fulle, og med de relativt enkle metadataene som er anvendt i dette systemet, ville de samme resultatene kunne blitt oppnådd ved hjelp av Dublin Core som felles metadataformat.

7.2.3 OAI-høster

I *kap. 4.2.4, s. 54* ble det besluttet at den ferdigutviklete høsterapplikasjonen OAIHarvester2 skulle brukes som utgangspunkt i denne prototypen. OAIHarvester2 fungerer slik den skal i henhold til OAI-protokollen *[OAI-PMH]*, og gevinsten i form av tids- og arbeidsbesparelse har vært stor sammenlignet med å måtte utvikle høsteren på egen hånd. Dette prosjektet bekrefter dermed lignende erfaringer fra systemene OAIster og Scirus.

Ettersom kildekoden til høsteren er åpen, har det vært lett å videreutvikle høsteren, slik at den er tilpasset dette systemet. Utvidelsen, i form av en ekstra klasse som sørger for omforming av de innhøstede metadataene ved hjelp av stilark, fungerer også etter hensikten. Det vil si at høsting og omforming av metadata utføres i den samme operasjonen.

Den kommandolinjebaserte høsteren løser oppgaven med å høste inn og omforme metadata. Dersom en ønsker å gjøre OAI-høsteren mer brukervennlig, finnes det flere muligheter for ytterligere utvidelser:

- Webbasert administrasjonsgrensesnitt
- Automatisk oppgavefordeler
- Automatisk høstingsplanlegger

Denne typen utvidelser er ikke prioritert i dette systemet på grunn av at de har liten betydning i forhold til oppgavens hovedmål.

7.2.4 Metadataomforming

Prototypen demonstrerer at strukturomforming og normalisering av metadata uttrykt i xml kan gjøres relativt enkelt ved hjelp av stilark. Denne metoden er meget anvendelig på grunn av utvidbarheten. Dersom det for eksempel oppstår behov for å omforme de innhøstede metadataene til andre strukturer eller gjøre flere normaliseringer, kan dette uttrykkes ved hjelp av nye stilark. Det kreves med andre ord ingen endringer av OAI-høsteren som utfører omformingen.

Hovedårsaken til at det utføres en normalisering av de innhøstede metadataene er:

- Varierende metadatakvalitet internt i enkelte samlinger
- Forskjellige uttrykksmåter for de ulike samlingene

I denne prototypen er datokonverteringen brukt som en illustrasjon på hvordan verdier i de innhøstede metadataene kan normaliseres ved hjelp av stilark.

I prototypen utføres både strukturomformingen og datanormaliseringen ved hjelp av ett stilark. Denne løsningen er tilstrekkelig til å illustrere prinsippene, men har begrenset utvidbarhet. Dersom en ønsker å normalisere andre metadatafelt enn dato, for eksempel navn, bør det utvikles egne stilark for hver enkelt type normalisering. I tillegg bør strukturomforming og normalisering utføres ved hjelp av adskilte stilark.

En svakhet ved stilarkmetoden, som også erfart i utviklingen av OAIster *[HAG03]*, er at systemet kun er laget for et kjent antall datoformater. Hvis en ny samling skal inkluderes, er det sannsynlig at nye datoformater vil dukke opp, men disse vil ikke bli fanget opp av den eksisterende normaliseringsrutinen. Årsaken er at det ikke er definert omformingsregler for det nye datoformatet på forhånd. Dersom en ønsker å gjennomføre normaliseringer av flere metadatafelt eller inkludere flere samlinger i systemet, bør normaliseringskomponenten videreutvikles, for eksempel ved å ta i bruk kontrollerte vokabularer.

7.2.5 Fast Data Search

Prototypen demonstrerer at det er mulig å integrere Fast Data Search i et søkesystem hvor datagrunnlaget er metadata som er innhøstet ved hjelp av OAI-PMH. Det webbaserte administra-

sjonsgrensesnittet og konseptet med en indeksprofil som kan reflektere strukturen i metadataformatet, gjør Fast Data Search lett å konfigurere. Dessuten gjør de tydelig definerte API-ene, både på indekserings- og søkesiden, det uproblematisk å integrere Fast Data Search i dette søkesystemet.

En mer detaljert evaluering av anvendelsen av Fast Data Search finnes i *kap. 7.3.2, s. 98*.

7.2.6 Søketjeneste

Søketjenesten er testet og evaluert i *kapittel 6*. Kort oppsummert fungerer søketjenesten som forventet, det vil si den demonstrerer et enkelt og avansert søkegrensesnitt som gjør det mulig å søke i metadatabeskrivelser fra flere kilder på ett sted. Dersom en ønsker å gjøre tjenesten mer interessant for brukerne, kan søketjenesten utvides slik at flest mulig metadatabeskrivelser også inneholder pekere som gir direkte tilgang til informasjonsobjektene som er beskrevet. Dette er et av de grunnleggende prinsippene i OAIster (se *kap. 3.3, s. 38*).

7.3 Forutsetninger for løsningen

Som nevnt i *kap. 1.2, s. 2* og *kap. 4.1, s. 47* er hele dette prosjektet basert på to grunnleggende forutsetninger: Anvendelse av teknologiene OAI-PMH og Fast Data Search. Selv om hovedmålet i prosjektet er oppnådd, er det betimelig med en evaluering av hvilke muligheter og begrensninger disse forutsetningene medfører.

7.3.1 OAI-PMH

Metadatahøsting ved hjelp av OAI-PMH kan anvendes som grunnlag for å lage en felles søketjeneste for flere datakilder internt i en organisasjon, det vil si oppnå lokal interoperabilitet (se *kap. 2.3.1, s. 20*). Løsningen er spesielt virksom og effektiv når mange og varierte datakilder skal inkluderes i søketjenesten.

Hvis det først er utviklet en OAI tjenestetilbyder som for eksempel tilbyr en søketjeneste, gjør OAI-PMH det enkelt å inkludere nye datakilder i denne tjenesten. Protokollen krever kun at det implementeres en OAI datatilbyderapplikasjon og at det designes en metadataovergang til felles metadataformat hos hver enkelt datakilde.

Bieffekter av at en datakilde gjøres om til en OAI datatilbyder er:

- Andre, eksterne systemer og organisasjoner kan høste metadata fra Nasjonalbibliotekets samlinger, hvis dette er ønskelig
- Man åpner muligheten for søk i metadata fra samlinger/databaser som ikke tilbyr søkefunksjonalitet i utgangspunktet

Begrensningene til OAI-PMH ligger først og fremst i at det er en relativt enkel protokoll som kun fokuserer på metadatahøsting. I *kap. 2.6.2, s. 26* ble det beskrevet hva OAI-PMH ikke er, og de to viktigste begrensningene er:

- OAI-PMH er ingen søkeprotokoll. Dette vil si at det ikke søkes direkte i datakilden, men i en kopi av innholdet i datakilden.
- OAI-PMH har ingen innebygd mekanisme for høsting av selve informasjonsobjektene, men muliggjør bare høsting av metadata

Ettersom det søkes i kopier av innholdet i datakildene, kan det oppstå synkroniseringsproblemer dersom nye data registreres i en datakilde uten at disse blir høstet inn og lagret i søkesystemets indeks. Denne problemstillingen er først og fremst aktuell for datakilder hvor det skjer hyppige oppdateringer av innholdet, og løsningen er å utvikle automatiseringsrutiner som sørger for at det jevnlig utføres høstinger av nye metadata.

7.3.2 Fast Data Search

Anvendelse av søkemotorteknologi i forbindelse med digitale bibliotek er et lite utforsket område, men som nevnt i *kap. 7.2.4, s. 96* er Fast Data Search relativt godt egnet som søke- og indekseringsløsning i dette systemet.

Den største fordelen med Fast Data Search er først og fremst at det er et komplett system for søking og indeksering som inkluderer en mengde søkemuligheter. Disse søkemulighetene kan benyttes ved hjelp av et ferdig definert spørrespråk. I dette prosjektet har dette ført til at utviklingstid har kunnet kanaliseres mot andre viktige oppgaver.

En annen fordel med systemet er den høye søkeytelsen, sammenlignet med tilsvarende systemer som anvender relasjonsdatabaser som lokale søkeindekser. Denne fordelen har imidlertid liten betydning i denne prototypen, men kan være avgjørende i et operativt system.

Fast Data Search har også enkelte begrensninger, og i dette prosjektet er det først og fremst den manglende støtten for hierarkiske metadatabeskrivelser i MODS-formatet som er lite ideell. Ettersom Fast sitt interne xml-format FastXML bare håndterer xml-elementer på ett nivå, er det ikke mulig å opprettholde de hierarkiske koblingene mellom metadataelementer i MODS-formatet. Et mulig alternativ kunne vært å benytte en XML-database som lokal indeks. Dette ville imidlertid gitt hele utviklingsprosessen et annet fokus. Flere detaljer angående dette alternativet finnes i *kap. 7.5.2, s. 99*.

7.4 Erfaringer underveis i prosjektet

I løpet av et prosjektarbeid blir det kontinuerlig foretatt vurderinger, beslutninger og avgrensninger. Etter å ha gjennomført en utviklingsprosess som i dette prosjektet, sitter man igjen med viktige erfaringer som kan komme til nytte i lignende prosjekter i framtiden.

En interessant problemstilling som oppstod underveis var lanseringen av nye versjoner av OAI-Harvester og metadataformatet MODS. Det er ikke mulig å forsikre seg mot at det lanseres nye versjoner av eksterne programmer og standarder underveis i en utviklingsprosess. Når en slik si-

tuasjon oppstår, må det gjøre en evaluering av hvor store endringer det dreier seg om, og hvor krevende det vil være å inkludere nyhetene i utviklingsprosjektet. I dette prosjektet ble den nye versjonen av OAIHarvester inkludert på bekostning av den gamle, hovedsakelig på grunn av en enklere oppbygning. Ettersom utviklingen av omformingsrutinen for den eksisterende versjonen av MODS var kommet godt i gang, ble ikke den nye versjonen inkludert i prosjektet.

En annen faktor som har påvirket utviklingsprosessen er at det underveis ble større fokus på integrasjonen av Fast Data Search enn opprinnelig planlagt. I utgangspunktet ble Fast Data Search sett på som en "svart boks" som kunne anvendes direkte sammen med de andre komponentene i systemet. Underveis i utviklingsprosessen ble det imidlertid klart at det måtte gjøres både tilpasninger og konfigurasjoner. I ettertid har dette vist seg å være en nyttig erfaring som har gitt en bedre helhetsforståelse av systemet.

7.5 Alternative løsningsmetoder

I dette prosjektet var mange av løsningsmetodene indirekte gitt på forhånd, som følge av forutsetningene for prosjektet. I ettertid er det derfor nyttig å belyse alternative løsningsmetoder som kunne ha påvirket sluttresultatet.

Bruk av OAI-sett er en detalj som kunne ha påvirket løsningen i positiv retning uten at det ville endret utviklingsprosessen i særlig grad. Bruk av xml-databaser og distribuert søking er løsningsmetoder som er grunnleggende forskjellige, og som ville ha ført til annerledes fokus i utviklingsprosessen.

7.5.1 Bruk av OAI-sett (mengder)

Ingen av Nasjonalbibliotekets OAI-datalagre har benyttet OAI-sett til å lage en logisk inndeling av innholdet i databasene. I databaser med variert innhold, som for eksempel Mavis som inneholder metadatabeskrivelser av forskjellige typer informasjonsobjekter, for eksempel lyd og video, ville bruk av sett gjort det mulig å gjøre selektive høstinger av bestemte typer ressurser.

I dette prototypeprosjektet er settinndeling underordnet. I et operativt system ville imidlertid settinndeling kunne gi flere og mer varierte høstemuligheter.

7.5.2 Bruk av "native" xml-database istedenfor Fast Data Search

I *kap. 3.1.1, s. 33* beskrives et generelt forslag til systemarkitektur for en OAI tjenestetilbyder. Alle tjenestetilbydere som har høstet inn metadata, har behov for å lagre disse metadataene i en lokal indeks. Det eksisterer flere alternative løsninger for indeksering, blant annet relasjonsdatabaser, XML-databaser og indekserings- og søkesystemer.

Som nevnt i *kap. 7.2.5, s. 96*, krever Fast Data Search at dataene som skal indekseres er beskrevet i et ikke-hierarkisk format. Hierarkiske strukturer, som blant annet finnes i MODS-beskrivelser, kan dermed ikke indekseres. Dette har medført at fordelene man oppnådde ved å bruke hierarkisk

oppbygde MODS som metadataformat, forsvinner når de innhøstede metadataene etter hvert må konverteres til et ikke-hierarkisk format som Fast Data Search kan håndtere.

En alternativ løsning, som trolig ville forhindret denne problematikken, er å bruke en "native" xml-database som lokal indeks istedenfor Fast Data Search. Ifølge *[NAT03]* kan slike xml-databaser lagre, indeksere og gjenfinne hele xml-dokumenter med deres opprinnelige struktur. Indeksene i xml-databasene gjør det effektivt å søke etter et hvilket som helst punkt i xml-dokumentet. På denne måten kunne de innhøstede metadatabeskrivelsene i MODS blitt indeksert direkte i xml-databasen, uten konvertering til et annet dataformat. Det eksisterer i dag flere xml-databasesystemer, blant annet eXist, Berkeley DB XML og Xindice (åpen kildekode), samt Tamino (kommersiell).

Fordeler ved å bruke en xml-database er:

- Bedre utnyttelse av opprinnelig xml-struktur
- Effektiv søkemekanisme

Ulempen er relativt lav ytelse ved oppdatering av innholdet, hovedsakelig på grunn av at hele indeksen må konstrueres på nytt. Sammenlignet med Fast Data Search ville det dessuten tatt lengre tid å utvikle søkefunksjonaliteten for en xml-database.

Det svenske prosjektet *Digitala vetenskapeliga arkivet [DIVA]* er et eksempel på en søketjeneste hvor dataene er indeksert i en xml-database.

7.5.3 Bruk av Search/Retrieve Web Service (SRW)

I innføringen av ulike interoperabilitetsmetoder (se *kap. 2.4, s. 20*), ble metadatahøsting med påfølgende sentral indeksering sammenlignet med interoperabilitetsmetoden distribuert søking. Search/Retrieve Web Service *[SRW]* er en protokoll for distribuert søking som kombinerer de grunneleggende prinsippene i søkeprotokollen Z39.50 med webteknologi. SRW framstår derfor som et alternativ til OAI-PMH når målet er å utvikle en felles søketjeneste for flere samlinger/databaser.

Search/Retrieve Web Service (SRW) og tvillingprotokollen Search/Retrieve URL Service (SRU) gjør det mulig for en søkeklient å sende spørringer direkte til databaser og deretter motta søkereultater fra disse databasene. SRW og SRU tilbyr akkurat den samme funksjonaliteten, men forskjellen ligger i måten spørringene og resultatene uttrykkes og formidles mellom klient og tjener.

Hovedprinsippet i SRU/SRW er at en søketjeneste (klient) kringkaster brukerens søkeuttrykk som spørringer (uttrykt i språket Common Query Language) til de ønskede databasene (servere). Når disse databaseserverene mottar en spørring, utføres det et lokalt søk i databasene, før søkeresultatene returneres i ønsket format, for eksempel ren tekst, Dublin Core eller MODS. Når søketjenes-

ten (klienten) har mottatt søkeresultatene fra alle databasene, flettes disse resultatene sammen i en enhetlig resultatliste som presenteres for brukeren.

Sammenlignet med OAI-PMH har SRU/SRW et klart avgrenset anvendelsesområde - søking - ettersom det bare er en søkeprotokoll. OAI-PMH en metadatahøstingsprotokoll som har flere bruksområder enn en ren søkeprotokoll, for eksempel metadataaggregering. Ytterlige eksempler på hvordan OAI-PMH kan anvendes, fins i *[YOU03]*.

Begge protokollene krever implementasjon både på klient- og serversiden. Hvis det etter hvert kan oppstå behov for å anvende metadataene på andre måter enn i en søketjeneste, er OAI-PMH trolig et bedre valg enn Search/Retrieve Web Service.

7.6 Forslag til videre arbeid

Selv om prototypen som er utviklet i dette prosjektet fungerer etter hensikten, er det likevel bare en prototyp som illustrerer prinsipper - ikke et operasjonelt system. Det eksisterer med andre ord flere potensielle muligheter for videre utredning og utvikling. I tillegg til alternativene i *kap. 7.6, s. 101*, er det i løpet av prosjektet avdekket flere områder som det kan være interessant å jobbe videre med:

- På grunn av varierende kvalitet i metadataene, kan det være nyttig med en grundigere evaluering av hvorvidt alle metadatabaser har egnede metadata til å kunne inngå i et system som dette.
- For å oppnå større enhetlighet i søketjenesten, kan det gjøres en ytterligere behandling av de innhøstede metadataene ved hjelp av normalisering før indeksering. Metadatafelt som en bør vurdere å normalisere er for eksempel navn, emneord, forlag og ressurstype.
- For å gjøre søketjenesten mer helhetlig for brukerne, kan søketjenesten utvides slik at den bare inneholder metadatabeskrivelser med pekere som gir direkte tilgang til de digitale objektene. I enkelte tilfeller vil dette kreve avklaringer av opphavsrettslige spørsmål.
- Dersom en ønsker å høste metadata beskrevet i hierarkiske formater som for eksempel MODS, bør det gjøres en grundigere utredning av fordeler og ulemper knyttet til bruk av en XML-database istedenfor Fast Data Search.
- På administratorsiden er det store rom for forbedringer, for eksempel automatiserte rutiner for metadatahøsting og indeksering, samt bedre konfigurasjonsmuligheter av metadatahøsteren. På denne måten vil høstingen kunne foregå dynamisk med jevne mellomrom.
- Search/Retrieve Web Service (SRW) er i *kap. 7.5.3, s. 100* beskrevet som et alternativ til OAI-PMH. Ifølge *[VEE04]* er det imidlertid også mulig å kombinere disse to protokollene, for eksempel ved å bygge ut en OAI tjenestetilbyder som har samlet inn metadata fra mange kilder til også å fungere som en SRW-server som kan motta henvendelser fra en SRW-søketjeneste. Det kan være verdt å undersøke om det er mulig å utvide dette systemet med SRW-funksjonalitet, slik at søketjenesten kan inngå som en søkenode i SRW-baserte The European Library *[TEL]*.

