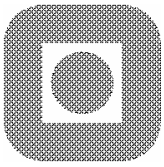
REAP

Et system for rettighetsstyring i digitale bibliotek

Hovedfagsoppgave ved
Institutt for Datateknikk og Informasjonsvitenskap
Norges teknisk-naturvitenskapelige universitet.
september 2002

Forfatter: Øyvind Vestavik
Veileder: Professor Ingeborg Sølvberg

NORGES TEKNISK-NATURVITENSKAPELIGE UNIVERSITET
FAKULTET FOR INFORMASJONSTEKNOLOGI, MATEMATIKK OG ELEKTROTEKNIKK



HOVEDFAGSOPPGAVE I INFORMASJONSFORVALTNING

Kandidatens navn: Øyvind Vestavik

Fag: Informatikk

Oppgavens tittel (norsk): REAP, et system for rettighetsstyring i digitale bibliotek.

Oppgavens tittel (engelsk): REAP, a system for rights management in digital libraries.

Oppgavens tekst:

Målet med oppgaven er å vise at man kan lage systemer som tar vare på rettigheter over intellektuelle arbeider som publiseres gjennom et digitalt bibliotek gjennom å foreslå en modell for publisering av materiale som inkorporerer digital rettighetshåndtering (DRM).

Gjennom en prototype på en adgangsprotokoll for ressurser i et digitalt bibliotek vil oppgaven vise at man kan beskytte rettigheter i forbindelse med publisering og bruk av digitale ressurser. Beskyttelse av rettigheter og adgangskontroll vil i fremtiden være et viktig kompetanseområde for digitale bibliotek da spørsmålet om kontroll av adgang til, og styring av bruk av, digitale ressurser i et nettverksbasert miljø vil være avgjørende for hva slags materiale som vil bli distribuert i et digitalt bibliotek.

Prototypen kalles REAP som er en forkortelse for ”Rights Enforcing Access Protocol”

Oppgaven gitt: august 2000

Besvarelsen leveres innen: september 2002

Besvarelsen levert: september 2002

Utført ved: Institutt for datateknikk og informasjonsvitenskap

Veileder: Professor Ingeborg Sølvberg

Trondheim, 20. september 2002

Prof. Ingeborg Sølvberg

Abstract:

This master thesis is aimed at demonstrating that intellectual property can be published in the Internet by digital libraries in accordance with copyright laws. It proposes an architecture/paradigm for managing rights when publishing information through Digital Libraries involving digital rights management. Through an analysis of copyright laws and international treaties, the needs of Digital Libraries and evolving standards and systems for Digital Rights Management the foundations for a such an architecture are identified. Based on this analysis an architecture/paradigm for a system for digital rights management in digital libraries is proposed. As part of this work a prototype of such a system has been implemented and is documented in this paper.

Om oppgaven:

Denne hovedfagsoppgaven tar sikte på å vise at digitale bibliotek kan publisere intellektuelle arbeider i Internet i overenstemmelse med opphavsretten. Den foreslår en arkitektur/et paradigme for å behandle rettigheter ved publikasjon gjennom Digitale Bibliotek som innebærer digital rettighetshåndtering. Gjennom en analyse av opphavsrettslovgivning og internasjonale avtaler, digitale biblioteks behov og standarder og system for digitale rettighetshåndtering under utvikling fastlegges premissene for en slik arkitektur. Basert på denne analysen foreslås en arkitektur/paradigme for et system for digital rettighetsbehandling for digitale bibliotek. Som del av denne oppgaven har det blitt implementert en prototype av et slikt system, og dette systemet er dokumentert i denne oppgaven.

Forord

Dette er min hovedfagsoppgave ved hovedfagsstudiet i informatikk ved Institutt for Datateknikk og Informasjonsvitenskap, NTNU.

Jeg har skrevet om digitale bibliotek og rettighetsstyring som har vist seg å være et særdeles interessant område. Temaet dekker mange tekniske problemstillinger, men det er kanskje grenseflatene mot juridiske og samfunnsmessige aspekter ved systemer for rettighetsstyring som har inspirert meg mest i arbeidet.

Oppgaven starter med å definere hva digitale bibliotek er og med en beskrivelse av DIGLIB, et forskningsprosjekt på Digitale Bibliotek her ved IDI. Deretter redegjøres det for opphavsretten og hvordan denne stiller seg i forhold til de nye muligheter for konsumering av ressurser som er gjort mulig vha nettverksteknologi. Deretter presenteres ”state of the art” innen digitalrettighetshåndtering før oppgaven går over til å presentere et system for å ta vare på rettigheter i forbindelse med publisering fra digitale bibliotek.