Referanseliste

[AHL02] Ahlborn, Benjamin, Wolfgang Nejdl og Wolf Siberski. OAI-P2P: A Peer-to-Peer Network for Open Archives. Proceedings of the International Conference on Parallel Processing Workshops (ICPPW'02), 2002, s. 462-468.

[ARC] Arc - A Cross Archive Search Service.
http://arc.cs.odu.edu/

[ARC03] Arms, Caroline R. Available and useful: OAI at the Library of Congress. Library Hi Tech, 2003, 21:2, s. 129 - 139.

[ARW02] Arms, William Y. med flere. A Spectrum of Interoperability - The Site for Science Prototype for the NSDL. D-Lib Magazine, 2002, 8:1.
http://www.dlib.org/dlib/january02/arms/01arms.html

[BAK02] Baker, Thomas, Makx Dekkers, Rachel Heery, Manjula Patel og Gauri Salokhe, What Terms Does Your Metadata Use? Application Profiles as Machine-Understandable Narratives. Journal of Digital Information, 2002, 2:2.
http://jodi.ecs.soton.ac.uk/Articles/v02/i02/Baker/baker-final.pdf

[BES02] Besser, Howard. The Next Stage: Moving from Isolated Digital Collections to Interoperable Digital Libraries. First Monday, 2002, 7:6.
http://www.firstmonday.org/issues/issue7_6/besser/index.html

[BOW94] Bowman, C. Mic, Peter B. Danzig, Darren R. Hardy, Udi Manber og Michael F. Schwartz. The Harvest Information Discovery and Access System. Proceedings of the Second International World Wide Web Conference, 1994, s. 763-771.
http://www.codeontheroad.com/papers/Harvest.Conf.pdf

[CON02] Conallen, Jim. Building Web Applications With UML 2nd Ed. Addison-Wesley, Boston, Massachusetts, 2002.

[DIL03] Dillon, Martin. Prefatory Commentary from the Editor i [GUE03b], 2003, s. 137-138.

[DC-Lib] Guenther, Rebecca. DC-Library Application Profile (DC-Lib), Dublin Core Metadata Initiative (DCMI) Working Draft, 2002.
http://dublincore.org/documents/library-application-profile/

[DCMI03] Baker, Thomas. DCMI Usage Board Review of Application Profiles, 2003.
http://dublincore.org/usage/documents/profiles

[DIVA] Digitala Vetenskapliga Arkivet. Enheten for digital publisering ved Uppsala universitetsbibliotek.
http://www.diva-portal.org/

[DLXS] Digital Library eXtension Service.
http://www.dlxs.org

[DP9] DP9 - An OAI Service Provider for Web Crawlers.
http://dlib.cs.odu.edu/dp9/

[FAST] Fast Search & Transfer.
http://www.fastsearch.com/

[FDS03] Fast Data Search System Reference Guide, versjon 3.2.1. Fast Search & Transfer, 2003.

[GILS] Global Information Locator Service (GILS).
http://www.gils.net/overview.html

[GIL00] Gilliland-Swetland, Anne J. Setting the Stage: Defining Metadata i Introduction to Metadata: Pathways to Digital Information, second edition, Murtha Baca, red. Los Angeles: Getty Information Institute, 2000.
http://www.getty.edu/research/conducting_research/standards/intrometadata/pdf/swetland.pdf

[GNOR] Galleri NOR, Nasjonalbibliotekets fotodatabase.
http://www.nb.no/gallerinor/

[GUE02] Guenther, Rebecca. MODS: overview, uses and experimentation. Presentasjon på DLF Fall Forum 2002, Seattle, Washington.

[GUE03a] Guenther, Rebecca og Sally McCallum. New Metadata Standards for Digital Resources: MODS and METS. Bulletin of the American Society for Information Science and Technology, 2003, 29:2, s. 12-15.
http://www.asis.org/Bulletin/Dec-02/ASISTDecJan.pdf

[GUE03b] Guenther, Rebecca. MODS: The Metadata Object Description Schema. portal: Libraries and the Academy, 2003, 3:1, s. 137-150.

[GUE03c] Guenther, Rebecca. E-post med svar på henvendelsen “Questions about MODS/DC-Lib”, 18. juni 2003.

[GUE04] Guenther, Rebecca. E-post med svar på henvendelsen "[MODS] <subject> element: <name> question", 29. januar 2004.

[HAG03] Hagedorn, Kat. OAIster: a "no dead ends" OAI service provider. Library Hi Tech, 2003, 21:2, s. 170-181.

[HEE00] Heery, Rachel og Manjula Patel. Application profiles: mixing and matching metadata schemas. Ariadne, 2000, Issue 25.
http://www.ariadne.ac.uk/issue25/app-profiles/

[KRI01] Krichel, Thomas og Simeon M. Warner. A metadata framework to support scholarly communication. Proceedings of the International Conference on Dublin Core and Metadata Applications. National Institute of Informatics, 2001.
http://dublincore.org/archives/2001/10/public_html/proceedings/product/paper-22.pdf

[LAG95] Lagoze, Carl og J. R. Davis, Dienst - An Architecture for Distributed Document Libraries. Communications of the ACM, 1995, 38:4, s. 47.

[LAG01] Lagoze, Carl og Herbert Van de Sompel, The Open Archives Initiative: Building a Low-Barrier Interoperability Framework. Proceedings of the 1st ACM/IEEE-CS joint conference on Digital Libraries, ACM Press, New York, NY, USA, 2001, s. 54-62.

[LAG03] Lagoze, Carl og Herbert Van de Sompel. The making of the Open Archives Initiative Protocol for Metadata Harvesting. Library Hi Tech, 2003, 21:2, s. 118-128.

[LC] Library of Congress.
http://loc.gov/

[LER04] Lervik, John M. Search Engine Industry Trends - Impact for Digital Libraries. Presentasjon på 7th International Bielefeld Conference, 2004.
http://conference.ub.uni-bielefeld.de/proceedings/lervik.pps

[LIU01] Liu, Xiaoming, Kurt Maly, Mohammad Zubair og Michael L. Nelson. Arc - An OAI Service Provider for Digital Library Federation. D-Lib Magazine, 2001, 7:4.
http://www.dlib.org/dlib/april01/liu/04liu.html

[LIU02] Liu, Xiaoming, Kurt Maly, Mohammad Zubair, Qiaoling Hong, Micheal L. Nelson, Frances Kundson og Irma Holtkamp, Federated Searching Interface Techniques for Heterogenous OAI Repositories. Journal of Digital Information, 2002, 2:4.
http://jodi.ecs.soton.ac.uk/Articles/v02/i04/Liu/

[MODS] Metadata Object Description Schema.
http://loc.gov/standards/mods/

[MODS04] MODS Version 3.0 changes, Library of Congress, 2004.
http://www.loc.gov/standards/mods/changes-3-0-final.html

[MODX] MODS Version 2.0 xml schema, Library of Congress, 2003.
http://www.loc.gov/standards/mods/mods.xsd

[MUG] MODS User Guidelines Version 2.0, Library of Congress, 2003.
http://www.loc.gov/standards/mods/mods-userguide.html

[NAT03] Natu, Shalaka og John Mendonca. Digital Asset Management - Using A Native XML Database Implementation. Proceeding of the 4th conference on Information technology curriculum, ACM Press, 2003, s. 237 - 241.

[NBDB] Nasjonalbibliotekets databasetjenester.
http://nb.no/baser/

[NEL02] Nelson, Michael L. Service Providers: Future Perspectives. Presentasjon på 2nd Workshop on the Open Archives Initiative, CERN, oktober 2002.
http://agenda.cern.ch/askArchive.php?base=agenda&categ=a02333&id=a02333s5t5/transparencies

[NKKM] Feltkatalogen for NKKMs (Norske kunst- og kulturhistoriske museer) EDB-prosjekter.
http://helmer.aksis.uib.no/regimus/feltkode.html

[OAFOT] Open Archives Forum, OAI for Beginners - the Open Archives Forum online tutorial, 2003. Aksessert 26.01.2004.
http://www.oaforum.org/tutorial/

[OAI] Open Archives Initiative.
http://www.openarchives.org

[OAIC] OAICat. OCLC Research, 2003.
http://www.oclc.org/research/software/oai/cat.htm

[OAIDPR] Open Archives Initiative Data Provider Registry.
http://www.openarchives.org/Register/BrowseSites.pl

[OAIH] OAIHarvester2. OCLC Research, 2003.
http://www.oclc.org/research/software/oai/harvester2.htm

[OAIHG] Implementation Guidelines for the OAI-PMH - Guidelines for Harvester Implementers, Open Archives Initiative, 2002.
http://www.openarchives.org/OAI/2.0/guidelines-harvester.htm

[OAI-PMH] Open Archives Initiative Protocol for Metadata Harvesting version 2.0, Open Archives Initiative, 2002.
http://www.openarchives.org/OAI/openarchivesprotocol.html

[OAISTER] OAIster.
http://www.oaister.org/

[OAIT] Open Archives Initiative Tools.
http://www.openarchives.org/tools/tools.html

[ORE] OAI Repository Explorer.
http://purl.org/net/oai_explorer

[RMHP] Research Metadata Harvest Project. Draft Meeting Summary, Technical Issues Meeting. Harvard Univerity, 2000.
http://www.diglib.org/architectures/may1minute.pdf

[ROS02] Rosenfeld, Louis og Peter Morville. Information Architecture for the World Wide Web, 2nd edition. O'Reilly & Associates, Sebastopol, California, 2002, s. 132-175.

[SCI03] Scirus White Paper, How Scirus Works. Elsevier Science Inc., 2003.
http://www.scirus.com/press/pdf/WhitePaper_Scirus.pdf

[SCI04] Clarke, Clive. Product Support Manager Scirus. E-post med svar på henvendelsen "Questions about Scirus", 8. juli 2004.

[SCIRUS] Scirus.
http://www.scirus.com/

[SRW] Search/Retrieve Web Service.
http://www.loc.gov/z3950/agency/zing/srw/

[TEL] The European Library.
http://www.europeanlibrary.org

[UIUC] University of Illinois at Urbana-Champaign. Open Archives Initiative Metadata Harvesting project.
http://oai.grainger.uiuc.edu/

[VAN00a] Van de Sompel, Herbert og Carl Lagoze. The Santa Fe Convention of the Open Archives Initiative. D-Lib Magazine, 2000, 6:2.
http://www.dlib.org/dlib/february00/vandesompel-oai/02vandesompel-oai.html

[VAN00b] Van de Sompel, Herbert, m.fl. The UPS Prototype: An Experimental End-user Service across E-print Archives. D-Lib Magazine, 2000, 6:2.
http://www.dlib.org/dlib/february00/vandesompel-ups/02vandesompel-ups.html

[VEE03] Veen, Theo van, Harvesting metadata and the use of application profiles. Workshop introduction paper. ELAG 2003 Conference, 2003.
http://www.elag2003.ch/ws/ws_1.pdf

[VEE04] Veen, Theo van og Bill Oldroyd, Search and Retrieval in the European Library - A New Approach. D-Lib Magazine, 2004, 10:2.
http://www.dlib.org/dlib/february04/vanveen/02vanveen.html

[WAR01] Warner, Simeon, Exposing and harvesting metadata using the OAI metadata harvesting protocol: A tutorial. High Energy Physics Libraries Webzine, 2001, issue 4.
http://library.cern.ch/HEPLW/4/papers/3/

[WEI02] Weibel, Stuart L., Erik Duval, Wayne Hodgins og Stuart Sutton. Metadata Principles and Practicalities. D-Lib Magazine, 2002, 8:4.
http://www.dlib.org/dlib/april02/weibel/04weibel.html

[WOL04] Woldering, Britta. The European Library: Integrated access to the national libraries of Europe. Ariadne, 2004, issue 38.
http://www.ariadne.ac.uk/issue38/woldering/

[WOO00] Woodley, Mary. Crosswalks: the Path to Universal Access? i Introduction to Metadata: Pathways to Digital Information, second edition, Murtha Baca, red. Los Angeles: Getty Information Institute, 2000.
http://www.getty.edu/research/conducting_research/standards/intrometadata/pdf/woodley.pdf

[XPAT] XPAT Search Engine.
http://dlxs.org/products/xpat.html

[YOU03] Young, Jeffrey A., Herbert Van de Sompel og Thomas B. Hickey. Using the OAI-PMH ... differently. D-Lib Magazine, 2003, 9:7/8.
http://www.dlib.org/dlib/july03/young/07young.html

Ordliste

API
Application Programming Interface. Beskrivelse av metodene og deres input/output i en applikasjon

CSS
Cascading Stylesheet. Stilark som brukes til å utforme layouten for f.eks. en webside

DC
Dublin Core

DC-Lib
Dublin Core Library Application Profile

DCMES
Dublin Core Metadata Element Set

FDS
Fast Data Search

HTTP
Hypertext Transfer Protocol

LC
Library of Congress. USAs nasjonalbibliotek

MARC
Machine-Readable Cataloging

Metadata
Strukturerte data som er assosiert med informasjonsobjekter på en slik måte at potensielle brukere får kunnskap om objektenes eksistens eller karakteristikk

MODS
Metadata Object Description Schema, metadataformat for bibliografiske beskrivelser utviklet av Library of Congress

OAI datatilbyder
Aktør i OAI-PMH (også kalt datalagre eller åpne arkiver) som administrerer systemer/samlinger, og tilbyr fri tilgang til metadata gjennom OAI-PMH

OAI tjenestetilbyder
Aktør i OAI-PMH som bruker innhøstede metadata til å tilby tjenester, for eksempel søking

OAI-PMH
Open Archives Initiative Protocol for Metadata Harvesting

OCLC
Online Computer Library Center

PHP
PHP Hypertext Preprocessor. Serverskriptspråk som brukes til å lage mellomvare i webapplikasjoner

Repository
Datalager for metadata (relasjonsdatabase) og/eller for data (filsystem)

SOAP
Simple Object Access Protocol. SOAP gjør det mulig for programmer (som kjører på forskjellige operativsystemer) å kommunisere med hverandre ved å bruke HTTP og XML til å utveksle informasjon

SRU/SRW
Search/Retrieve URL/Web Service. Søkeprotokoll som muliggjør distribuert søking ved hjelp av web-teknologi

UML
Unified Modelling Language. Modelleringsspråk og notasjon som brukes til å designe objektorienterte programvaresystemer

XHTML
eXtensible HyperText Markup Language

XML
eXtensible Markup Language, markupspråk for definering av datastrukturer

XSLT
eXtensible Stylesheet Language Transformations. Språk for å uttrykke stilark, som definerer omforming av xml-filer

APPENDIKS A

OAI datatilbydere

OAI datatilbydere er en forutsetning for å kunne benytte OAI-PMH, og selv om dette prosjektet ikke har involvert direkte utvikling av datatilbydere, utgjør disse en del av helheten. I tilknytning til Nasjonalbibliotekets OAI datatilbydere presenteres det i dette appendikset:

- Generelle problemstillinger for OAI datatilbydere
- Problemstillinger relatert til OAI datalager
- Problemstillinger relatert til metadata i de opprinnelige samlingene
- Metadataoverganger fra strukturen i hver enkelt datakilde til metadataformatet MODS

De fleste problemstillingene ble oppdaget etter gjennomgang av om lag 30 tilfeldig utvalgte metadataposter fra hvert av de tre datalagrene.

A.1 Generelle problemstillinger for OAI datatilbydere

A.1.1 Metadataprefiks og navneromsprefiks

Metadataprefikset, det vil si termen som angir hvilket format metadataene som høstes skal være i, bør være kort men samtidig beskrivende. Dublin Core, som er standard metadataformat i OAI-PMH, har prefikset *oai_dc*, mens det foreløpig ikke er definert et standard metadataprefiks for MODS. Opprinnelig ble metadataprefikset *oai_mods* benyttet i Nasjonalbibliotekets OAI-datalagre, men ettersom Library of Congress (LC) bare benytter *mods* i sine OAI-datalagre, benyttes også denne formen i dette prosjektet.

I praksis betyr dette at en OAI-henvendelse har følgende form:
`http://nwa1.nb.no:8080/{dra|galnor|mavis}repository/servlet/OAIHandler?verb=ListRecords&`**`metadataPrefix=mods`**

Som en direkte følge av denne endringen, måtte også navneromsprefikset for metadataelementene endres fra oai_mods til mods.

A.1.2 MODS-versjon

Datatilbyderne eksporterte opprinnelig metadata som inneholdt en referanse til MODS xml schema versjon 3, mens mappingen i dette prosjektet er basert på versjon 2. Ettersom MODS versjon 3 er blitt utviklet etter at dette prosjektet startet, er det bestemt at datatilbyderne skal eksportere metadata som samsvarer med MODS versjon 2. De viktigste forskjellene mellom de to versjonene er beskrevet i *[MODS04]*.

A.1.3 Bugs

I tillegg til de ovenfornevnte problemstillingene, dukket det også opp et par småproblemer som skyldtes implementasjonsfeil. Disse ble raskt korrigert av Nasjonalbiblioteket.

- **OAI-identifikator.** oai:dra.nb.no:xxx ble opprinnelig brukt som oai-identifikator for OAI-datalageret til GalleriNOR, men ble korrigert til oai:galnor.nb.no:xxx.
- **Metadataelementnavn.** I Galleri NOR ble opprinnelig geographics istedenfor geographic (uten s) brukt som subelement til subject.

A.2 Problemstillinger relatert til OAI datalager

A.2.1 Usikkerhet rundt bruk av elementene namePart og role

Innledningsvis i utviklingsfasen var det usikkerhet knyttet til hvordan elementet *name* skulle brukes som et underelement til *subject*. Årsaken var at elementet *name* i metadataene fra Galleri NOR ble brukt til å beskrive avbildede bygninger, noe som virket i uoverensstemmelse med MODS-spesifikasjonene *[MUG]*. Disse sier at *name* kun skal brukes til å beskrive personer/organisasjoner/bedrifter.

Når typeattributtet *personal*, som er forbeholdt beskrivelser av personer, dessuten ble brukt til å beskrive en avbildet kirke, ble inntrykket av uoverensstemmelse forsterket.

ID: oai:galnor.nb.no:59000

```
<mods:name type="personal">
     <mods:namePart>Vang kirke</mods:namePart>
     <mods:role>
          <mods:text>Avbildet</mods:text>
     </mods:role>
</mods:name>
```

Årsaken til at attributtet type fikk verdien *personal*, var at de opprinnelige metadataene ikke inneholdt de nødvendige opplysningene for å skille forskjellige typer navn fra hverandre. Ytterligere informasjon om denne problemstillingen finnes i *kap. A.2.3, s. 113*.

Etter en henvendelse til e-postlisten for MODS, fikk jeg tilbakemelding fra en av utviklerne av formatet, *[GUE04]*, om at det ikke ligger noen uoverensstemmelse i å bruke elementet *name* som et underelement til *subject* som i eksempelet ovenfor. Å beskrive bygninger og andre motiver ved hjelp av navn og rolle ”avbildet” er i overensstemmelse med MARC21-standarden.

A.2.2 Usikkerhet rundt rollebeskrivelser

Enkelte av metadatapostene inneholder rollebeskrivelsen *sikker*, som kan være litt vanskelig å tolke betydningen av. Opplysningen kan kanskje være en form for bekreftelse på at den registrerte rollen er sikker, men kan også skape forvirring ettersom det er en metaopplysning om metadataene.

Eksempel fra GalleriNOR:

```
<!—Original oai:galnor.nb.no:59000 -->
<mods:name type="personal">
     <mods:namePart>Wilse, Anders Beer</mods:namePart>
     <mods:role>
          <mods:text>
               Produsent, tilvirker, fotograf, sikker
          </mods:text>
     </mods:role>
</mods:name>
```

I tillegg til usikkerheten rundt rollebeskrivelsene, er denne metadatabeskrivelsen dessuten et eksempel på at flere verdier er registrer inni det samme elementet. I henhold til MODS-standarden *[MUG]* burde derfor denne beskrivelsen hatt følgende format:

```
<mods:name type="personal">
     <mods:namePart>Wilse, Anders Beer</mods:namePart>
     <mods:role>
          <mods:text>Produsent</mods:text>
     </mods:role>
     <mods:role>
          <mods:text>tilvirker</mods:text>
     </mods:role>
     <mods:role>
          <mods:text>fotograf</mods:text>
     </mods:role>
     <!--
     <mods:role>
          <mods:text>sikker</mods:text>
     </mods:role>
     -->
</mods:name>
```

A.2.3 Attributtet type for elementet navn (personal | corporate)

Da de første metadataene ble høstet fra OAI datalagrene, oppstod det usikkerhet rundt hvordan de forskjellige navnetypene (personal, corporate, conference) kunne skilles. Innledende analyser av de innhøstede metadataene tydet på at alle navn var beskrevet som personal, det være seg person-, organisasjons- eller bedriftsnavn.

Nedenfor følger eksempler fra henholdsvis GalleriNOR og Mavis som illustrerer problemstillingen.

ID: oai:galnor.nb.no:59049

```
<mods:subject>
     <mods:topic>fabrikk</mods:topic>
     <mods:geographics>Oslo</mods:geographics>
     <mods:name type="personal">
          <mods:namePart>Lilleborg fabriker A/S</mods:namePart>
          <mods:role>
               <mods:text>Avbildet</mods:text>
          </mods:role>
     </mods:name>
</mods:subject>
```

ID: oai:mavis.nb.no:8757

```
<mods:name type="personal">
     <mods:namePart>Phillips Petroleum Company Norway</
mods:namePart>
     <mods:role>
          <mods:text>Produksjon/produsent</mods:text>
     </mods:role>
</mods:name>
```

Både Lilleborg fabrikker A/S og Philips Petroleum Company Norway er bedrifter, og attributtet type skulle derfor ha hatt verdien ”corporate”. Siden det ikke eksisterer noe skille mellom forskjellige typer navn i de opprinnelige metadataene, var det ikke mulig å nyttiggjøre typeattributtet. Løsningen ble i stedet å kutte ut typeattributtet i alle navnelementer (både i elementet name på toppnivå og i subelementet name under subject).

A.3 Problemstillinger relatert til opprinnelige metadata

A.3.1 Parenteser og forkortelser (i tittelinformasjon)

I flere av metadatabeskrivelsene finnes det forkortelser i tittelfeltet, for eksempel *prot* og *konv*.

Eksempler fra GalleriNOR

ID: oai:galnor.nb.no:59000

```
<mods:title>Prot: Hamar - Vangs kirke 6. Sep. 1902</mods:title>
```

ID: oai:galnor.nb.no:59499

```
<mods:title>Konv: tiur og røy</mods:title>
```

ID: oai:galnor.nb.no:59699

```
<mods:title>Prot: Horten - Harald Hårfagre i dokken med Værftet 3/
4 baug  Prot: 4. Oct. 1903</mods:title>
```

I ettertid har det vist seg at ved registrering av bilder så refererer teksten *prot* til at teksten er hentet fra en protokoll, mens teksten *konv* refererer til at teksten er hentet fra en konvolutt.

Eksempler fra DRA og Mavis

Enkelte av metadatapostene i DRA og Mavis inneholder opplysninger skrevet i parenteser i tittelen.

ID: oai:dra.nb.no:1994/01651.P

```
<mods:title>Kongeferden til Nord-Norge. [Bodø]</mods:title>
```

ID: oai:dra.nb.no:1994/22491.P

```
<mods:title>Om O E Rølvaag og hans dikting. Samtale med slektninger
på Dønna og i USA. (Råstoff)</mods:title>
```

ID: oai:mavis.nb.no:159992

```
<mods:title>JEKK OPP : [MEDIUM VERSJON]</mods:title>
```

Kommentarer om kvalitet i hakeparentes i tittelfeltet bør unngås og i stedet plasseres i et *note*-element. Siden dataene allerede er registrert på denne måten, er det lite som kan gjøres for å endre på det.

A.3.2 Opplysninger om navn/rolle i tittelfeltet

Opplysninger om navn/rolle og type program i tittelfeltet burde vært plassert et annet sted. Det er imidlertid ikke så mye som kan gjøres med dette ettersom de allerede er registrert på denne måten i de originale metadatabasene.

ID: oai:dra.nb.no:1996/02173.P

```
<mods:title>Lørdagskåseri: "Farvel, Mo i Rana - Velkommen, Mo i Ra-
na". Installatør Hans P Hansen</mods:title>
```

A.3.3 Flere verdier i ett og samme metadatafelt (name, role, subject)

Det finnes flere eksempler på at flere termer blir beskrevet innenfor samme element (kommaseparert i GalleriNOR, plusstegn i DRA) istedenfor i separate elementer. Dette gjelder spesielt *name* i DRA og *subject* i GalleriNOR. Det finnes også eksempler på at flere roller er beskrevet i ett og samme metadataelement. Dette er, i likhet med rollebeskrivelsene, en bekreftelse på at de originale metadataene ikke er registrert på optimal måte med tanke på videre bruk.

Å korrigere slike feilregistreringer er mulig, men vil være relativt ressurskrevende. I teorien er det mulig å lage et skript som utfører en slik konvertering, men i et prototypeprosjekt som dette vil gevinsten være liten. Både fordi metadataene senere blir konvertert til et flatt format (i hht. FastXML) og fordi man ikke må unnlate informasjon selv om de er registrert i en ikke-optimal struktur. Konklusjonen er derfor at dette ikke bør prioriteres, hovedsakelig på grunn av at gevinsten i forhold til antatt tidsforbruk vil være relativt liten.

Eksempler

ID: oai:galnor.nb.no:59005

```
<mods:topic>landskap, skog, mann</mods:topic>
```

bør være

```
<mods:topic>landskap</mods:topic>
<mods:topic>skog</mods:topic>
<mods:topic>mann</mods:topic>
```

ID: oai.dra.nb.no

```
<mods:name type="personal">
<mods:namePart>Westrheim, Harry + Berg, Lars</mods:namePart>
     <mods:role>
          <mods:text>programleder</mods:text>
     </mods:role>
</mods:name>
```

Bør være:

```
<mods:name type="personal">
<mods:namePart>Westrheim, Harry</mods:namePart>
     <mods:role>
          <mods:text>programleder</mods:text>
     </mods:role>
</mods:name>

<mods:name type="personal">
<mods:namePart>Berg, Lars</mods:namePart>
     <mods:role>
          <mods:text>programleder</mods:text>
     </mods:role>
</mods:name>
```

ID for metadataposter med dette problemet:

- oai:dra.nb.no:1990/00100.P
- oai:dra.nb.no:1990/00462.P
- oai:dra.nb.no:1990/00647.P (tre navn)
- oai:dra.nb.no:1994/01651.P

Problemet eksisterer også i Mavis, for eksempel i ID: oai:mavis.nb.no:83599:

```
<mods:language>Engelsk(Dubbet), Norsk(Originalspråk)</mods:lan-
guage>
```

Korrekte registringer

Det finnes imidlertid unntak, det vil si beskrivelser som er følger standarden.

I det korrekte eksempelet på en beskrivelse av flere navn som følger, bør imidlertid parentesen (prod.) etter det ene navnet fjernes, siden rollen produsent er beskrevet like under.

ID: oai:dra.nb.no:1990/00389.P

```
<mods:name type="personal">
     <mods:namePart>Havgar, Kristin (prod.)</mods:namePart>
     <mods:role>
          <mods:text>produsent</mods:text>
     </mods:role>
</mods:name>
<mods:name type="personal">
     <mods:namePart>Solli, Jens</mods:namePart>
     <mods:role>
          <mods:text>programleder</mods:text>
      </mods:role>
</mods:name>
```

A.3.4 Forskjellige måter å angi dato på

I de tre OAI-datalagrene eksisterer det mange forskjellige måter å beskrive datoer på. I *kap. 5.4.2, s. 70* beskrives de forskjellige datoformatene, samt hvilke datalagre de forskjellige formatene er benyttet i.

A.3.5 Inkonsekvent formattering av tidsintervaller (innad i et datalager)

Innad i DRA var det opprinnelig en inkonsekvent formatering av verdien for tidsintervaller i enkelte av metadataelementene *mods:extent*. I de fleste postene ble formatet *hh: mm: ss* benyttet, men i enkelte poster ble bindestrek benyttet på følgende måte *hh: mm:-ss*, for eksempel *00: 26:-27*. Problemet ble oppdaget i sju av de om lag 20 postene som ble undersøkt.

Denne uregelmessigheten i tidsformateringen skyldtes en liten bug hos Nasjonalbiblioteket. Etter at denne ble rettet opp, er derfor alle tidsintervaller i DRA beskrevet på en enhetlig måte, *hh:mm:ss*.

A.3.6 Redundans

I noen få metadatabeskrivelser fra Mavis eksisterer det redundante opplysninger, blant annet er den samme verdien for internetMediaType, Video/mpeg2, registrert to ganger.

ID: oai:mavis.nb.no:10450:

```
<physicalDesc>
     <internetMediaType>Video/mpeg2</internetMediaType>
     <internetMediaType>Video/mpeg2</internetMediaType>
     <extent>00:21:00</extent>
</physicalDesc>
```

Dette skyldes ifølge Nasjonalbiblioteket en feil i registreringen av nye metadata i Mavis.

A.4 Metadatoverganger mellom relasjonsdatabaser og MODS

Prinsippet om overganger til et felles metadataformat er beskrevet både i *kap. 2.5.1, s. 23* og i *kap. 5.2, s. 65*. Dette appendikset inneholder en visuell oversikt over hvilke tabellfelt i relasjonsdatabasene som er avbildet til hvilke elementer i metadataformatet MODS. Som det går fram av tabelle-

ne, utnyttes ikke alle feltene i MODS på grunn av at enkelte typer opplysninger ikke er registrert i de originale metadatabasene.

A.4.1 Digitalt radioarkiv

Tabell A-1. *Metadataovergang fra Digitalt Radioarkiv til MODS*

MODS (felles metadataformat)	Felt i DRA
<titleInfo>	
<title> *{tittel}* </title>	Tittel
<title type="alternative"> *{alternativ tittel}* </title>	
<subTitle> *{undertittel}* </subTitle>	
</titleInfo>	
<name type=" *{personal* | *corporate* | *conference}* ">	Fast verdi: personal (fjernet i ettertid)
<namePart> *{navn}* </namePart>	Navn
<role>	
<text> *{rollebeskrivelse}* </text>	Rolle
</role>	
</name>	
<typeOfResource> *{ressurstype}* </typeOfResource>	Fast tekst: sound recording
<genre> *{sjanger}* </genre>	Fast tekst: radioprogram
<originInfo>	
<publisher> *{utgiver}* </publisher>	Fast tekst: NRK
<dateCreated> *{dato}* </dateCreated>	Sendedato
</originInfo>	
<language authority=" *{rfc3066* | *iso639-2b}* "></language>	
<physicalDesc>	
<internetMediaType> *{MIME-type}* </internetMediaType>	[Brukes ikke i DRA]
<extent> *{verdi}* </extent>	Tid
</physicalDesc>	
<abstract> *{fritekstbeskrivelse}* </abstract>	Innholdsfeltet
<note> *{bemerkning}* </note>	[Brukes ikke i DRA]
<subject>	[Brukes ikke i DRA]
<topic> *{generelt emne}* </topic>	

Tabell A-1. *Metadataovergang fra Digitalt Radioarkiv til MODS (forts.)*

MODS (felles metadataformat)	Felt i DRA
<geographic> *{geografisk emne}* </geographic>	
<temporal> *{tidsmessig emne}* </temporal>	
<name type=" *{personal* | *corporate* | *conference}* ">	
<namePart></namePart>	
<role>	
<text></text>	
</role>	
</name>	
</subject>	
<relatedItem type="original">	[Brukes ikke i DRA]
<identifier type="..."> *{id}* </identifier>	
</relatedItem>	
<identifier type="..."> *{id}* </identifier>	URN
<accessCondition> *{fritekstbeskrivelse}* </accessCondition>	Fast tekst: Copyright NRK

A.4.2 Galleri NOR

Tabell A-2. *Metadataovergang fra Galleri NOR til MODS*

MODS (felles metadataformat)	Felt i Galleri NOR
<titleInfo>	
<title> *{tittel}* </title>	Foto -> Tittel
<title type="alternative"> *{alternativ tittel}* </title>	
<subTitle> *{undertittel}* </subTitle>	
</titleInfo>	
<name type=" *{personal* | *corporate* | *conference}* ">	Fast verdi: personal (fjernet i ettertid)
<namePart> *{navn}* </namePart>	Juridisk_Person -> Navn
<role>	
<text> *{rollebeskrivelse}* </text>	Rolle -> Beskrivelse
</role>	
</name>	

Tabell A-2. *Metadataovergang fra Galleri NOR til MODS (forts.)*

MODS (felles metadataformat)	Felt i Galleri NOR
<typeOfResource> *{ressurstype}* </typeOfResource>	Fast tekst: still image
<genre> *{sjanger}* </genre>	Fast tekst: fotografi
<originInfo>	
<publisher> *{utgiver}* </publisher>	
<dateCreated> *{dato}* </dateCreated>	Foto -> Datering_fra, Datering_til
</originInfo>	
<language authority=" *{rfc3066* | *iso639-2b}* "></language>	[Brukes ikke i GN]
<physicalDesc>	
<internetMediaType> *{MIME-type}* </internetMediaType>	Fast tekst: image/jpeg ??
<extent> *{verdi}* </extent>	[Brukes ikke i GN]
</physicalDesc>	
<abstract> *{fritekstbeskrivelse}* </abstract>	Foto -> Utfyllende info
<note> *{bemerkning}* </note>	[Brukes ikke i GN]
<subject>	
<topic> *{generelt emne}* </topic>	Foto -> Motiv
<geographic> *{geografisk emne}* </geographic>	Sted -> Presisering
<temporal> *{tidsmessig emne}* </temporal>	
<name type=" *{personal* | *corporate* | *conference}* ">	Fast verdi: personal (fjernet i ettertid)
<namePart></namePart>	Juridisk_Person -> Navn
<role>	
<text></text>	Avbildet
</role>	
</name>	
</subject>	
<relatedItem type="original">	
<identifier type="..."> *{id}* </identifier>	Foto -> Invnr
</relatedItem>	
<identifier type="..."> *{id}* </identifier>	Foto -> Repronr
<accessCondition> *{fritekstbeskrivelse}* </accessCondition>	Fast tekst: Eier: Norsk Folkemuseum. Kopiering og publisering kun etter avtale.