Prosessen med å komme fram til denne oppgaven har vært lang og til tider tung, men jeg vil gjerne takke min veileder Professor Ingeborg Sølvberg for verdifull veiledning. Ikke minst har hennes råd og vink på prosessen med å komme i mål med denne oppgaven vært verdifulle. Uten hennes hjelp hadde jeg nok vært fanget i en limbo mellom tekniske detaljer i all evighet.

Trondheim 20. sept. 2002

Innholdsfortegnelse

Vedlegg

Figurliste

1 Innledning

Internet og datateknikk har gitt oss en enorm tilgang til informasjon de siste ti årene. På samme måte som da Gutenberg revolusjonerte verden med boktrykkerkunsten er Internet i ferd med å revolusjonere verden og dens syn på informasjon igjen. Noen deler verdenshistorien inn i ”før” og ”etter” Internet, og har begynt å kalle den perioden vi nå er på vei inn i som ”informasjonssamfunnet” med klar referanse til andre begreper som ”jordbrukssamfunnet” og ”industrisamfunnet”. Det er klart at Internet, selv om det forløpig i det store og det hele er et fenomen av den industrialiserte verden, innehar potensiale både til å fremme økonomisk vekst og politisk stabilitet og å utfordre etablerte politiske, sosiale og økonomiske maktstrukturer over hele verden.

I flere tusen år har biblioteksinstitusjonene fungert som et samfunns viktigste kilde til viktig informasjon og bærer av kunnskap fra generasjon til generasjon. Biblioteker har samlet inn viktig og kvalitetssikret informasjon på vegne av sine brukere og har gjort denne tilgjengelig for publikum samt preservert denne informasjonen for fremtidige generasjoner. Biblitekstradisjonen ser seg nå utfordret av Internet. På grunn av at informasjonen i Internet er så lett tilgjengelig begynner brukere å velge bort de tradisjonelle bbliotekene som informasjonskilder. I noen tilfeller er dette en positiv utvikling fordi informasjon ofte er tidligere tilgjengelig over Internet enn den vanligvis har vært i biblioteker, men samtidig stiller bruken av Internet større krav til brukeren fordi det ofte er veldig vanskelig å skille kvalitetsinformasjon fra informasjon som kanskje virker tilforlatelig, men som ikke er basert på fagkunnskap eller rasjonelle vurderinger. For at bibliotek skal kunne fungere som et samfunns kilde til kvalitetssikret informasjon og felles hukommelse også i en nettverksbasert verden må bibliotek åpne seg mot Internet og aktivt bruke Internet som innfallsport til den informasjon og kunnskap som er tilgjengelig i bibliotekene. Dette stiller bibliotekstradisjonen overfor enorme utfordringer.

I en verden der informasjon blir viktigere og viktigere blir også informasjonen mer og mer verdt, både for de som skaper informasjonsressurser og for samfunnet som helhet. Etter som det utvikler seg flere og flere institusjoner og firma som baserer sin virksomhet på kunnskap og informasjon blir det viktigere og viktigere å holde viktig kunnskap innenfor disse institusjonene for å beholde et konkurransefortrinn. Dette står ofte i konflikt med et samfunnets behov for kunnskap og informasjon. For å avveie behovene til disse to gruppene har de fleste land en eller annen form for opphavsrettslovgivning. Denne lovgivningen blir nå utfordret av Internet med de muligheter for brudd på opprettshavlige prinsipper som nettverksteknologien besitter. Hvem som har rett til å aksessere informasjon er et av de viktigste og mest omdiskuterte spørsmål i Internetsammenheng i dag.

Fordi man ikke har hatt systemer til å kunne styre hvem som skal ha tilgang til informasjon i nettverksbaserte medier, har eiere og skapere av viktig informasjon vært nølende til å publisere denne informasjon på Internet så langt. Dette har ført til at kvaliteten på informasjonen på Internet har sunket. Her kan bibliotekene spille en viktig rolle ved å fungere som ”øyer” av kvalitetsinformasjon. Forutsetningen for dette er at det

skapes systemer som fungerer som garantister for at ikke opphavsrettslige prinsipper brytes ved publikasjon av informasjon over Internet.

Denne oppgava tar for seg temaet Digital Rettighetsbehandling (Digital Rights Management), og prøver å finne fram til en modell for hvordan man kan lage systemer som tar vare på opphavsrettslige prinsipper i forbindelse med elektronisk publisering fra digitale bibliotek. Gjennom en analyse av opphavsrettens prinsipper, Internets egenskaper og muligheter sett i forhold til opphavsrettens prinsipper, nytt lovverk og systemer og standarder under utvikling forsøker oppgava å gi et bilde av hva digital rettighetshåndtering er i dag og hvordan vi kan lage systemer i fremtiden som tar vare på opphavsrettigheter rettigheter i forbindelse med publisering fra digitale bibliotek i Internet.