A.4.3 Mavis

Tabell A-3. *Metadataovergang fra Mavis til MODS*

MODS (felles metadataformat)	Felt i Mavis
<titleInfo>	
<title> *{tittel}* </title>	TITLE_DESC
<title type="alternative"> *{alternativ tittel}* </title>	ALTERNATE TITLE + ALT_TITLE_PURPOSE
<subTitle> *{undertittel}* </subTitle>	
</titleInfo>	
<name type=" *{personal* | *corporate* | *conference}* ">	Fast verdi: personal (fjernet i ettertid)
<namePart> *{navn}* </namePart>	NAME
<role>	
<text> *{rollebeskrivelse}* </text>	ROLE + ROLE_NOTE
</role>	
</name>	
<typeOfResource> *{ressurstype}* </typeOfResource>	Fast tekst: moving image
<genre> *{sjanger}* </genre>	Fast tekst: film
<originInfo>	
<publisher> *{utgiver}* </publisher>	NAME
<dateCreated> *{dato}* </dateCreated>	OBJECT_DATE_TYPE = PRODUCTION
</originInfo>	
<language authority=" *{rfc3066* | *iso639-2b}* "></language>	ITEM.LANGUAGE
<physicalDesc>	
<internetMediaType> *{MIME-type}* </internetMediaType>	FILE_FORMAT
<extent> *{verdi}* </extent>	DURATION_HR, DURATION_MIN, DURATION_SEC
</physicalDesc>	
<abstract> *{fritekstbeskrivelse}* </abstract>	TITLE_INDEX_POINT.SUMMARY
<note> *{bemerkning}* </note>	[Brukes ikke i Mavis]
<subject>	[Brukes ikke i Mavis]
<topic> *{generelt emne}* </topic>	
<geographic> *{geografisk emne}* </geographic>	
<temporal> *{tidsmessig emne}* </temporal>	

Tabell A-3. *Metadataovergang fra Mavis til MODS (forts.)*

MODS (felles metadataformat)	Felt i Mavis
<name type=" *{personal* | *corporate* | *conference}* ">	
<namePart></namePart>	
<role>	
<text></text>	
</role>	
</name>	
</subject>	
<relatedItem type="original">	[Brukes ikke i Mavis]
<identifier type="..."> *{id}* </identifier>	
</relatedItem>	
<identifier type="..."> *{id}* </identifier>	OBJECT_RESOURCE_IDENTIFIER.RESOURCE_IDENTIFIER
<accessCondition> *{fritekstbeskrivelse}* </accessCondition>	[Brukes ikke i Mavis]

APPENDIKS B

OAIHarvester2

Java-applikasjonen OAIHarvester2 *[OAIH]* er et "åpen kildekode"-prosjekt som er utviklet av OCLC (Online Computer Library Center). De viktigste egenskapene til OAIHarvester2 er:

- Innebygd støtte både for OAI-PMH v1.1 og v2.0.
- Inkluderer et fungerende eksempel på en OAI-høster som høster inn metadata og lagrer disse som xml-filer.
- Javaklassene kan også brukes som utgangspunkt for andre Java-applikasjoner.

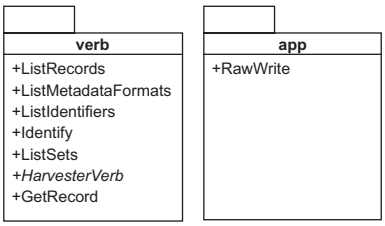
For å få en grundigere forståelse OAIHarvester2, beskriver dette appendikset:

- Oppbygning av original OAIHarvester2
- Utvidelser av OAIHarvester2
- Beskrivelse av Java-kode
- Java-kode
- Skript for å kjøre OAIHarvester2

B.1 Oppbygging av original OAIHarvester2

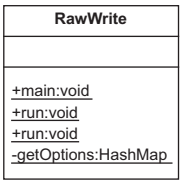
For å få oversikt over oppbygningen til OAIHarvester2, er det gjennomført en omvendt utvikling med utgangspunkt i den eksisterende kildekoden. Hovedresultatet av den omvendte utviklingsprosessen er UML-diagrammer som gir en visuell oversikt over oppbygningen i den eksisterende applikasjonen. Modellerings- og utviklingsverktøyet Together Control Center er benyttet til denne oppgaven.

Hovedpakken i høsterapplikasjonen er harvester2:



Figur B-1. *UML-diagram for pakken harvester2*

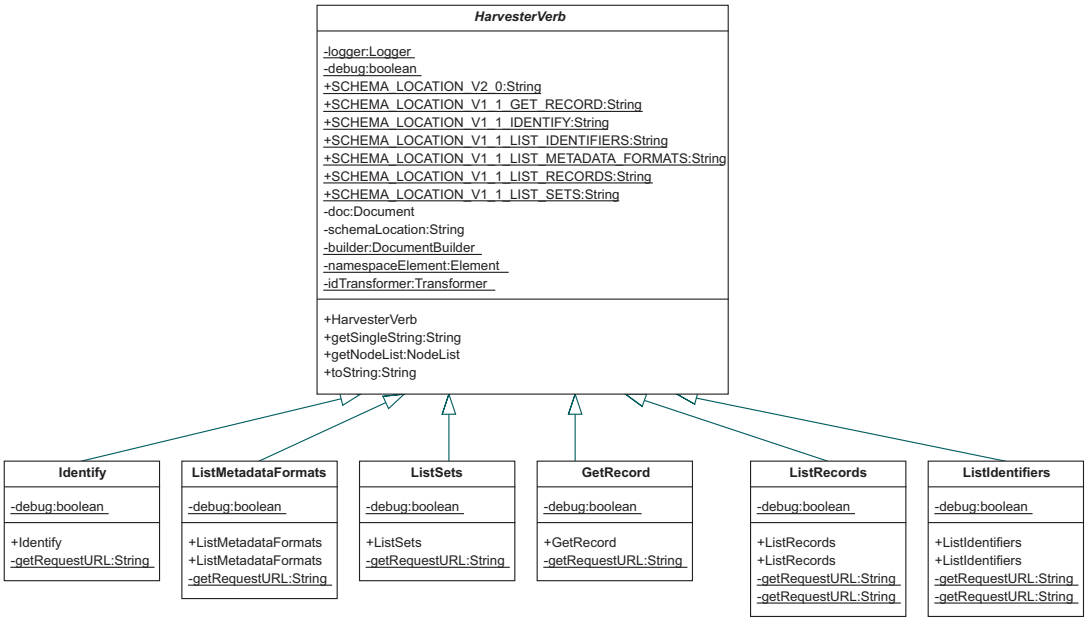
OAIHarvester2 har en relativt enkel oppbygning. Pakken app inneholder applikasjonsklassen `RawWrite`, som er et fungerende eksempel på en OAI-høster.



Figur B-2. *UML-diagram for pakken app*

Når `RawWrite` gjennomfører en høsting, kaller main-metoden de nødvendige OAI-verbene som er definert i pakken `verb`. Verb-pakken inneholder de klassene som er nødvendige for å kunne oppfylle funksjonalitetskravene for en generell OAI-høster slik de er definert i de offisielle retningslinjene for implementasjon av OAI-høstere *[OAIHG]*. I tillegg til en generell superklasse, `HarvesterVerb`, som inneholder funksjonalitet som er felles for alle verbene, er hvert av de seks OAI-verbene definert i egne klasser:

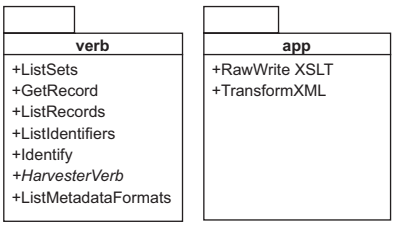
- Identify
- GetRecord
- ListSets
- ListRecords
- ListIdentifiers
- ListMetadataFormats



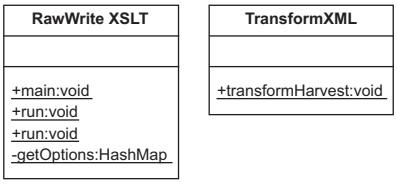
Figur B-3. *UML-diagram for pakken verb*

B.2 Utvidelser av OAIHarvester2

I dette prosjektet er det gjennomført en utvidelse av OAIHarvester2 som gjør høsteren i stand til å omforme de innhøstede metadataene ved hjelp av stilark. Prinsippet bak utvidelsen er at det, i tillegg til de eksisterende klassene, er behov for en ny klasse som kan omforme xml-filer som inneholder innhøstede metadata. I figur B-4 og B-5 er denne utvidelsen representert ved hjelp av klassen `TransformXML` i pakken `app`. Klassen `RawWrite` er dessuten endret slik at den kan anvende den metodene i den nye klassen, og dette reflekteres i det nye klassenavnet `RawWrite_XSLT`.



Figur B-4. *UML-diagram for utvidet pakke harvester2*



Figur B-5. *UML-diagram for utvidet pakke app*

B.3 Beskrivelse av Java-kode

Klassen baserer seg på transformasjon av statiske xml-data lagret i fil, siden dette er den enkleste måten å gjennomføre en xsl-transformasjon på. Det er ikke tatt hensyn til effektivitetsspørsmål i utviklingen av denne funksjonaliteten. Klassen har to inputparametere av typen String som representerer filnavnene til xml-filen og xslt-filen, og klassens eneste metode, `transformHarvest`, bruker disse to filnavnene til å åpne de aktuelle filene. Ved å benytte den innebygde funksjonaliteten for xsl-transformering i JAXP (Java API for XML Processing), omformes xml-kildefilen i henhold til beskrivelsene i xslt-filen. Resultatene blir lagret i en fil som navngis med en kombinasjon av den originale xml-filens navn og stilarkets navn.

Når høsteren for eksempel høster metadata fra Digitalt Radioarkiv (DRA), blir de innhøstede metadataene først lagret i xml-filen `dra.xml`. Deretter starter omformingen av denne filen i henhold til omformingsreglene beskrevet i stilarket `FastXML.xslt`. Resultatet er en ny xml-fil med navnet `dra_FastXML.xml`, som inneholder de omformede metadataene.

I tillegg gis det tilbakemeldinger på skjerm om hva som foregår.

`TransformXML` er imidlertid bare en hjelpeklasse til `RawWrite`, og for å kunne ta i bruk funksjonaliteten i `TransformXML` må main-metoden i `RawWrite` omskrives. Endringene som er utført er:

- Legge til støtte for en ekstra inputparameter for spesifisering av at det skal foregå en transformasjon, samt navnet på xslt-filen som inneholder reglene for transformasjonen.
- Sjekke om parameterene `-out` og `-xslt` er satt, det vil si om de innhøstede metadataene skal lagres i en fil (out) og om det skal utføres en omforming av innholdet i filen etter at høstingen er ferdig (xslt). Hvis begge parameterene er satt, skal det opprettes et objekt av type `TransformXML` som iverksetter metoden `transformHarvest`.

Et eksempel på bruk av den utvidete harvesteren er:

```
java
     -classpath log4j.jar:harvester2.jar
     ORG.oclc.oai.harvester2.app.RawWrite
     -out mods_test.xml
```

```
          -metadataPrefix mods
          -xslt FastXML.xsl
          http://www.stud.ntnu.no/~eskilhoy/OAI-XMLFile-2.1/XMLFile/
     test_mods/oai.cgi
```

Endringen i forhold til utgangspunktet er markert med fet skrift.

B.4 Javakode

Dette delkapittelet inneholder Javakoden som er endret eller lagt til OAIHarvester2 i dette prosjektet. Klasser som det ikke er gjort endringer i, er ikke tatt med i dette appendikset. All kode for den originale OAIHarvester2 er tilgjengelig på *[OAIH]*.

B.4.1 Utvidet OAIHarvester2

```
/*********************
* RawWrite_XSLT.java *
*********************/
*The Original Code is RawWrite.java.
*The Initial Developer of the Original Code is Jeff Young.
*Copyright (c) 2000-2002 OCLC Online Computer Library Center
*Portions created by Eskil Høyen Solvang are
*Copyright (C) 2003. All Rights Reserved.
*/

package harvester2.app;

import java.io.*;
import java.lang.NoSuchFieldException;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.HashMap;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.TransformerException;
import harvester2.verb.*;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

public class RawWrite_XSLT {
   public static void main(String[] args) {
   try {
      System.out.println(new Date());

         HashMap options = getOptions(args);
         List rootArgs = (List)options.get("rootArgs");
      String baseURL = null;
         if (rootArgs.size() > 0) {
            baseURL = (String)rootArgs.get(0);
         } else {
            throw new IllegalArgumentException();
         }

         OutputStream out = System.out;
         String outFileName = (String)options.get("-out");
         String from = (String)options.get("-from");
```

```
        String until = (String)options.get("-until");
        String metadataPrefix = (String)options.get("-metadataPrefix");
        if (metadataPrefix == null) metadataPrefix = "oai_dc";
        String resumptionToken = (String)options.get("-resumptionToken");
        String setSpec = (String)options.get("-setSpec");
        //Read input argument -xslt (added 21.11.2003)
        String xsltStylesheet = (String)options.get("-xslt");

        if (resumptionToken != null) {
           if (outFileName != null)
           out = new FileOutputStream(outFileName, true);
           run(baseURL, resumptionToken, out);
        } else {
           if (outFileName != null)
           out = new FileOutputStream(outFileName);
           run(baseURL, from, until, metadataPrefix, setSpec, out);
        }

        if (out != System.out) out.close();
     System.out.println(new Date());

     //XSLT transformation (start)
     if ((outFileName != null) && (xsltStylesheet != null)) {
        System.out.println("starting xsl transformation: "
                           + outFileName
                           + ", "
                           + xsltStylesheet);
        //instantiate converter object
        TransformXML converter = new TransformXML();
        //do xsl transformation
        converter.transformHarvest(outFileName, xsltStylesheet);
        System.out.println("xsl transformation succeeded: "
                           + outFileName
                           + ", "
                           + xsltStylesheet);
     //XSLT transformation (end)}

     } catch (IllegalArgumentException e) {
        System.err.println("RawWrite <-from date> <-until date> <-metadataPrefix
           prefix> <-setSpec setName> <-resumptionToken token> <-out fileName>
           <-xslt stylesheetName> baseURL");
   } catch (Exception e) {
        e.printStackTrace();
        System.exit(-1);
   }
   }

   public static void run(String baseURL, String resumptionToken, OutputStream
out)
     throws IOException, ParserConfigurationException, SAXException, Trans-
formerException,NoSuchFieldException {
     ListRecords listRecords = new ListRecords(baseURL, resumptionToken);
     while (listRecords != null) {
        NodeList errors = listRecords.getErrors();
        if (errors != null && errors.getLength() > 0) {
           System.out.println("Found errors");
           int length = errors.getLength();
           for (int i=0; i<length; ++i) {
              Node item = errors.item(i);
```

```
            System.out.println(item);
          }
          System.out.println("Error record: " + listRecords.toString());
          break;
        }
        //System.out.println(listRecords);
        out.write(listRecords.toString().getBytes("UTF-8"));
        out.write("\n".getBytes("UTF-8"));
        resumptionToken = listRecords.getResumptionToken();
        System.out.println("resumptionToken: " + resumptionToken);
        if (resumptionToken == null || resumptionToken.length() == 0) {
          listRecords = null;
        } else {
          listRecords = new ListRecords(baseURL, resumptionToken);
        }
      }
      out.write("</harvest>\n".getBytes("UTF-8"));
    }

    public static void run(String baseURL, String from, String until,String meta-
    dataPrefix, String setSpec,OutputStream out)
      throws IOException, ParserConfigurationException, SAXException, Trans-
      formerException,NoSuchFieldException {
      out.write("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n".getBytes("UTF-
      8"));
      out.write("<harvest>\n".getBytes("UTF-8"));
      out.write(new Identify(baseURL).toString().getBytes("UTF-8"));
      out.write("\n".getBytes("UTF-8"));
      out.write(new ListMetadataFormats(baseURL).toString().getBytes("UTF-8"));
      out.write("\n".getBytes("UTF-8"));
      out.write(new ListSets(baseURL).toString().getBytes("UTF-8"));
      out.write("\n".getBytes("UTF-8"));
      ListRecords listRecords = new ListRecords(baseURL, from, until, set-
      Spec,metadataPrefix);
      while (listRecords != null) {
        NodeList errors = listRecords.getErrors();
        if (errors != null && errors.getLength() > 0) {
          System.out.println("Found errors");
          int length = errors.getLength();
          for (int i=0; i<length; ++i) {
          Node item = errors.item(i);
          System.out.println(item);
          }
          System.out.println("Error record: " + listRecords.toString());
          break;
        }
        //System.out.println(listRecords);
        out.write(listRecords.toString().getBytes("UTF-8"));
        out.write("\n".getBytes("UTF-8"));
        String resumptionToken = listRecords.getResumptionToken();
        System.out.println("resumptionToken: " + resumptionToken);
        if (resumptionToken == null || resumptionToken.length() == 0) {
          listRecords = null;
        } else {
          listRecords = new ListRecords(baseURL, resumptionToken);
        }
      }
      out.write("</harvest>\n".getBytes("UTF-8"));
    }
```

```
   private static HashMap getOptions(String[] args) {
      HashMap options = new HashMap();
      ArrayList rootArgs = new ArrayList();
      options.put("rootArgs", rootArgs);

      for (int i=0; i<args.length; ++i) {
         if (args[i].charAt(0) != '-') {
            rootArgs.add(args[i]);
         } else if (i+1 < args.length) {
            options.put(args[i], args[++i]);
         } else {
            throw new IllegalArgumentException();
         }
      }
      return options;
   }
}
```