2 Hva er digitale bibliotek?

2.1 Definisjoner, avgrensninger og hype-ord

Digitale Bibiotek befinner seg i skjæringspunktet mellom to verdener. På den ene siden har vi den lange tradisjonen med biblioteker som samlinger av kunnskap i form av bøker og skrifter på forskjellige media. Informasjonsressurene er fysisk samlet og vedlikeholdt av organisasjoner av bibliotekarer og andre eksperter på indeksering og katalogisering. På den andre siden befinner digitalteknologien seg med sitt fokus på digitalteknologiens muligheter for distribuerte samlinger med flytende og delvis individuelle dokumenter som kan vedlikeholdes nærmere produsentene av dokumentene. Forskere og andre som er interessert i digitale bibliotek ser ut til å ha en litt forskjellig oppfatning av digitale biblioteker alt etter hvilken av disse verdener de har sitt utspring i. Christine Borgman mener å se at forskere innen digital teknikk ofte ser på digitale biblioteker som innhold som er samlet på vegne av brukermiljøer, mens bibliotekarer ofte ser på digitale biblioteker som institusjoner eller tjenester som viderefører en lang tradisjon i et nytt miljø. **[Borgman2000].** Hun mener også at fordi det har vært relativt god tilgang på finansiering av forskning under paraplyen digitale biblioteker har mange forskere med vidt forskjellig bakgrunn ønsket å komme inn under denne paraplyen. Videre hevder hun at disse ofte har manglet kunnskaper om fagfelter rundt digitale bibliotek, forankret enten i bibliotekstradisjonen eller i digitalteknikken, og at deres bilder og oppfatninger av digitale bibliotek har vært farget av og kanskje begrenset av dette.

I tillegg til dette ser vi at flere og flere som på et eller annet vis publiserer materiale via CD-ROM, databaser og/eller Internet i dag kaller sine samlinger for digitale bibliotek. Grunnen til dette kan være et ønske om at den distribuerte informasjonen skal oppfattes som seriøs og kvalitetssikret fordi digitale bibliotek er en etablert term som det er bekvemt å bruke for å definere innholdet i den distribuerte informasjonen eller rett og slett at termen digitale bibliotek gjør seg godt i markedsføringssammenheng. Digitalt bibliotek har blitt et hype-ord, men forvirringen om hva ”digitalt bibliotek” står for er stor.

Selv om mange forskere innen fagfeltet digitale bibliotek vil si at fagfeltet er godt definert, kan det synes vanskelig for utenforstående å forstå hva som menes med ”Digitale Bibliotek”. Det er vanskelig å finne en entydig definisjon på hva ”Digitale Bibliotek” er, men det finnes mange aspekter ved digitale bibliotek som dekkes av forskjellige definisjoner. Derfor vil jeg ta med et lite utvalg av definisjoner som sammen kan gi et bilde av hva digitale biblioteker er og kan være i fremtiden. Det er også viktig å være klar over at vårt bilde av digitale bibliotek i dag ikke nødvendigvis er det samme som vi hadde for 10 år siden, og helt sikkert ikke er det samme som vi vil ha i framtiden.

En av de tidligste definisjoner fra 1992 ser ut til å fortsatt være en av de mest omfattende. Den sier at ”.. a national Electronic Library is (1) a service; (2) an architechture; (3) a set of information resources, databases of text, numbers, graphics, sound, video, etc; (4) a set of tools and capabilities to locate, retrieve, and utilize the information resources available. **[Borgman2000]**

Termen ”Electronic Library” er idag byttet ut med ”Digital Library” eller ”digitalt bibliotek” på norsk, men ellers vil mange forskere fra det digitale miljø trekke gjenkjennende på skuldrene til denne definisjonen. Mange som har sitt utspring i biblioteksverdenen vil nok kanskje likevel si at digitale bibliotek ikke kan erstatte de tradisjonelle biblioteksinstitusjoner, og vil legge mer vekt på at digitale bibliotek er en videreføring av biblioteksinstitusjonene i en elektronisk verden. Dette kommer tydelig fram i den neste definisjonen:

Digitale biblioteker er ”an institution that performs and/or supports (at least) the functions of a library in the context of distributed, networked collections of information objects in digital form” **[Belkin1998]**

Arbeidsdefinisjon fra Digital Library Federation legger også stor vekt på dette aspektet: *Digital Libraries are organizations that provide the resources, including the specialized staff, to select, structure, offer intellectual access to, interpret, distribute, preserve the integrity of, and ensure the persistence over time of collections of digital works so that they are readily and economically available for use for a defined community or set of communities.* ***[DLFstrategy&business2000]***

Forskere fra datatekniske disipliner vil ofte legge mer vekt på hva som befinner seg i de digitale bibliotek og tjenestene de tilbyr, mens institusjonsaspektet ikke vektlegges så sterkt. Carl Lagooze definerer digitale bibliotek som:

“a managed collection of digital objects (content) and services (functionality) associated with storage, discovery, retrieval and preservation of these objects.” **[Lagoze2000]**

Men også i slike definisjoner finner vi begrepet ”managed” som antyder en slags menneskelig kontroll og vedlikeholdsfunksjon. Nettopp denne spenningen mellom hvilke aspekter som kan automatiseres i digitale bibliotek og hvilke som må ivaretas av ved hjelp av menneskelig fagkunnskap har opptatt Lagoze.

Levy og Marshall **[LevyMarshal95]** mener vi kan definer tre begreper eller perspektiver som til sammen definerer et digitalt bibliotek. Det første er **dokumenter**. Dette tar utgangspunkt i hvilke informasjonsressurser som finnes i det digitale bibliotek. Det andre er **teknologi**, og tar som utgangspunkt hvilken teknologi som brukes, og ikke minst i hvor stor grad den brukes. Det tredje er **arbeid**. Dette tar utgangspunkt i hva slags arbeid som utføres innenfor det digitale bibliotek. Arbeidsaspektet kan vi dele i det arbeid som gjøres når brukere tar i bruk materialet/dokumentene (ofte kalt ”research”) og det arbeid som gjøres av bibliotekarer og andre som er del av biblioteksorganisjonen(ofte kalt ”service”). Forskere og andre fra den datatekniske verden ser ut til å favorisere teknologiaspektet over ”service” aspektet, mens de som har sitt utgangspunkt i biblioteksverdenen mener at teknologi ikke kan erstatte den ekspertise som befinner seg i biblioteksorganisasjonene.

Selv om vi har denne forskjellen i perspektiv og bruksområder kan vi likevel oppsummere en del egenskaper for digitale bibliotek

- Digitale bibliotek er samlingsorientert. Dvs at det finnes en eller flere samlinger av informasjon lokalt eller distribuert som er samlet inn for et formål.

- Informasjonen i digitale bibliotek er organisert i henhold til et paradigme eller en standard på en slik måte at den kan gjenfinnes, gjenhentes, omorganiseres, oppdateres, gjenbrukes og slettes.

- Digitale bibliteker er opprettet og vedlikeholdt på vegne av en brukergruppe.

- Digitale bibliotek innholder kvalitetssikret informasjon

- Digitale bibliotek har tjenester som støtter gjenfinning og bruk av informasjon.

Ved å fristille digitale bibliotek fra de tradisjonelle biblioteksinstitusjonene og fokusere mer på innholdsorganisering og tjenester utvides begrepet digitale bibliotek til også å kunne gjelde for eksempel kunnskapsbaser i store bedrifter og organiserte samlinger av leserinnlegg i en avis som det er knyttet visse tjenester opp mot.

2.2 Relevante kompetanse-områder for digitale bibliotek

2.2.1 Metadata

Tradisjonelt har bibliotek vært veldig opptatt av å beskrive ressurser som befinner seg i deres samlinger ved hjelp av metadata. Metadata kan kalles data om data, og er en beskrivelse av viktige aspekter ved ressursen eller entiteter som er umiddelbart knyttet til ressursen. Disse beskrivelsene har ofte vært meget detaljerte og har krevd at de som skulle håndtere beskrivelsene har hatt høy kompetanse på fagfeltet. Beskrivelsene har vært nødvendige for å kunne finne relevant materiale i store samlinger og for å kunne evaluere ressurser uten å aksessere innholdet i en ressurs.

Digitale bibliotek er også opptatt av å beskrive ressurser vha metadata for effektivt å kunne finne igjen og evaluere ressurser. Det finnes mange skjemaer og forslag til standarder for beskrivelse av materiale for å beskrive ressurser i Internet-sammenheng, men ingen fast standard som alle kan enes om. I mange tilfeller vil det da også være at et metadataskjema som er adekvat i en sammenheng er fullstendig unyttig i en annen sammenheng.

I artikkelen ”Defining metadata” **[Gillian-Swetland]** skiller Ann J. Gillian-Swetland mellom administrative, deskriptive, preservative, tekniske og bruksrelaterte metadata. Administrative metadata angir hvordan man behandler og administrerer informasjonsressurser(inkludert rettighetsbehandling), deskriptive metadata angir beskrivelser av eller identifisering av informasjonsressurser, preservative metadata angir hvordan man kan preservere informasjonsressurser, tekniske metadata angir hvordan et system fungerer eller metadata oppfører seg og bruksrelaterte metadata angir hvordan og i hvilken grad informasjonsressurser brukes. Det er viktig for utviklere av digitale bibliotek å vite hva disse metadataene er, hvordan de kan kodes i henhold til standarder og hvordan de kan brukes i systemer.