B.4.2 XML-omforming

```
/********************
* TransformXML.java *
********************/

package harvester2.app;

import java.io.*;
import javax.xml.transform.*;
import javax.xml.transform.stream.*;

/**
 * This class transforms xml input to some other format in accordance with an
 * xslt stylesheet.
 *
 * @author Eskil H. Solvang, NTNU-IDI.
 */
public class TransformXML {

   /**
      * Method for transforming xml harvest file
      *
      * @param xmlSource the xml file containing input data.
      * @param xsltSource the xslt styleshee file.
      * @exception IOException an I/O error occurred.
      * @exception TransformerException a transformation error occured.
      */
   public static void transformHarvest(String xmlFileName, String xsltFileName)
      throws IOException, TransformerException {

      File xmlFile = new File(xmlFileName);
      File xsltFile = new File(xsltFileName);

      //transformation sources
      Source xmlSource = new StreamSource(xmlFile);
      Source xsltSource = new StreamSource(xsltFile);

      //strip file endings from input file names, generate output file name
      String newName = "default";
      if ((xmlFileName.endsWith(".xml")) && (xsltFileName.endsWith(".xsl"))) {
         String xmlStrip = xmlFileName.substring(0,xmlFileName.indexOf(".xml"));
```

```
        String xsltStrip = xsltFileName.substring(0,xsltFileName.index-
Of(".xsl"));
        newName = xmlStrip + "_" + xsltStrip;
      }

      //send the result to a file (and display it on screen)
      File resultFile = new File(newName + ".xml");
      Result result_file = new StreamResult(resultFile);

      //get a transformer factory
      TransformerFactory transFact = TransformerFactory.newInstance();
      //get a transformer for this particular stylesheet
      Transformer trans = transFact.newTransformer(xsltSource);

      //do the transformation
      trans.transform(xmlSource, result_file);
   }
}
```

B.5 Skript for å kjøre OAIHarvester2

En skriptfil (shell/batch) inneholder instruksjoner for hvordan innhøstingen skal foregå for de forskjellige datatilbyderne, det vil si hvilke OAI datalagre det skal høstes metadata fra og i hvilken rekkefølge dette skal gjøres. Denne skriptfilen gjør det mulig å starte høstingen fra alle OAI datatilbyderne ved hjelp av én kommando.

```
REM Harvesting batch file (GalleriNOR, DRA, Mavis)

java -classpath "log4j.jar;harvester2.jar" harvester2.app.RawWrite_XSLT -out
dra.xml -xslt FastXML.xsl -metadataPrefix mods http://nwa1.nb.no:8080/drarepos-
itory/servlet/OAIHandler

java -classpath "log4j.jar;harvester2.jar" harvester2.app.RawWrite_XSLT -out
galnor.xml -xslt FastXML.xsl -metadataPrefix mods http://nwa1.nb.no:8080/galnor-
repository/servlet/OAIHandler

java -classpath "log4j.jar;harvester2.jar" harvester2.app.RawWrite_XSLT -out
mavis.xml -xslt FastXML.xsl -metadataPrefix mods http://nwa1.nb.no:8080/mavis-
repository/servlet/OAIHandler
```

APPENDIKS C

Metadataomforming

Stilark (xslt) benyttes til å gjennomføre metadataomforming fra MODS-struktur til FastXML-struktur, og til å gjøre en enkel normalisering av de innhøstede metadataene. I dette appendikset beskrives:

- MODS versjon 2.0 xml-skjema
- FastXML DTD
- Beskrivelse av stilark
- Flytskjema for datokonvertering
- Kode for stilark

C.1 MODS versjon 2.0 xml-skjema

Alle metadatabeskrivelsene som skal omformes ved hjelp av stilarket har en struktur som samsvarer med xml-skjemaet for MODS versjon 2.0. Dette finnes i sin helhet på *[MODX]*. Grunnstrukturen i MODS-dokumentene er basert på de 19 elementene som presenteres i *kap. 2.2.4, s. 13*. Flere av elementene har dessuten hierarkiske substrukturer.

C.2 FastXML DTD

Metadataomformingen skal resultere i xml-dokumenter som er i samsvar med dokumenttypedefinisjonen for FastXML:

```
<!-- FAST Document Format 1.0 -->
<!ELEMENT document (element)+>

<!ELEMENT element (meta-info*, value)>
<!ATTLIST element
          name CDATA #REQUIRED>

<!ELEMENT meta-info EMPTY>
<!ATTLIST meta-info
          name CDATA #REQUIRED
          value CDATA #REQUIRED>
```

```
<!ELEMENT value (#PCDATA)>

<!-- Predefined built-in entities -->
<!ENTITY lt "&#38;#60;">
<!ENTITY gt "&#62;">
<!ENTITY amp "&#38;#38;">
<!ENTITY apos "&#39;">
<!ENTITY quot "&#34;">
```

C.3 Beskrivelse av stilark

Den overordnede strukturen i stilarket er:

- Én hovedmal identifiserer strukturen i hele det opprinnelige metadatadokumentet, som kan inneholde mange metadatabeskrivelser
- Én metadatamal idenfiserer hver enkelt metadatabeskrivelse, som inneholder mange metadataelementer
- For hvert enkelt metadataelement, for eksempel *titleInfo* eller *abstract*, eksisterer det en egen elementmal som definerer hvordan det aktuelle elementet skal omformes
- For metadataelementer som inneholder subelementer, for eksempel *subject* med subelementer som *topic, geographic* og *temporal*, må det utformes elementmaler som identifiserer hvert av subelementene, og som definerer hvordan disse skal omformes

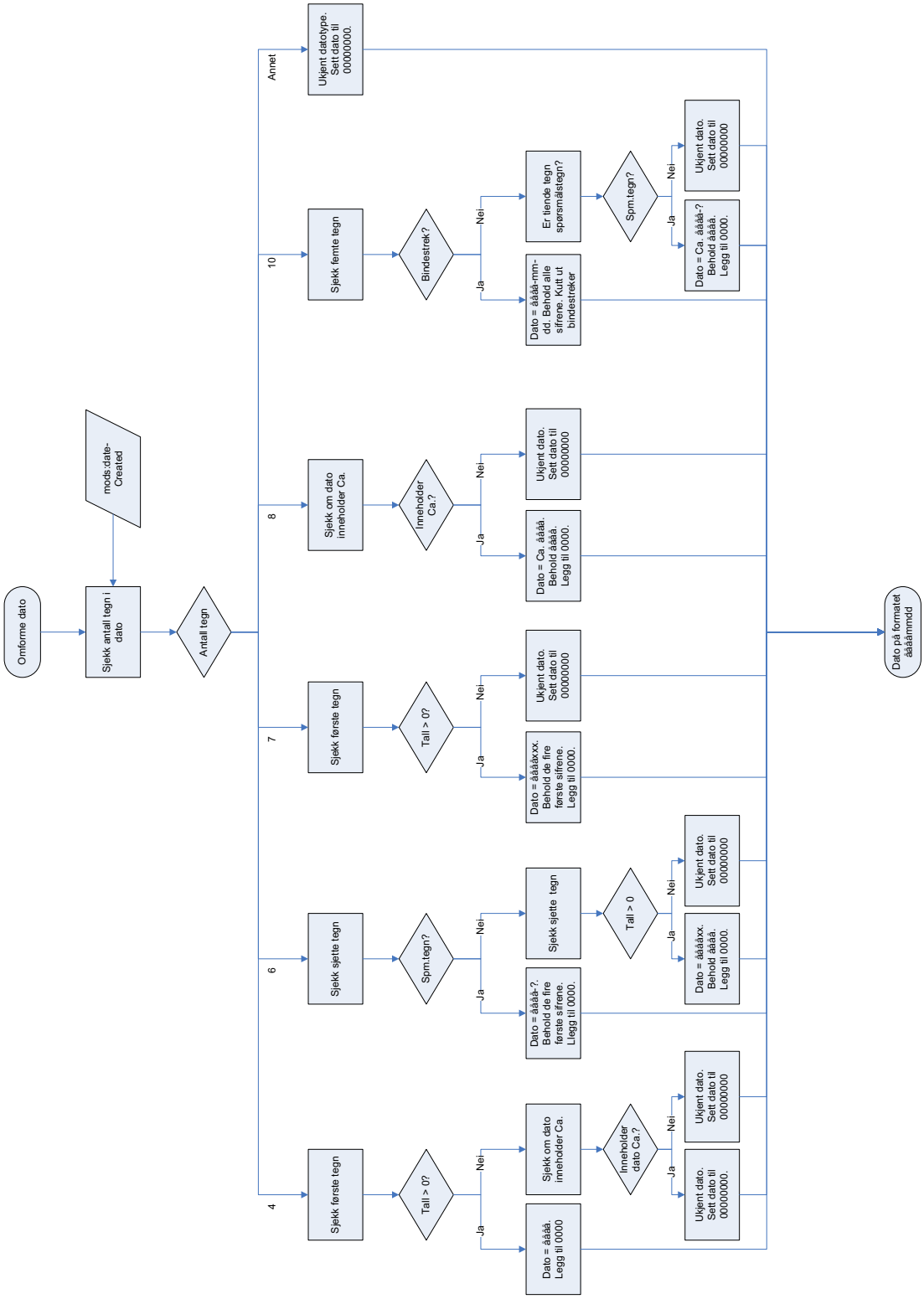
På grunn av at MODS-formatet har en oppbygning som tillater repeterende elementer i forskjellige sammenhenger, må det lages omformingsmaler som kan takle slike situasjoner. Et eksempel på repeterende elementer er at elementet *name* både brukes til å beskrive navn (og rolle) som er direkte knyttet til ressursen (elementet name er på toppnivå) i tillegg til at det kan brukes som et subelement til *subject*. For å kunne prosessere elementet *name* på forskjellig måte i disse to sammenhengene, benyttes mekanismen template-modi (template modes). Dette innebærer at det må defineres forskjellige templates for det samme elementet avhengig av hvilket modi av elementet som skal benyttes.

I Fast-dokumentasjonen anbefales det at hvert elementnavn gis et prefiks, slik at det ikke oppstår forviklinger i forhold til forhåndsdefinerte interne navn. Alle metadataelementene i dette prosjektet har derfor fått prefikset *mods*, men dette kan endres ved behov siden prefikset er definert som en variabel i stilarket.

For å koble sammen lokalt elementnavn med prefiks, brukes XPath-funksjonen `concat()` med variabelnavn og lokalt navn som inputparametere:

```
<element name="{concat($prefix, local-name(.))}">
```

C.3.1 Flytskjema for datokonvertering



Figur C-1. *Flytskjema for datokonvertering*

C.4 Kode for stilark

```
<?xml version="1.0" encoding="UTF-8"?>
<!--==============================
Document: MODS_FastXML.xsl
Created on: 29. november 2003
Last edited: 5. mars 2004
Author: Eskil Høyen Solvang
Description:
   Omformer metadata fra xml-formatet MODS til formatet FastXML
   Konverterer datoer fra diverse formater til et felles format ååååmmdd
==============================-->

<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:oai="http://www.openarchives.org/OAI/2.0/"
   xmlns:mods="http://www.loc.gov/mods/">

   <!-- Definerer en global variabel, prefix, som brukes som prefiks til alle nye
   elementnavn (i hht. anbefalinger i Fast-dokumentasjon) -->
   <!-- Variabelen refereres v.h.a. <xsl:value-of select="$prefix"/> -->
   <xsl:variable name="prefix" select="'mods.'"/>

   <!-- Angir outputtype, samt om denne skal formatteres med innrykk. Attributtet
   doctype-system="URI til FastXML DTD" er lagt til slik at output kan valideres
   mot dtd -->
   <xsl:output
      method="xml"
      version="1.0"
      encoding="iso-8859-1"
      indent="yes"
      doctype-system="FastXML.dtd"/>

<!--========== Template:main ======== -->
   <xsl:template match="/">
      <!-- Struktur i nytt FastXML-dokument: rotelement = documents -->
      <documents>
         <!-- Elementet ListRecords fører til anvendelse av en template -->
         <xsl:apply-templates select="harvest/oai:OAI-PMH/oai:ListRecords"/>
      </documents>
   </xsl:template>

<!--========= Template:record =========-->
   <xsl:template match="oai:record">
      <!-- Struktur i ny metadatapost -->
      <document id="{oai:header/oai:identifier}">
         <!-- OAI-identifikator -->
         <element name="oaiid">
            <value><xsl:value-of select="oai:header/oai:identifier"/></value>
         </element>
         <!-- OAI-opprinnelse. Bruker kombinasjon av xpath-funksjonene substring-
         before/-after. -->
         <element name="oaisource">
            <xsl:variable name="v_oaiid" select="oai:header/oai:identifier"/>
            <value><xsl:value-of select="substring-before(substring-af-
            ter($v_oaiid,'oai:'),'.nb')"/></value>
         </element>
         <!-- Referanse til ny template for å hente ut metadata-elementene -->
         <xsl:apply-templates select="oai:metadata/mods:mods"/>
      </document>
```

```
   </xsl:template>

<!--========= Metadataelementer, Toppnivå ========-->
   <!-- titleInfo -->
   <xsl:template match="mods:titleInfo">
      <!-- Subelementene title og subTitle skal prosesseres videre-->
      <xsl:apply-templates select="mods:title | mods:subTitle"/>
   </xsl:template>

   <!-- name -->
   <xsl:template match="mods:name">
      <!-- Subelementene namePart og role skal prosesseres videre-->
      <xsl:apply-templates select="mods:namePart | mods:role"/>
   </xsl:template>

   <!-- typeOfResource -->
   <xsl:template match="mods:typeOfResource">
      <!-- Lager et nytt element mods.typeOfResource. Kopierer verdi fra opprin-
      nelig dokument -->
      <element name="{concat($prefix, local-name(.))}">
         <value><xsl:value-of select="."/></value>
      </element>
   </xsl:template>

   <!-- genre -->
   <xsl:template match="mods:genre">
      <!-- Lager et nytt element mods.genre. Kopierer verdi fra opprinnelig do-
      kument -->
      <element name="{concat($prefix, local-name(.))}">
         <value><xsl:value-of select="."/></value>
      </element>
   </xsl:template>

   <!-- originInfo -->
   <xsl:template match="mods:originInfo">
      <!-- Subelementene publisher og dateCreated skal prosesseres videre -->
      <xsl:apply-templates select="mods:publisher | mods:dateCreated"/>
   </xsl:template>

   <!-- language -->
   <xsl:template match="mods:language">
      <!-- Lager et nytt element mods.language. Kopierer verdi fra opprinnelig
      dokument -->
      <element name="{concat($prefix, local-name(.))}">
         <value><xsl:value-of select="."/></value>
      </element>
   </xsl:template>

   <!-- physicalDescription -->
   <xsl:template match="mods:physicalDescription">
      <!-- Subelementene iMT og extent skal prosesseres videre -->
      <xsl:apply-templates select="mods:internetMediaType | mods:extent"/>
   </xsl:template>

   <!-- abstract -->
   <xsl:template match="mods:abstract">
      <!-- Lager et nytt element mods.abstract. Kopierer verdi fra opprinnelig
      dokument -->
      <element name="{concat($prefix, local-name(.))}">
```

```
        <value><xsl:value-of select="."/></value>
      </element>
   </xsl:template>

   <!-- subject -->
   <xsl:template match="mods:subject">
      <!-- Subelementene topic, geographic, temporal skal prosesseres videre -->
      <xsl:apply-templates select="mods:topic | mods:geographic | mods:tempo-
      ral"/>
      <!-- Bruker attributtet mode for å angi at elementet mods:name skal behan-
      dles annerledes når det er et subelement i subject. Dette krever en egen
      template -->
      <xsl:apply-templates select="mods:name/mods:namePart | mods:name/
      mods:role" mode="subject"/>
   </xsl:template>

   <!-- relatedItem -->
   <xsl:template match="mods:relatedItem">
      <!-- Bruker attributtet mode for å angi at elementet mods:identifier skal
      behandles annerledes når det er subelement i relatedItem. Dette krever en
      egen template -->
      <xsl:apply-templates select="mods:identifier" mode="relatedItem"/>
   </xsl:template>

   <!-- identifier -->
   <xsl:template match="mods:identifier">
      <!-- Lager et nytt element mods.identifier. Kopierer verdi fra opprinnelig
      dokument -->
      <!-- meta-info beskriver hva slags type identifikator som benyttes -->
      <element name="{concat($prefix, local-name(.))}">
         <meta-info name="type" value="{@type}"/>
         <value><xsl:value-of select="."/></value>
      </element>
   </xsl:template>

   <!-- accessCondition -->
   <xsl:template match="mods:accessCondition">
      <!-- Lager et nytt element mods.accessCondition. Kopierer verdi fra opprin-
      nelig dokument -->
      <element name="{concat($prefix, local-name(.))}">
         <value><xsl:value-of select="."/></value>
      </element>
   </xsl:template>

<!--========= Metadataelementer, Subelementer ========-->
   <!-- title (titleInfo)-->
   <xsl:template match="mods:title">
      <!-- IF type=alternative THEN meta-info-element ELSE ingen meta-info -->
      <xsl:choose>
         <xsl:when test="@type='alternative'">
            <element name="{concat($prefix, local-name(.))}">
            <meta-info name="type" value="{@type}"/>
            <value>
               <xsl:value-of select="."/>
            </value>
            </element>
         </xsl:when>
         <xsl:otherwise>
            <element name="{concat($prefix, local-name(.))}">
```

```
        <value>
           <xsl:value-of select="."/>
        </value>
        </element>
      </xsl:otherwise>
   </xsl:choose>
</xsl:template>

<!-- subTitle (titleInfo) -->
<xsl:template match="mods:subTitle">
   <element name="{concat($prefix, local-name(.))}">
      <value>
         <xsl:value-of select="."/>
      </value>
   </element>
</xsl:template>

<!-- namePart (name) -->
<xsl:template match="mods:namePart">
   <element name="{concat($prefix, local-name(.))}">
      <value>
         <xsl:value-of select="."/>
      </value>
   </element>
</xsl:template>

<!-- role (name) -->
<xsl:template match="mods:role">
   <element name="{concat($prefix, local-name(.))}">
      <!-- kan knytte forbindelse til selve navnet ved å plassere dette i et
      eget meta-info-element -->
      <!-- <meta-info name="relation" value="{preceding-sib-
      ling::mods:namePart}"/> -->
      <value>
         <xsl:value-of select="mods:text"/>
      </value>
   </element>
</xsl:template>

<!-- publisher (originInfo)-->
<xsl:template match="mods:publisher">
   <element name="{concat($prefix, local-name(.))}">
      <value>
         <xsl:value-of select="."/>
      </value>
   </element>
</xsl:template>

<!-- dateCreated (originInfo) -->
<xsl:template match="mods:dateCreated">
   <element name="{concat($prefix, local-name(.))}">
      <value>
         <xsl:value-of select="."/>
      </value>
   </element>

   <!--====== Datokonvertering/-normalisering ======-->
   <!-- Kontekstnode . = mods:dateCreated -->
   <!-- Variabel som inneholder antall tegn i mods:dateCreated -->
```

```
<xsl:variable name="v_datelength" select="string-length(.)"/>

<!-- Variabel som skal inneholde en 8-tegns heltallsdato -->
<xsl:variable name="v_dateint">
   <xsl:choose>
      <!-- 4 tegn -->
      <xsl:when test="$v_datelength = 4">
      <xsl:choose>
         <!-- 1980 -->
         <xsl:when test="substring(.,1,1) >=0">
            <xsl:value-of select="concat(.,'0000')"/>
         </xsl:when>
         <!-- Ca.  -->
         <xsl:when test="contains(.,'Ca.')">00000000</xsl:when>
         <!-- Ukjent datotype -->
         <xsl:otherwise>00000000</xsl:otherwise>
      </xsl:choose>
      </xsl:when>

      <!-- 6 tegn -->
      <xsl:when test="$v_datelength = 6">
      <xsl:choose>
         <!-- 1980-? -->
         <xsl:when test="contains(substring(.,6,1),'?')">
            <xsl:value-of select="concat(substring(.,1,4),'0000')"/>
         </xsl:when>
         <!-- 194006, usikker på om dette er juni 1940-->
         <xsl:when test="substring(.,6,1) >= 0">
            <!-- p.g.a. usikkerhet fjernes mer nøyaktig info enn år -->
            <xsl:value-of select="concat(substring(.,1,4),'0000')"/>
         </xsl:when>
         <!-- Ukjent datotype -->
         <xsl:otherwise>00000000</xsl:otherwise>
      </xsl:choose>
      </xsl:when>

      <!-- 7 tegn -->
      <xsl:when test="$v_datelength = 7">
      <xsl:choose>
         <!-- 1098911 -->
         <xsl:when test="substring(.,1,1) >= 0">
            <!-- p.g.a. usikkerhet fjernes mer nøyaktig info enn år -->
            <xsl:value-of select="concat(substring(.,1,4),'0000')"/>
         </xsl:when>
         <!-- Ukjent datotype -->
         <xsl:otherwise>00000000</xsl:otherwise>
      </xsl:choose>
      </xsl:when>

      <!-- 8 tegn -->
      <xsl:when test="$v_datelength = 8">
      <xsl:choose>
         <!-- Ca. 1990 -->
         <xsl:when test="contains(.,'Ca.')">
            <xsl:value-of select="concat(substring-after(.,'Ca.
            '),'0000')"/>
         </xsl:when>
         <!-- Ukjent datotype -->
         <xsl:otherwise>00000000</xsl:otherwise>
```

```
        </xsl:choose>
        </xsl:when>

        <!-- 10 tegn -->
        <xsl:when test="$v_datelength = 10">
        <xsl:choose>
           <!-- 1980-01-01 -->
           <xsl:when test="contains(substring(.,5,1),'-')">
              <xsl:value-of select="concat(substring(.,1,4),sub-
              string(.,6,2),substring(.,9,2))"/>
           </xsl:when>
           <!-- Ca. 1980-? -->
           <xsl:when test="contains(substring(.,10,1),'?')">
              <xsl:value-of select="concat(substring(.,5,4),'0000')"/>
           </xsl:when>
           <!-- Ukjent datotype -->
           <xsl:otherwise>00000000</xsl:otherwise>
        </xsl:choose>
        </xsl:when>

        <!-- Andre datolengder settes til 00000000 -->
        <xsl:otherwise>00000000</xsl:otherwise>

     </xsl:choose>
  </xsl:variable>

  <!-- Nytt element som inneholder en enhetlig dato -->
  <element name="dateinteger">
     <value>
        <xsl:value-of select="$v_dateint"/>
     </value>
  </element>
  <!--===== SLUTT: Datokonvertering =====-->
</xsl:template>

<!-- internetMediaType (physicalDescription) -->
<xsl:template match="mods:internetMediaType">
  <element name="{concat($prefix, local-name(.))}">
     <value>
        <xsl:value-of select="."/>
     </value>
  </element>
</xsl:template>

<!-- extent (physicalDescription) -->
<xsl:template match="mods:extent">
  <element name="{concat($prefix, local-name(.))}">
     <value>
        <xsl:value-of select="."/>
     </value>
  </element>
</xsl:template>

<!-- topic (subject) -->
<xsl:template match="mods:topic">
  <!-- elementet skal ha navnet subject (som er navnet til parent ../.) -->
  <element name="{concat($prefix, local-name(../.))}">
     <value>
        <xsl:value-of select="."/>
```

```
      </value>
    </element>
  </xsl:template>

  <!-- geographic (subject) -->
  <xsl:template match="mods:geographic">
    <!-- elementet skal ha navnet subject (som er navnet til parent ../.) -->
    <element name="{concat($prefix, local-name(../.))}">
      <value>
        <xsl:value-of select="."/>
      </value>
    </element>
  </xsl:template>

  <!-- temporal (subject) -->
  <xsl:template match="mods:temporal">
    <!-- elementet skal ha navnet subject (som er navnet til parent ../.) -->
    <element name="{concat($prefix, local-name(../.))}">
      <value>
        <xsl:value-of select="."/>
      </value>
    </element>
  </xsl:template>

  <!-- namePart (subject/name). Bruker attributtet mode for å angi at
  mods:namePart skal behandles på en bestemt måte når det er subject -->
  <xsl:template match="mods:namePart" mode="subject">
    <!-- name=mods.subject (som er to trinn høyere i hierarkiet ../../.) -->
    <element name="{concat($prefix, local-name(../../.))}">
      <value>
        <xsl:value-of select="."/>
      </value>
    </element>
  </xsl:template>

  <!-- role (subject/name). Bruker attributtet mode for å angi at mods:namePart
  skal behandles på en bestemt måte når det er subject. -->
  <xsl:template match="mods:role" mode="subject">
    <!-- name=mods.subject (som er to trinn høyere i hierarkiet ../../.) -->
    <element name="{concat($prefix, local-name(../../.))}">
      <value>
        <xsl:value-of select="mods:text"/>
      </value>
    </element>
  </xsl:template>

  <!-- identifier (relatedItem). Bruker attributtet mode="relatedItem" for å
  angi at mods:identifier skal behandles på en bestemt måte når det er relate-
  dItem -->
  <xsl:template match="mods:identifier" mode="relatedItem">
    <!-- elementet skal ha navnet mods.relatedItem, dvs. parent (../.) -->
    <element name="{concat($prefix, local-name(../.))}">
      <value>
        <xsl:value-of select="."/>
      </value>
    </element>
  </xsl:template>

</xsl:stylesheet>
```

APPENDIKS D

Fast Data Search

Indekserings- og søkesystemet Fast Data Search spiller en viktig rolle i prototypen i dette prosjektet. I *kap. 4.1.3, s. 48* fins det en introduksjon til systemet og dets virkemåte. I dette appendikset presenteres:

- Prinsippet bak indeksprofiler
- Beskrivelse av indeksprofilen i dette prosjektet
- Kode for indeksprofil
- FileTraverser

D.1 Prinsippet bak indeksprofiler

En Fast Data Search indeksprofil er en xml-basert konfigurasjonsfil som definerer måten indekserte dokumenter skal være søkbare på. Indeksprofilen definerer med andre ord layouten til den søkbare indeksen, og spesifiserer på hvilken måte de ulike feltene skal behandles under spørring og resultatprosessering. Følgende egenskaper beskrives i en indeksprofil:

- Hvilke dokumentelementer som skal være søkbare felt, det vil si en avbildning fra elementer i FastXML-dokumenter (mods.titleInfo) til søkbare felt i indeksen (modstitle)
- Hvilke dokumentelementer som skal returneres som en del av resultatsettet
- Hvordan verdier som brukes til sortering og rangering skal beregnes
- Hvilke enkeltfelt som skal inngå i sammensatte felt. Et sammensatt felt er en gruppering av enkle felt som kan refereres ved hjelp av ett unikt navn i spørringer
- Én eller flere resultatpresentasjoner (result views) som definerer hvilke felt i indeksen som skal returneres og vises i resultatlisten etter et treff for en spørring

Hovedelementene i en standard indeksprofil er strukturert på følgende måte:

```
<index-profile name=”ip”>
     <field-list>
     <composite-field>
     <result-specification>
</index-profile>
```

D.2 Beskrivelse av indeksprofilen i dette prosjektet

Prinsippet i indeksprofilen i denne prototypen er at hvert metadataelement i FastXML-dokumentene som skal indekseres av Fast Data Search tilordnes et tilsvarende felt i indeksprofilen. Feltnavnene i indeksen har tilsvarende navn som elementnavnene i FastXML-dokumentet, det vil si med prefikset mods. Siden Fast Data Search krever at alle felt i indeksprofilen har navn skrevet med små bokstaver, må enkelte felt ha en eksplisitt beskrivelse av tilsvarende elementnavn (som også inneholder store bokstaver). Dette spesifiseres ved hjelp av attributtet `element-name`.

Eksempel på felt:

```
<field name="modstitle" element-name="mods.title" index="yes" wild-
card="yes"/>
```

I dette eksempelet opprettes det et felt med navnet *modstitle* i indeksen, siden verdien for attributtet index er yes. Dataene som skal indekseres i dette feltet befinner seg i elementet *mods.title* i kildedokumentet i FastXML-format.

Kun enkelte felt i indeksen skal være søkbare (`index="yes"`), mens andre bare skal vises som en del av resultatet (`index="no"`). Ved behov er det er også mulig å gruppere flere enkle felt i et sammensatt felt, "composite-field". I denne indeksprofilen anvendes denne teknikken til å definere ett felt, *simplesearch*, som benyttes når det søkes i det enkle søkegrensesnittet.

I avansert søk benyttes hvert enkelt felt i indeksen direkte.

Denne indeksprofilen inneholder også et eksempel på et såkalt *result-view*, som kan brukes dersom en ikke ønsker at alle opplysningene skal returneres i søkeresultatene. Result-view anvendes imidlertid ikke i dette systemet.

D.3 Kode for indeksprofil

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE index-profile SYSTEM "index-profile-2.0.dtd">
<!--============================
Document: indeksprofil.xml
Created on: 4. desember 2003
Author: Eskil Høyen Solvang
Description: Definerer indeksstrukturen i FAST Data Search, dvs. hvilke element-
        er som er søkbare, og på hvilken måte.
============================-->
<index-profile name="mods_ehs">
   <!--========== Feltliste ==========-->
   <field-list>
      <!-- OAI-identifikator -->
      <field name="oaiid" index="no"/>
      <!-- OAI-opprinnelse -->
      <field name="oaisource" index="yes"/>
      <!-- Title -->
      <field name="modstitle" element-name="mods.title" index="yes" wild-
         card="yes"/>
```

```
    <field name="modssubtitle" element-name="mods.subTitle" index="yes"/>
    <!-- Name -->
    <field name="modsnamepart" element-name="mods.namePart" index="yes" wild-
       card="yes"/>
    <!-- Role: Ingen indeksering siden det ikke søkes i rollefeltet -->
    <field name="modsrole" element-name="mods.role" index="no"/>
    <!-- Type of resource -->
    <field name="modstypeofresource" element-name="mods.typeOfResource" in-
       dex="yes"/>
    <!-- Genre -->
    <field name="modsgenre" element-name="mods.genre" index="yes"/>
    <!-- Publisher -->
    <field name="modspublisher" element-name="mods.publisher" index="yes"/>
    <!-- Language-->
    <field name="modslanguage" element-name="mods.language" index="yes"/>
    <!-- InternetMediaType -->
    <field name="modsinternetmediatype" element-name="mods.internetMediaType"
       index="yes"/>
    <!-- Extent: Ingen indeksering siden det ikke søkes i extent-feltet -->
    <field name="modsextent" element-name="mods.extent" index="no"/>
    <!-- Abstract: Result="dynamic" medfører at det genereres en s.k. dynamic
       teaser for feltet -->
    <field name="modsabstract" element-name="mods.abstract" max-result-
       size="1024" result="dynamic" index="yes"/>
    <!-- Subject: Alle forskjellige subjects i ett felt. Kunne vært strukturert
       annerledes (se rapport) for bedre granularitet -->
    <field name="modssubject" element-name="mods.subject" index="yes" wild-
       card="yes"/>
    <!-- RelatedItem -->
    <field name="modsrelateditem" element-name="mods.relatedItem" index="no"/>
    <!-- Identifier-->
    <field name="modsidentifier" element-name="mods.identifier" index="yes"/>
    <!-- Date: Utvidet med feltet dateinteger som inneholder datoen konvertert
       til åttesiffers heltall. Dette gjør det mulig å søke etter datoer  -->
    <field name="modsdatecreated" element-name="mods.dateCreated" index="no"/>
    <!-- dateinteger skal kun brukes "internt" ved søk etter datoer -->
    <field name="dateinteger" type="integer" index="yes"/>
    <!-- AccessCondition -->
    <field name="modsaccesscondition" element-name="mods.accessCondition" in-
       dex="no"/>
</field-list>
<!---------- Sammensatte felt ---------->
<!-- Enkelt søk -->
<composite-field name="simplesearch" rank="yes" default="yes" type="text"
      wildcard="yes" substring="0" lemmas="no">
    <field-ref name="modsabstract" level="1" type="normal"/>
    <field-ref name="modspublisher" level="2" type="normal"/>
    <field-ref name="modsnamepart" level="3" type="normal"/>
    <field-ref name="modssubject" level="4" type="normal"/>
    <field-ref name="modssubtitle" level="4" type="normal"/>
    <field-ref name="modstitle" level="5" type="normal"/>
    <!-- Vektliste for feltene i simplesearch -->
    <weight-list name="defaultrank" and-boost="0" or-boost="5000" rank-
       boost="5000" phrase-boost="5000">
       <field-weight field-ref="modsabstract" value="100000"/>
       <field-weight field-ref="modspublisher" value="125000"/>
       <field-weight field-ref="modsnamepart" value="125000"/>
       <field-weight field-ref="modssubtitle" value="125000"/>
       <field-weight field-ref="modstitle" value="150000"/>
```

```
        <field-weight field-ref="modssubject" value="200000"/>
      </weight-list>
   </composite-field>
   <!-- Felles tittelfelt: Brukes i avansert søk ved søking i tittel -->
   <composite-field name="titlecomp">
      <field-ref name="modstitle"/>
      <field-ref name="modssubtitle"/>
   </composite-field>
   <!--========= Resultater =========-->
   <!--
   * "The default result view (search, min komm.) contains all fields and com-
        posite-fields that have been specified with the result attribute not
        equal to no. (config guide s. 121).
   * Siden resultatlisten bare skal gjenspeile strukturen i indeksen (og ingent-
        ing annet), er dette elementet foreløpig tomt (desember 2003).
   -->
   <result-specification>
      <!-- Kortform av resultatlisten der bare de viktigste parametrene tas med
         (bør vurdere om denne egner seg bedre enn default (search)) - må spesi-
         fiseres eksplisitt ved søking at denne skal brukes (se Integration Guide
         s.192) -->
      <result-view name="short">
         <field-ref name="oaiid"/>
         <field-ref name="modstitle"/>
         <field-ref name="modsnamepart"/>
         <field-ref name="modsrole"/>
         <field-ref name="modspublisher"/>
         <field-ref name="modssubject"/>
         <field-ref name="modsabstract"/>
         <field-ref name="modsgenre"/>
         <field-ref name="modslanguage"/>
         <field-ref name="modsidentifier"/>
         <field-ref name="dateinteger"/>
         <field-ref name="modsaccesscondition"/>
      </result-view>
   </result-specification>
</index-profile>
```

D.4 FileTraverser

FileTraverser er én av mange små applikasjoner som inngår i totalsystemet Fast Data Search. Hensikten med FileTraverser er å gå gjennom en rekke filer som enten befinner seg i en bestemt katalog i filsystemet eller er tilgjengelig gjennom en bestemt URL.

FileTraverser startes manuelt fra kommandolinjen:

```
./filetraverser -r /disk1/fast/data/ -c nbntnu1 -x :document:id
```

Denne kommandoen fører til at fast går gjennom (`-r`) alle filene i katalogen `/disk1/fast/data` og sender dem (`-c`) til indeksering i samlingen `nbntnu1`. Alle filene er FastXML-filer (`-x`) og hvert element `document` med identifikator `id` representerer én metadatabeskrivelse og blir derfor indeksert som et eget dokument.

Mens FileTraverser går gjennom filene som skal indekseres, gis det tilbakemelding på skjerm om indekseringsprosessen:

```
[fast@utvikling1 bin]$ ./filetraverser -r /disk1/fast/data/ -c nbntnu1 -x :doc-
        ument:id

Filetraverser@utvikling1.nb.no: nbntnu1: Starting filetraverser

Filetraverser@utvikling1.nb.no: nbntnu1: Sending batch of 200 documents
        (original data size = 385.64 kB)

Filetraverser@utvikling1.nb.no: nbntnu1: Sending batch of 200 documents
        (original data size = 375.69 kB)

Filetraverser@utvikling1.nb.no: nbntnu1: Sending batch of 200 documents
        (original data size = 366.89 kB)

Filetraverser@utvikling1.nb.no: nbntnu1: Sending batch of 200 documents
        (original data size = 364.55 kB)

Filetraverser@utvikling1.nb.no: nbntnu1: Sending batch of 200 documents
        (original data size = 368.28 kB)