2.2.2 Katalogisering og indexering

Systemer for å organisere kunnskap og informasjon er viktige for å kunne holde oversikt over informasjon, evaluere informasjon, og finne igjen informasjon. Teknikker for å organisasere informasjon inkluderer katalogisering og klassifisering av informasjon, indexering av informasjon og thesauribygging.
Digitale bibliotek må for å kunne være effektive ha systemer som kan enten automatisk generere slik kunnskapsorganisering eller systemer som støtter den manuelle generering av slik kunnskapsorganiseringen. Kunnskap om katalogisering og indexering er viktig for å kunne lage systemer som effektivt finner relevant informasjon i sine samlinger

2.2.3 Digitale formater

Før fantes de fleste intellektuelle ressurser i trykt eller håndskrevet form som et fysisk eksemplar som du kunne håndtere fysisk. Dagens ressurser er ofte i digital form, enten som analoge ressurser som er digitalisert (scannet el lignende) eller ressurser som er såkalt ”born digital”, dvs at de var skapt i et digitalt format. Grunnen til å konvertere ressurser til digitale formater kan være å tillate aksess til en kopi av et sjeldent eksemplar av en tekst til mange mennesker i stedet for orginalen for noen få, og rask transport over nettverk for å gjøre ressursen uavhengig av fysisk lokasjon. Kravet til antall hyllemeter blir også drastisk redusert med digitale formater.

Imidlertid setter også disse formatene også begrensninger på hvordan ressursene kan vises fram eller brukes og lagres, og hvilke medier som kan benyttes til dette. For digitale bibliotek er det viktig å kjenne til formatenes sterke sider og begrensninger, slik at man

kan representere materiale optimalt i forhold til kvalitet, overføringshastighet, konvertabilitet og lagringsopsjoner.

2.2.4 Samlingsbygging

Tradisjonelt har biblioteker vært sentrert rundt samlinger av fysiske eksemplarer lokalisert i bygninger. Digitale biblioteker er ikke så opptatt av fysisk lokasjon siden ressurser kan leveres ved hjelp av nettverksteknologi. Dette gjør at man kan bygge samlinger som er basert på organiserings- og gjenhentingsparadigmer uavhengige av fysisk lokasjon. Elementer i samlinger kan oppbevares nærmere produsent og vedlikeholdet av samlinger kan i større grad overlates til produsenter av elementer i samlinger med det digitale bibliotek som det overbyggende system som holder ellers muligens veldig heterogene samlinger og informasjonsenheter sammen.
For å kunne fungere som et slikt overbyggende system må både samlinger og informasjonsobjekter samt tjenester som tilbys brukeren kunne beskrives av metadata, og digitale biblioteker må definere metoder for å gjenhente ressurser. Det krever også at lokale samlinger presenterer en gjenhentings- og presentasjonsprotokoll mot det digitale bibliotek slik at den sentrale overbygningen kan aksessere de lokale samlingene. Kunnskap om hvordan man bygger overbyggende systemer for heterogene nettbaserte samlinger er viktig for å bygge store digitale bibliotek

2.2.5 Reformatteringsteknologi

I mange tilfeller vil det være ønskelig å konvertere et materiale fra et format til et annet. Dette kan være i forbindelse med preservering av materialet eller at det oppdages nye bruksområder for materialet. Det er heller ikke uvanlig at man må reformattere et materiale fordi mediene som var brukt til å presentere materialet ikke er tilgjengelig lengre eller har migrert over til nye formater. Hvordan man kan foreta en slik reformattering vil være et viktig kompetanseområde for digitale bibliotek.

2.2.6 Arbeidsflyt

Digitale bibliotek skal først og fremst yte service til mennesker. Disse er interessert i tilgang til informasjon som støtte i en arbeidsprosess. Ofte er også informasjonsressursen del av arbeidsprosessen, da flere kan sitte samtidig å skrive til en ressurs.(Computer Supported Cooperative Work) Også andre arbeidsprosesser som søking etter informasjon er viktige kunnskapsområder for digitale bibliotek. Uten kunnskap om hvordan brukerne ønsker å bruke og bruker digitale bibliotek og deres informasjonsresurser i arbeidsprosesser kan man ikke lage velfungerende digitale bibliotek.