Filetraverser@utvikling1.nb.no: nbntnu1: Sending batch of 122 documents
        (original data size = 118.72 kB)

Filetraverser@utvikling1.nb.no: nbntnu1: Sending batch of 200 documents
        (original data size = 550.45 kB)

Filetraverser@utvikling1.nb.no: nbntnu1: Sending batch of 81 documents (
        original data size = 158.09 kB)
```

Etter at FileTraverser har gjort sin jobb, er alle metadatabeskrivelsene som opprinnelig var xml-filer i henhold til FastXML-formatet, indeksert som søkbare dokumenter i Fast Data Search.

APPENDIKS E

Søketjeneste

Søkegrensesnittet, som utgjør systemets kontaktpunkt med brukerne, spiller en svært viktig rolle i ethvert søkesystem. Siden hensikten med prosjektet er å gjøre informasjon lettere tilgjengelig, er det naturlig å anvende et webbasert søkegrensesnitt som kan brukes over hele verden. I dette appendikset presenteres:

- Beskrivelse av søkegrensesnitt og mellomvarelag
- Oversikt over filer
- Kode for søketjenesten

E.1 Beskrivelse av søkegrensesnitt og mellomvarelag

Som nevnt i *kap. 5.6, s. 73* er søkegrensesnittet utviklet ved hjelp av en kombinasjon av html, css og skriptspråket php. Mellomvarelaget er utviklet i php. En medvirkende årsak til at php er valgt som skriptspråk, er at Fast Data Search sine medfølgende søkefunksjoner for webgrensesnitt er utviklet i php. Disse funksjonene kan dermed gjenbrukes på en enkel måte, istedenfor å måtte utvikle denne funksjonaliteten på nytt i et annet språk.

`Index.php` utgjør en av de viktigste filene i denne delen av systemet siden den binder sammen søkegrensesnittet og Fast Data Search. Når `index.php` mottar et søkeuttrykk fra en bruker av det enkle søkegrensesnittet, sendes denne spørringen umiddelbart videre til Fast Data Search. I indeksprofilen er det definert et sammensatt felt – *simplesearch* – som inneholder feltene tittel, forfatter, utgiver, emneord, abstract. Alle søkeuttrykk i enkelt søk sendes direkte til dette sammensatte feltet.

Dersom `index.php` mottar et søkeuttrykk fra en bruker av det avanserte søkegrensesnittet, må søketermene fra de forskjellige feltene omformuleres til en spørring i Fast Data Search sitt spørrespråk. Hvis en av søketermene for eksempel gjelder en tittel, må søketermen sendes som en spørring til indeksfeltet *modstitle*, som alle titler er indeksert under. Slike indeksfelt er tidligere blitt definert i indeksprofilen.

Koden bak søkegrensesnittet må med andre ord være oppdatert i henhold til strukturen i indeksprofilen, slik at brukerens søkeuttrykk formidles til de korrekte feltene i søkeindeksen.

E.2 Oversikt over filer

Søkegrensesnittet og mellomvarelaget inneholder følgende filer:

```
om.php
hjelp.php
avansert.php
index.php

inc/web_common.inc
inc/common.inc
inc/search.inc
inc/search.js

css/websok.css
```

Med unntak av de filene som er utviklet av Fast Search & Transfer for Fast Data Search, presenteres koden for alle disse filene i dette appendikset. Sammenhengen mellom filene i søkegrensesnitt og mellomvarelag er uttrykk i uml-modellen i *figur 5-4, s. 74*.

E.3 Kode for søketjenesten

E.3.1 Enkelt søkegrensesnitt (index.php)

```
<?php
/*
=========================
index.php
-------------------------
Dato: Januar 2004
Oppdatert: 1.3.2004
Funksjoner:
 - Enkelt søkegrensesnitt
 - Behandle enkelt søk
 - Behandle avansert søk
=========================
*/

//Inkluderer filer med funksjoner
include 'include/common.inc';// DisplaySearchBar()
include 'include/search.inc';// search(), printsearchresults()
include 'include/web_common.inc';// commonHeader(), commonFooter()

//Variabler
$param = $_GET;
$collection = "nbntnu2";
$fastserver = "utvikling1.nb.no";

//Skiller mellom enkelt og avansert søk
if ($_GET['adv'] == 1) {
   $section = 'sectiontwo';
   $title = 'Søkemotor | Avansert søk';
```

```
} else {
   $section = 'sectionone';
   $title = 'Søkemotor | Enkelt søk';
}

//Header
commonHeader($title, $section);

/*
========================
Enkelt søkeskjema (form)
(vises både i søk og søkeresultater)
========================
*/

print '

<!-- content -->
<div id="sok">
   <form action="'; $PHP_SELF; print '" method="get" name="simpleform">';
   DisplaySearchBar();

   print '
   </form>
</div>
';

//angir output-format
$output_format = "generic"; //tekst, dvs. ikke xml, html eller wml

//søkeparametere
if(!isset($hits))
   $hits = 10;
if(!isset($charset))
   $charset = "utf-8";
if(!isset($clustering))
   $clustering = 1;

$dynteaser = "";

/*
================
Utfør enkelt søk
================
*/
if ($query != "") {// Søkefeltet $query inneholder verdi
   print '

   <!-- results -->
   <div id="resultater">
   ';
   PrintSearchResultTableStart();

   $query = urlencode($query);
   // initialize vars from parameters transferred
   if(array_key_exists('exact', $param) and ($param['exact'] == "on"))
      $type = "phrase";
   else if(!isset($type))
      $type = "all";
```

```
   if(array_key_exists('offset', $param))
      $offset = $param['offset'];
   else
      $offset = "";

   // Utfør søket i Fast Data Search
   $totalnumhits = Search($output_format, $query, $type, $offset, $hits,
   $charset, $lang, $collection, $dynteaser, $fastserver, "15100", $clustering,
   $filter );
   DisplayResultNavbar() ;

   PrintSearchResultTableStop();

   print '</div>';
}

/*
====================
Utfør avansert søk
(hvis inputparameteret adv er satt til 1, dvs. input kommer fra avansert
søkeskjema)
====================
*/
else if ($_GET['adv'] == 1) {
   //Bygger opp en streng med alle GET-parametrene
   $urlparam = "";
   foreach ($_GET as $key => $value) {
      if ($key == 'submit') {
         echo '';
      } else {
         if($urlparam != "") {
            $urlparam = $urlparam.'&';
         }
         $urlparam = $urlparam.$key.'='.$value;
      }
   }

   //Lenke tilbake til avansert søkeside
   echo '<a href="avansert.php?'.$urlparam.'">Endre søkeparametre</a>';

   //Nullstiller query- og filtervariablene
   $query = "";
   $filter = "";

   //Behandling av søkeparametre + oppbygging av querystreng
   $kilde = $_GET['arkiv'];
   $tittel = $_GET['tittel'];
   $navn = $_GET['navn'];
   $utgiver = $_GET['utgiver'];
   $subject = $_GET['subject'];
   $abstract = $_GET['abstract'];
   $lang = $_GET['lang'];

   $txt = $_GET['txt'];
   $snd = $_GET['snd'];
   $img = $_GET['img'];
   $vid = $_GET['vid'];

   $dato = $_GET['dato'];
```

```
   $fra = $_GET['fra'];
   $til = $_GET['til'];

   $hits = $_GET['hits'];

   /*
   Filter: Opprinnelig samling/datakilde
   Filtrerer søket avhengig av opprinnelig samling
   */
   if ($kilde != 'alle') { //lag filter for aktuelt arkiv
      $kildefilter = "+oaisource:".$kilde;
   }

   /*
   Query: Søkbare felt
   Søker i feltene tittel, navn, utgiver, emneord og abstract i søkeindeksen
   */
   //titlecomp er et sammensatt felt (modstitle + modssubtitle) i indeksprofilen
   if ($tittel != '') {$tittelq = "+titlecomp:".$tittel;}
   if ($navn != '') {$navnq = "+modsnamepart:".$navn;}
   if ($utgiver != '') {$utgiverq = "+modspublisher:".$utgiver;}
   if ($subject != '') {$subjectq = "+modssubject:".$subject;}
   if ($abstract != '') {$abstractq = "+modsabstract:".$abstract;}

   /*
   Filter: Ressurstype
   Filtrerer søket avhengig av verdier i modsresourceType i søkeindeksen
   */
   $resourcefilter = "+(";
   if ($txt == "on") {$resourcefilter .= "modstypeofresource:text ";}
   if ($snd == "on") {$resourcefilter. = "modstypeofresource:\"sound recording\"
";}
   if ($img == "on") {$resourcefilter .= "modstypeofresource:\"still image\" ";}
   if ($vid == "on") {$resourcefilter .= "modstypeofresource:\"moving image\" ";}
   $resourcefilter .= ")";

   /*
   Query: Dato
   Søker i feltet dateinteger (dato i heltallsform ååååmmdd) i søkeindeksen
   * Alle: Definerer start- og sluttdato her
   * Eksakt: Søker etter ressurser med ett bestemt år, men bruker intervallet
      fraÅ0000 - fraÅ1231 for å få treff på alle datoer i betraktet år
   * Mellom: Søker etter ressurser mellom to årstall, og benytter intervallet
      fraÅ0000 - tilÅ1231 for å få treff på alle datoer mellom starten på 'fra'
      og slutten på 'til'
   */
   if ($dato == "alle") { //søk i alle datoer
      //setter variabler for tidligste og seneste år
      $startdato = '18000000';
      $sluttdato = '20040000';
      //bygger query
      $datoquery = "+dateinteger:[" . $startdato . ";" . $sluttdato. "]";
   } else if ($dato == "eksakt") { //søk etter eksakt dato
      $datoquery = "+dateinteger:[" . $fra . "0000;" . $fra . "1231]";
   } else if ($dato == "mellom") { //søk mellom angitte datoer
      $datoquery = "+dateinteger:[" . $fra . "0000;" . $til . "1231]";
   } else { //ingen datosøk
      $datoquery = "";
   }
```

```
   //Danner fullstendig query og filter ut fra delene som er beskrevet ovenfor
   $query = $tittelq .' '. $navnq .' '. $utgiverq .' '. $subjectq .' '. $abstractq
   .' '. $datoquery;

   $filter = $kildefilter .' '. $resourcefilter;

   print "
   <!-- results -->
   <div id=\"resultater\">
   ";

   PrintSearchResultTableStart();
      $query = urlencode($query);
      // initialize vars from parameters transferred
      if(array_key_exists('exact', $param) and ($param['exact'] == "on"))
         $type = "phrase";
      else if(!isset($type))
         $type = "all";

      if(array_key_exists('offset', $param))
         $offset = $param['offset'];
      else
         $offset = "";

      //Utfører søk i Fast Data Search
      $totalnumhits = Search($output_format, $query, $type, $offset, $hits,
      $charset, $lang, $collection, $dynteaser, $fastserver, "15100", $cluster-
      ing, $filter );
      DisplayResultNavbar() ;
   PrintSearchResultTableStop();

   print "</div>";

}

//Footer
commonFooter();
?>
```

E.3.2 Avansert søkegrensesnitt (avansert.php)

```
<?php
/*
=========================
avansert.php
-------------------------
Dato: Januar 2004
Oppdatert: 1.3.2004
Funksjoner:
 - Vise avansert søkefelt
=========================
*/

//Inkluderer filer med funksjoner
include 'include/common.inc';// DisplayAdvancedSearchBar()
include 'include/web_common.inc';// commonHeader(), commonFooter()

//Variabler
$param = $_GET;
```

```
//Header
commonHeader('Søkemotor | Avansert søk', 'sectiontwo');

   print "

   <!-- content -->
   <div id=\"sok\">
      <form action=\"index.php\" method=\"get\" name=\"advform\">";
      DisplayAdvancedSearchBar();

      print "
      </form>
   </div>
   ";

//Footer
commonFooter();
?>
```

E.3.3 Søkefunksjon (search.inc)

Søkefunksjonen i search.inc er laget av Fast, og tas derfor ikke med i dette appendikset. Den overordnede strukturen i funksjonen er:

```
function Search() {
   Undersøk søkeparametere i spørring (fra index.php)
   Utfør søk ved å kalle funksjonen performSearch()
   Motta søkeresultater
   Skriv ut søkeresultater
}
```

E.3.4 Include-filer

web_common.inc

```
<?php
/*
==============
web_common.inc
--------------
Funksjon:
- Felles funksjoner for alle sidene
==============
*/
function commonHeader($title, $id) {
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
   <title><?php echo $title; ?></title>
   <link rel="stylesheet" type="text/css" href="css/websok.css" />
   <script type="text/javascript" src="include/sok.js">
      <!-- Din nettleser støtter ikke javascript -->
   </script>
</head>
<body id="<?php echo $id; ?>">
<!-- hovedramme -->
<div id="frame">
```

```
   <!-- header -->
   <div id="menu">
      <ul>
         <li id="one"><a href="index.php">Enkelt søk</a></li>
         <li id="two"><a href="avansert.php">Avansert søk</a></li>
         <li id="three"><a href="hjelp.php">Hjelp</a></li>
         <li id="four"><a href="om.php">Om</a></li>
      </ul>
   </div>
<?php
}
?>

<?php
//footer
function commonFooter() {
?>
   <!-- footer -->
   <div id="footer">
   <p>
      20&copy;04 <br />
      <!-- W3C XHTML, W3C CSS <br /> -->
      <a href="http://www.openarchives.org" target="_link">
         <img src="img/oailogo.gif" border="0"/>
      </a>
      <a href="http://www.oclc.org/research/software/default.htm" tar-
      get="_link">
         <img src="img/oaiharvesterlogo.gif" border="0" />
      </a>
      <br />
      <img src="img/pb_fast.gif">
   </p>
   </div>
</div>
</body>
</html>
<?php
}
?>

<?php
/*
PHP-koden nedenfor er hentet fra FAST Data Search

Resultatene etter et søk skrives ut i en tabell, og disse to funksjonene sørger
for å skrive ut starten og slutten på resultattabellen.
*/
function PrintSearchResultTableStart()
{
    print "<table align='center' border='0 cellspacing='0' cellpadding='1'
     width='90%'>\n<tr>\n<td>\n";
    print "<table align='center' border='0' cellspacing='0' cellpadding='1'
     width='100%'>\n<tr>\n<td>\n";
}
```

```
function PrintSearchResultTableStop()
{
    print "</td></tr></table>\n";
    print "</td></tr></table>\n";
}
?>
```

common.inc

```
<?php
//require_once ("admin/include/guiConfig.php");

// The simple query form
$form_simple   = array (
        "all of the words" => "all",
        "any of the words" => "any",
        "the exact phrase" => "phrase"
        ) ;

// The default number of hits
$default_num_hits = 10 ;

// Maximum number of hits
$max_num_hits = 100 ;

// Displays the search bar used both in simple search and advanced search.
// The beginning and end of the table must be written outside this function
// Oppdatert 13.1.2004: Eskil
function DisplaySearchBar()
{
    global $param;
    global $form_simple;    // array - simple search bar
    global $self;

    // Resets the internal PHP variables in the arrays so we can call this
    // function more than once. Kind of weird PHP behaviour...
    reset($form_simple);

    $encoding = GetEncoding() ;

    //Search form + Submit button
   print "
      <p>
         <input type=\"text\" size=\"50\" name=\"query\" value=\""; echo
         $_GET['query']; print "\"/>
         <input type=\"submit\" value=\"Søk »\" name=\"submit\" />
      </p>";

    // Menu line
    if (array_key_exists('lemmatize', $param) and $param['lemmatize']   == "on")
        $qt_lChecked  = "checked";
    else
        $qt_lChecked  = "";

    if (array_key_exists('exact', $param) and $param['exact'] == "on")
        $exactChecked = "checked";
    else
        $exactChecked = "";

    if (array_key_exists('dynteaser', $param) and $param['dynteaser'] == "off")
```

```
        $dynTeaserChecked = "checked";
    else
        $dynTeaserChecked = "";
}

/*
* Funksjon: DisplayAdvancedSearchBar()
* Hensikt: Skriver ut feltene i avansert søk
* Dato: 15.1.2004
* Laget av: Eskil
*/
function DisplayAdvancedSearchBar() {
    //variabler
    global $param;
    global $self;

   /*
   27.2.2004: Opprinnelig var tanken at man skulle kunne filtrere etter forskjellige språk, men siden det kun er enkelte ressurser hvor språk er registrert, kuttes dette søkefeltet ut.