2.2.7 Opphavsrett

Åndsverksloven gir automatisk opphavsrett over intellektuelt materiale i det øyeblikk det skapes. Ideelle rettigheter hørere alltid til skaperen av verket. Økonomiske rettigheter hører også i de fleste tilfeller til opphavsmannen, med mindre arbeidet utføres som del av et kontraktsforhold der det er deklarert at økonomiske rettigheter tilfaller den som bestiller arbeidet, helst da mot gjenytelser i form av betaling for arbeidet med mer. Mens de idelle rettigheter ikke kan selges eller på annen måte overføres mellom juridiske personer, kan økonomiske og administrative rettigheter overføres og handles med. Fordi digitale biblioteker tar sikte på å inneholde kvalitetssikret materiale må digitale bibliotek ta vare på rettighetene som er knyttet til materialet. Ellers vil ikke noen være villige til å publisere materiale gjennom digitale bibliotek. Internet som distribusjonskanal av natur er ikke begrenset av nasjonalstaters grenser og materiale kan distribueres over landegrenser nesten uhindret. På grunn av dette må digitale biblioteker også ta hensyn til internasjonal lov, og i mange tilfeller mangelen på slikt lovverk.

2.2.8 Lisensiering

I mange tilfeller vil det ikke være hensiktsmessig å overføre alle rettigheter fra en person til en annen. I mange tilfeller kan det være mer hensiktsmessig å tildele noen et sett av rettigheter over et materiale uten at de dermed får for eksempel eiendomsrett eller administrasjonsretter til et materiale. Typisk i et digitalt bibliotek kan man se for seg at det digitale bibliotek får tillatelse til å publisere materiale fra andres samlinger gjennom sine systemer, mens vedlikehold og administrasjon av samlinger fortsatt påligger eier av materialet. Kunnskap om hvordan man kan sette opp slike avtaler og gjennomføre intensjonene i en slik avtale vil ofte være et viktig kompetanseområde for digitale bibliotek.

2.2.9 Preservering av digitale ressurser

Både digitale og analoge medier og ressurser kan ofte preserveres ved hjelp av digital teknologi. Digitale kopier av fysisk materiale som i original versjon er i dårlig forfatning, eller som ville utsettes for unødvendig slitasje ved utstrakt håndtering, kan støtte preservering av originalen. Forskere med flere kan i mange tilfeller jobbe med kopien og dermed preserverer originalen. Dessuten har digital lagring potensiale i seg til å kunne preservere materiale over lengre tid enn analoge fysiske medier. Gammelt papir er avhengig av kontrollert miljø for ikke å enten bli sprøtt eller rotne, nytt syreholdig papir kan ikke sies å ha veldig lang holdbarhet og magnetiske medier avmagnetiseres. Digital lagring gjør at materialet kan kopieres fra et medium over til et annet og fra et format til et nytt uten tap av kvalitet. Det betyr at materialet blir gjort uavhengig av mediet som bærer det og dermed kan eksistere lenge etter at det orginale mediet er ødelagt eller utdatert. Men hvis denne transformasjonen blir utsatt for lenge er det store sjanser for at materiale ikke lengre kan spilles av eller at mediet som er bærer av ressursen er ødelagt. For digitale bibliotek som forvaltere av informasjon er det derfor viktig å vite hvordan man preserverer informasjon slik at den vil være tilgjengelig også i fremtiden.

2.2.10 Brukergrensesnitt

For å kunne lage hensiktsmessige digitale bibliotek er det viktig å ta hensyn til brukerne og deres behov. Det digitale bibliotek må være tilpasset deres behov og oppførsel. Dette gir føringer for utseende av og funksjonalitet i grensesnitt mot klienten, og dermed for resten av systemet. For eksempel er det viktig at søkeprosessen i systemet er tilpasset hva brukeren føler er ”riktig” framgangsmåte for søking. Det bør være brukerens behov som styrer utviklingen av systemet, ikke systemutviklerens. Kunnskap om hvordan mennesker bruker og interaksjonerer med datamaskiner og datasystemer, såkalt ”Human-Computer Interaction” er således et viktig kompetanseområde for digitale bibliotek

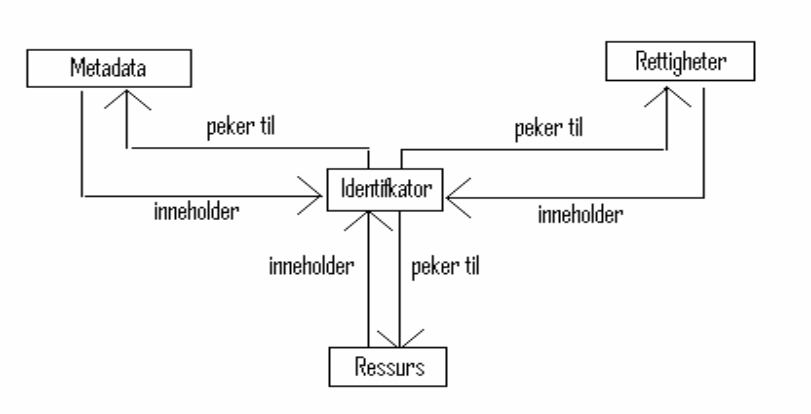
2.2.11 Tilgangskontroll, rettighetsstyring og sporing av bruk