   //ISO-639
   $lang_array = array (

      "any"=> "Alle språk",
      "no"=> "Norsk",
      "en"=> "Engelsk"
   );
   */

   // Søkbare arkiver
   // 17.1.2004, Eskil
   $arkiv_array = array (
      "alle"=> "Alle",
      "galnor"=> "Galleri NOR",
      "dra"=> "Digitalt Radioarkiv",
      "mavis"=> "Mavis"
   );

    $encoding = GetEncoding() ;

    // Menu line
    if (array_key_exists('lemmatize', $param) and $param['lemmatize']   == "on")
        $qt_lChecked  = "checked";
    else
        $qt_lChecked  = "";

    if (array_key_exists('exact', $param) and $param['exact'] == "on")
        $exactChecked = "checked";
    else
        $exactChecked = "";

    if (array_key_exists('dynteaser', $param) and $param['dynteaser'] == "off")
        $dynTeaserChecked = "checked";
    else
        $dynTeaserChecked = "";

?>
```

```
<!--
================================
HTML + PHP-kode for avansert søk

Laget: 17.1.2004, Eskil
Oppdatert: 27.2.2004
Search form + Submit button
================================
-->
<fieldset>
     <!-- Skjult input-felt som markerer at søket er av type avansert søk -->
     <input type="hidden" name="adv" value="1" />

     <legend>Arkiv</legend>
     Arkiv det skal søkes i
        <select name="arkiv">
        <?php
        //Skriver ut en liste over søkeparametre, og sjekker om noen av disse
        er valgt fra før. I så fall, merk valgte.
        //16.1.2004, EHS
        while (list($key, $value) = each($arkiv_array)) {
        print "<option ";
        if (array_key_exists('arkiv', $_GET) && $_GET['arkiv'] == $key)
           print "selected ";
        print "value=\"$key\">$value</option>\n";
        } // while
        ?>
        </select>
   </fieldset>

   <fieldset>
   <legend>Attributter</legend>
   <table border="0" cellspacing="5" cellpadding="0">
   <tr>
      <td>Tittel</td>
      <td><input type="text" size="50" name="tittel" value="<?php echo
         $_GET['tittel']; ?>"/></td>
   </tr>
   <tr>
      <td>Navn</td>
      <td><input type="text" size="50" name="navn" value="<?php echo
         $_GET['navn']; ?>" /></td>
   </tr>
   <tr>
      <td>Utgiver</td>
      <td><input type="text" size="50" name="utgiver" value="<?php echo
         $_GET['utgiver']; ?>"/></td>
   </tr>
   <tr>
      <td>Emneord</td>
      <td><input type="text" size="50" name="subject" value="<? echo
         $_GET['subject']; ?>"/></td>
   </tr>
   <tr>
      <td>Abstract</td>
      <td><input type="text" size="50" name="abstract" value="<? echo
         $_GET['abstract']; ?>"/></td>
   </tr>
   </table>
```

```
</fieldset>

<fieldset>
   <legend>Type</legend>
   <!--
   Velg ressurstype
   PHP-koden viser eventuelt valgte parametre hvis søkeren ønsker å endre
   søkeparametrene etter å ha søkt
   16.1.2004: EHS
   -->
   <input type="checkbox" name="txt" <?php if($_GET['txt']=='on') echo
   'checked'; ?>>Tekst</input>
   <input type="checkbox" name="snd" <?php if($_GET['snd']=='on') echo
   'checked'; ?>>Lyd</input>
   <input type="checkbox" name="img" <?php if($_GET['img']=='on') echo
   'checked'; ?>>Bilde</input>
   <input type="checkbox" name="vid" <?php if($_GET['vid']=='on') echo
   'checked'; ?>>Video</input>
</fieldset>

<fieldset>
   <legend>Dato</legend>
   <table border="0">

   <!--
   Merker type år
   <?php if ($_GET['dato'] == 'eksakt') echo 'checked'; ?>
   <?php if ($_GET['dato'] == 'mellom') echo 'checked'; ?>
   -->

   <tr>
      <td><input type="radio" name="dato" value="alle" checked on-
      click="hidediv('aar1'); hidediv('aar2');" />Alle år</td>
   </tr>
   <tr>
      <td><input type="radio" name="dato" value="eksakt" onclick="show-
      div('aar1'); hidediv('aar2');" <?php if ($_GET['dato'] == 'eksakt')
      echo 'checked'; ?> />Eksakt år</td>
      <td>    </td>
      <td>

      <!--
      php-kode som legger til et stilattributt, s.a. div-seksjonen som in-
      neholder årstall vises
      <?php //if (($_GET['dato'] == 'eksakt') || (($_GET['dato'] == 'mel-
      lom')) echo 'style="visibility: visible;"'; ?> i div: aar
      <?php //if ($_GET['dato'] == 'mellom') echo 'style="visibility: vis-
      ible;"'; ?> i div: aar2
      -->
      <div id="aar1" <?php if (($_GET['dato'] == 'eksakt') || ($_GET['dato']
      == 'mellom')) echo 'style="visibility: visible;"'; ?>>
      <select name="fra">
      <?php
      //Skriver ut en liste over søkeparametere, og sjekker om noen av disse
         er valgt fra før. I så fall, merk valgte.
      //17.1.2004, EHS.
      for ($i=2004; $i>=1800; $i--) {
         print "<option ";
```

```
            //Sjekker om verdien 'fra' er en parameter i $_GET, og om verdien
            av 'fra' tilsvarer betraktet år ($i). Hvis så, merk året som se-
            lected.
            if (array_key_exists('fra', $_GET) && $_GET['fra'] == $i)
               print "selected ";
            print "value=\"$i\">$i</option>";
         }
         ?>
         </select>
         </div>
         </td>
         <td>
         <div id="aar2" <?php if ($_GET['dato'] == 'mellom') echo 'style="vis-
         ibility: visible;"'; ?>>
         til år
         <select name="til">
         <?php
         for ($i=2004; $i>=1800; $i--) {
            print "<option ";
            //Sjekker om verdien 'til' er en parameter i $_GET, og om verdien
            av 'til' tilsvarer betraktet år ($i). Hvis så, merk året som se-
            lected.
            if (array_key_exists('til', $_GET) && $_GET['til'] == $i)
               print "selected ";
            print "value=\"$i\">$i</option>";
         }
         ?>
         </select>
         </div>
         </td>
      </tr>
      <tr>
         <td><input type="radio" name="dato" value="mellom" onclick="show-
            div('aar1'); showdiv('aar2');" <?php if ($_GET['dato'] == 'mellom')
            echo 'checked'; ?> />Fra år</td>
      </tr>
      </table>
   </fieldset>

   <fieldset>
      <legend>Resultatpresentasjon</legend>
         Vis
         <select name="hits">
            <option>5</option>
            <option>10</option>
         </select>
         resultater per side.
   </fieldset>
   <br />
   <input type="reset" value="Nullstill" />  
   <input type="submit" name="submit" value="  Søk »  " />

  <!-- SLUTT: HTML + PHP for avansert søkeform -->
<?php
}
```

```
// Display result navigation bar below the results
function DisplayResultNavbar() {

  global $param ;
  global $self ;
  global $totalnumhits ;

  $num_results = $totalnumhits;
  $offset = GetOffset() ;
  $hits = GetHits() ;

  echo "<p>\n" ;

  // We can only show 4000 search results... That's the limit from fsearch
  if ($num_results > 4000)
    $num_results = 4000 ;

  # The max number of page links to show.
  # Note: This number must be an odd number
  $max_pages = 7 ;

  $delta_offset = floor($max_pages / 2) * $hits ;

  $start_offset = $offset - $delta_offset ;
  $stop_offset = $offset + $delta_offset ;

  if ($start_offset < 0) {
    $stop_offset -= $start_offset ;
    $start_offset = 0 ;
  }

  if ($stop_offset + $hits > $num_results) {
    $tmp_offset = $num_results - $num_results % $hits ;
    if ($tmp_offset == $num_results)
      $tmp_offset -= $hits ;
    $start_offset -= ($stop_offset - $tmp_offset) ;
    $stop_offset = $tmp_offset ;
  }

  if ($start_offset < 0)
    $start_offset = 0 ;

  $url_string = "<a href=\"$self?" ;

  if ($num_results > 0)
    echo "<b><span class=\"resultbar\">  Resultater:</
span>  " ;

  // print the left << navigation arrows if we're not at the start of the offset
   range
  if ($offset > 1)
    echo "$url_string" . BuildUrlString($offset - $hits) . "\"><span class=\"re-
sultbar\">&lt;&lt;</span></a>   \n" ;

  for ($page = $start_offset; $page <= $stop_offset; $page += $hits) {
    $span_start = $page + 1 ;
    $span_stop = min($num_results, ($page + $hits)) ;
    $span = ($span_start != $span_stop) ? "$span_start-$span_stop" :
          "$span_start" ;
```

```
    if ($offset == $page)
      echo "<span class=\"resultbar\">$span</span>" ;
    else
      echo "$url_string" . BuildUrlString($page) . "\"><span class=\"result-
           bar\">$span</span></a>\n" ;
    echo "   " ;
  }

  // print the right >> navigation arrows if we're not at the end of the offset
range
  if ($offset + $hits < $num_results)
    echo "$url_string" . BuildUrlString($offset + $hits) . "\"><span
        class=\"resultbar\">&gt;&gt;</span></a></b>\n" ;

  print "<p> </p>\n" ;
}

function DisplayTipBoxHeader($size, $title)
{
    $border_size = $size - 3;
    $inner_size = $size - 4;

    print "<td class=mainresults valign=top>";
    print "<table width=$size border=0 cellspacing=0 cellpadding=0>";
    print '<tr><td colspan=5 class=border>';
    print '<img src=/admin/graphics/dot_clear.gif width=1 height=1 alt="">';
    print '</td></tr><tr><td class=border rowspan=4>';
    print '<img src=/admin/graphics/dot_clear.gif width=1 height=30 alt="">';
    print '</td><td class=boxheader valign=top>';
    print '<img src=/admin/graphics/dot_clear.gif width=4 height=1 alt="">';
    print "</td><td width=$border_size class=boxheader valign=top col-
           span=2> ";
   print '<img src=/frontend/graphics/lightbulb.gif width=14 height=17 align=ab-
           smiddle alt="">';

    print " <b>$title</b></td><td class=border rowspan=4>";
    print '<img src=/admin/graphics/dot_clear.gif width=1 height=10 alt="">';
    print '</td></tr><tr><td colspan=3 class=boxsearchtip valign=top>';
    print '<img src=/admin/graphics/dot_clear.gif width=3 height=4 alt="">';
    print '</td></tr><tr><td width=1 class=boxsearchtip valign=top>';
    print '<img src=/admin/graphics/dot_clear.gif width=3 height=1 alt="">';
    print "</td><td width=$inner_size class=boxsearchtip valign=top >";
}

function DisplayTipBoxFooter()
{
    print '</td><td class=boxsearchtip valign=top>';
    print '<img src=/admin/graphics/dot_clear.gif width=1 height=1 alt="">';
    print '</td></tr><tr><td class=boxsearchtip colspan=3 valign=top>';
    print '<img src=/admin/graphics/dot_clear.gif width=3 height=5 alt="">';
    print '</td></tr><tr class=border><td colspan=5 class=border>';
    print '<img src=/admin/graphics/dot_clear.gif width=1 height=1 alt="">';
    print '</td></tr><tr><td colspan=5>';
    print '<img src=/admin/graphics/dot_clear.gif width=1 height=3 alt="">';
    print '</td></tr></table></td>';
}

// Finds the decoding scheme to use for the page
function GetEncoding()
```

```
{
    global $HTTP_POST_VARS ;
    global $HTTP_COOKIE_VARS ;
    global $param ;
    global $language_encoding_map;

    // Default encoding is ISO-8859-1
    //
    if (isset($HTTP_POST_VARS['set_encoding']) && $HTTP_POST_VARS['conf'] !=
     "Cancel") {
        $encoding = $HTTP_POST_VARS['set_encoding'] ;
    } elseif (isset($param['enco'])) {
        $encoding = $param['enco'] ;
    } else {
   # figure out characterset based on selected language
   $lang = GetLanguage();

   if(isset($lang))
   {
      $encoding = $language_encoding_map[$lang];
      if(!isset($encoding))
         $encoding = "utf-8";
   }
   else
      $encoding = "utf-8";
    }
    return $encoding ;
}

// Finds the default language
function GetLanguage()
{
  global $HTTP_POST_VARS ;
  global $HTTP_COOKIE_VARS ;
  global $param ;

  // Default langauge is any
  //
  if (isset($HTTP_POST_VARS['set_language']) && $HTTP_POST_VARS['conf'] != "Can-
      cel") {
    $language = $HTTP_POST_VARS['set_language'] ;
  } elseif (isset($param['lang'])) {
    $language = $param['lang'] ;
  } elseif (isset($HTTP_COOKIE_VARS['atw_language'])) {
    $language = $HTTP_COOKIE_VARS['atw_language'] ;
  } else {
    $language = "any" ;
  }
  return $language ;
}

// Returns whether people want highlighting of query terms in search results
turned on or off. This variable
// isn't configurable from the Advanced Search page, only from the Customize page
function GetMark()
{
  global $HTTP_POST_VARS ;
  global $HTTP_COOKIE_VARS ;
  global $param ;
```

```
  // Default is 1
  //
  // First check set_mark to see if user is using the customize page. Then check
   the cookie variable.
  // If nothing found so far, use default.

  if (isset($HTTP_POST_VARS['set_mark']) && $HTTP_POST_VARS['conf'] != 'Cancel')
    $mark = $HTTP_POST_VARS['set_mark'] ;
  elseif (isset($HTTP_COOKIE_VARS['atw_mark']))
    $mark = $HTTP_COOKIE_VARS['atw_mark'] ;
  else
    $mark = 1 ;

  // Verify that users aren't messing with our variables

  $mark = (int)$mark ;
  if (($mark != 0) && ($mark  != 1) && ($mark != 3))
    $mark = 1 ;

  return $mark ;
}

// Finds the number of search results to show per page. Do not look in the cookies
function GetHits()
{
  global $param ;
  global $default_num_hits ;
  global $max_num_hits ;

  if (isset($param['hits'])) {
    $hits = $param['hits'] ;
  } else {
    $hits = $default_num_hits ;
  }
  // if (($hits <= 0) || ($hits > $max_num_hits)) {
  if ($hits <= 0){
    $hits = $default_num_hits ;
  }
  if ($hits > $max_num_hits) {
       $hits = $max_num_hits ;
  }
  return (int)$hits ;
}

// Finds the offset
function GetOffset()
{
  global $param ;

  if (isset($param['offset'])) {
    $offset = $param['offset'] ;
  } else {
    $offset = 0 ;
  }
  if (($offset < 0) || ($offset > 4000)) {
    $offset = 0 ;
  }

  return (int)$offset ;
```

```
}


// Checks if browser is Microsoft Internet Explorer
function IsMSIE()
{
  global $HTTP_USER_AGENT ;

  if (strstr($HTTP_USER_AGENT, "MSIE"))
    return 1 ;
  else
    return 0 ;
}


// Checks if browser is Netscape version prior to 6.0 under Windows. The text
form for those browsers
// are way too long, and needs to be shortened
function IsNetscape()
{
  global $HTTP_USER_AGENT ;

  if (strstr($HTTP_USER_AGENT, "Mozilla/4") && strstr($HTTP_USER_AGENT, "Win")
     && !strstr($HTTP_USER_AGENT, "MSIE") && !strstr($HTTP_USER_AGENT, "Op-
     era"))
    return 1 ;
  else
    return 0 ;
}

// Do not have any empty lines after the next line, or else PHP will freek out
?>
```

search.js

```
/*
====================
search.js
--------------------
Funksjon:
- skjuler/viser div
====================
*/
/*Viser angitt div*/
function showdiv(layerName) {
   status = document.getElementById(layerName).style.visibility;
   if (status="hidden") document.getElementById(layerName).style.visibili-
ty="visible";
}

/*Skjuler angitt div*/
function hidediv(layerName) {
   status = document.getElementById(layerName).style.visibility;
   if (status == "visible") document.getElementById(layerName).style.visibili-
ty="hidden";
}
```

E.3.5 Stilark (websok.css)

```
/*
```

```
==========
websok.css
==========
*/
body { font-family: Verdana, Geneva, Arial, Helvetica, sans-serif; }

/*******
STRUKTUR
********/

#frame {
   width: 600px;
   margin-right: auto;
   margin-left: auto;
   margin-top: 10px;
   padding: 0px;
   text-align: left;
}

#logo {
   margin-right: auto;
   margin-left: auto;
   padding: 0px;
   text-align: center;
   border: 1px solid #c0c;
}

#sok {
   margin-top: 40px;
   margin-right: auto;
   margin-left: auto;
   padding: 0px;
   text-align: center;
}

#footer {
   font-size: xx-small;
   text-align: center;
}

#frame menu ul {
   border: 0;
   margin: 0;
   padding: 0;
   list-style-type: none;
   text-align: center;
   width: 100%;
}

#menu ul li {
   display: block;
   float: left;
   text-align: center;
   padding: 0;
   margin: 0;
}

#menu ul li a {
   background: url(../img/gradient.gif);
```

```
   width: 130px;
   height: 2em;
   border-top: 1px solid #ccc;
   border-left: 1px solid #ccc;
   border-bottom: 1px solid #ccc;
   border-right: 1px dotted #999;
   padding: 0;
   margin: 0 0 10px 0;
   color: #00008b;
   text-decoration: none;
   display: block;
   text-align: center;
   font-weight: bold;
   letter-spacing: 1px;
   line-height: 2em;
   font-size: x-small;
}

#resources a:link, #resources a:visited {
   border-right: 1px solid #ccc;
}

#menu ul li a:hover {
   color: #000;
   background: url(../img/gradient_visited.gif);
}

#menu a:active{
   background: url(../img/gradient_visited.gif);
   color: #c60;
}

body#sectionone #menu li#one a,
body#sectiontwo #menu li#two a,
body#sectionthree #menu li#three a,
body#sectionfour #menu li#four a,
body#sectionfive #menu li#five a {
   background: url(../img/gradient_active.gif);
   border: 1px solid #c30;
   color: #fff;
   font-weight: bold;
}

/*************
FORM-elementer
*************/
legend {
   font-size: small;
   color: Blue;
}

fieldset {
   border: 1px solid #ccc;
   padding-top: 1em;
   padding-left: 1em;
   padding-right: 1em;
   padding-bottom: 0.5em;
   text-align: left;
}
```

```
fieldset.halvfelt {
   width: 40%;
}

/****
FORM: ID aar1 og aar2 brukes til å skjule/vise to div-seksjoner i dokumentet
****/
#aar1 {
   visibility: hidden;
}

#aar2 {
   visibility: hidden;
}

/************
RESULTATLISTE
************/
dl.margins-removed {
   margin: 0;
   padding: 0;
}

.margins-removed dt {
   margin: 0;
   padding: 0;
   font-weight: bold;
}

.margins-removed dd {
   margin: 0 0 1em 0;
   padding: 0;
}

.resultatterm {
   font-size: x-small;
   font-weight: bold;
   background-color: #eeeeee;
   vertical-align: top;
}

.resultatverdi {
   font-size: x-small;
}

.resultbar {
   font-size: x-small;
}
```