Kontroll av tilgang og rettighetsstyring mottar økende interesse i det forskningsmiljøet for digitale bibliotek og generelt ellers i miljøer som er interessert i digital publisering. Dette er et relativt lite utviklet felt innen digitale bibliotek, men vil antagelig få større og større relevans etter som fagfeltet modnes. Grunnen er at det ikke vil være mulig å kombinere digital nettverksbasert distribusjon av kvalitetsmateriale uten å håndheve rettigheter som er knyttet til materialet, styre distribusjonen av materialet, og ta betalt for distribusjon av materialet. Rettighetshavere vil rett og slett ikke være villig til å publisere sitt materiale i digitale bibliotek dersom ikke dette aspekt av digitale bibliotek tas vare på. Dette er således et viktig kompetanseområde innen digitale bibliotek. Kunnskap om og utvikling av systemer og standarder for å gi tilgang til informasjon etter en deklarert politikk som også kan kontrollere hva en bruker gjør med et materiale vil være essensielt for å kunne oppfylle digitale bibliotekers målsetning om å kunne levere kvalitetsmateriale over nettverksbaserte medier.

2.2.12 Persistente identifikatorer

Persistente identifikatorer er ”limet som holder alt sammen” innen digitale bibliotek. Bruk av persistente identifikatorer gjør at vi kan knytte sammen de tre viktige entitetene **”ressurser”**, **”metadata”** og **”rettighetersbeskrivelser”**.

Metadata inneholder nesten alltid en referanse til materiale de beskriver og ofte til rettighetsbeskrivelser der disse finnes (som forørig kan ses på som en spesialisert form for metadata). Rettighetsbeskrivelser inneholder nesten alltid en peker til ressursen de gjelder for, og ressurser *kan* inneholde pekere til både metadata og rettighetsbeskrivelser selv om dette er relativt uvanlig.



Figur 2-1Persistente identifikatorer binder sammen metadata, ressurser og rettighetsbeskrivelser

De mest brukte identifikatorer på Internet i dag er URL (Uniform Resource Locator). Disse angir hvor et materiale er lokalisert og hvordan det kan gjenhentes. Dette har i lang tid vist seg som en fruktbar løsning for enkel presentasjon av informasjon, men i forhold til digitale biblioteker er ikke dette nok. URL og lignende lokasjonsbaserte identifikatorer knytter identifikasjonen av et materiale til hvor informasjonen er plassert og ikke til materialet selv. Hvis materialet flyttes er identifikatoren ikke lengre gyldig eller peker til en annen ressurs som har tatt den første ressursens fysiske lokasjon.

Persistente identifikatorer skal i motsetning til lokasjonsbaserte identifikatorer være både lokasjons- og tidsuavhengige. Det vil si at selv om ressursen flyttes skal likevel identifikatoren være gyldig, dvs kunne brukes til å gjenfinne materiale, rettighetsbeskrivelser og metadata og at identifikatoren skal være unik også over tid. Slik knyttes identifikatoren til materialet selv og ikke til egenskaper ved materialet som format og plassering.

Når man frigjør identifikatorer fra lokasjon og gjenhentingsmetode vil man være avhengig av en oversettertjeneste som kan ta en persistent identifikator som input og returnere en fysisk lokasjon som ressursen kan hentes fra og en angivelse av hvordan den kan gjenhentes. Fordelen med slike oversettertjenester er at materiale kan flyttes og gjøres tilgjengelig i nye medier/over nye kanaler. Det eneste som må gjøres for at ressursen fortsatt skal være tilgjengelig er at oversettertjenesten oppdateres slik at den til enhver tid returnerer aktuell lokasjon og gjenhentingsmetode gitt den persistente identifikatoren som input.

Det finnes flere slike systemer i dag, selv om de ikke er utbredt på Internet som helhet. Tre eksempler på slike identifikatorsystemer er Digital Object Identifier (DOI) **[DOI]**, Persistent Uniform Resource Locator (PURL) **[Purl]** og Uniform Resource Names (URN) **[URN]**.

2.2.13 Submit-systemer

Noen digitale bibliotek er mest opptatt av å fungere som den digitale ekvivalenten til tradisjonelle bibliotek og er mest opptatt av å gi tilgang til informasjon som er samlet sammen av profesjonelle fagfolk på vegne av av et publikum. For disse vil ikke muligheten for brukerne til å legge til egen informasjon være så viktig. Andre digitale bibliotek, ofte kalt applikasjonsbibliotek eller informasjonssystemer inneholder ofte i tillegg til funksjonaliten til ordinære digitale bibliotek muligheten til å bruke informasjonen direkte i arbeidsprosesser, oppdatere informasjonen i det digitale bibliotek og legge til ny informasjon etter behov. For slike bibliotek vil forskjellige varianter av submitsystemer være av vital viktighet for å kunne opprettholde et bibliotek som gjenspeiler for eksempel en bedrifts samlede kunnskapsmasse. Dette et forskningsområde som inneholder viktige problemstillinger som kvalitetssikring, dynamisk tildeling av persistente identifikatorer, fysisk organisering av materialet i samlinger, generering av adekvate metadata for materialet osv.

2.2.14 Metatjenester

Metatjenester holder oversikt over hvor de andre ressurser og tjenester befinner seg. Dette er spesielt aktuelt i distribuerte systemer, der noder samarbeider om å tilby brukeren ressurser og tjenester fra et vidt spekter av tilbydere.

2.2.15 Må vi alle være eksperter?

Selvfølgelig kan ikke alle være eksperter på alle disse kompetanseområdene. Men det er viktig å være klar over, og til en viss grad kjenne innholdet i, sentrale problemstillinger innen fagområdene for å kunne sette sitt eget arbeid i en sammenheng og for å kunne være med å lage fremtidens digitale bibliotek.

3 Kort beskrivelse av DIGLIB

DIGLIB, (navnet er utledet fra ”DIGital LIBrary”) er et prosjekt som tar sikte på å støtte forskning og utvikling i forbindelse med fagfeltet digitalt bibliotek ved IDI, NTNU, og å gi en plattform for uttesting og utvikling av teorier rundt og mulige løsninger på problemer som finnes innen fagfeltet digitalt bibliotek. Prosjektet ble påbegynt våren 1999, og har vært støttet av Norges forskningsråd. Ansvarlig for prosjektet er Professor Ingeborg Sølvberg ved IDI, NTNU.

DIGLIB består av diverse samlinger og pågående og avsluttede delprosjekter i form av hovedfagsoppgaver, doktorgradsoppgaver og prototyper på løsninger som er utviklet i forbindelse med disse. Det har også vært kjørt diverse prosjekter som har tatt sikte på å utvikle DIGLIBs grunnleggende arkitektur slik at systemet skal kunne fungere som den

plattform for prototyping og utvikling som det var ment å være. Alle prosjektene jobber opp mot en felles, løst definert arkitektur som består av en base av informasjonsobjekter (samlinger) med metoder for aksessering, diverse tjenestemoduler som blant annet fungerer som innfallsport til ulike måter å hente frem informasjon på og diverse grensesnitt mot brukere. DigLibs samlinger er ikke konsentrert om et spesielt tema eller fagområde da de stort sett er hentet inn etter som behovet for en viss type materiale har meldt seg i forbindelse med utforskning av diverse aspekter innenfor fagfeltet digitale bibliotek. Prosjekter som foregår ved IDI idag er sentrert rundt følgende oppgaver: Utforskning av flere lyd og bildesamlinger, bruk av "The Handle System" for unik og persistent navnsetting av objekter i samlingene og samlingene selv, forskjellige tilnærminger til gjenhenting og presentasjon av multimedia objekter fra forskjellige databaser, forskning på relasjoner i innhold og struktur og systemarkitektur.

I sin nåværende form er DIGLIB en samling av mer eller mindre enkeltstående prosjekter og samlinger. Samlingene er ikke konsentrert om et spesielt tema eller fagområde da de stort sett er hentet inn etter som behovet for en viss type materiale har meldt seg i forbindelse med utforskning av diverse aspekter innenfor fagfeltet digitale bibliotek. DIGLIB er primært ikke ment å yte noen slags tjenester til et publikum, og slik sett er det ikke noe problem at DIGLIB ikke utgjør noe enhetlig brukervennlig system. På mange måter kan man si at DIGLIBs dynamiske natur er en av egenskapene som gjør prosjektet fruktbart som testplatform innen digitale bibliotek. Samtidig er det ønskelig å ta DIGLIB et steg videre til et fullversjons digitalt bibliotek for å høste videre erfaringer med byggingen av et helhetlig system. De harde realiteter er vel likevel at DIGLIB mangler ressursene som skal til for å oppnå dette målet både når det gjelder finansiering, organisasjon og menneskelige ressurser.
DIGLIB har innledet et samarbeid med prosjektet "Alexandria Digital Library Project" **[Alexandria]** ved University of California, Santa Barbara om implementering av en node i deres system som skal gi adgang til DigLibs samlinger gjennom deres system.

Av samlinger i DIGLIB finnes "**Trobib",** en samling av metadataposter som beskriver litteratur om Trondheim før 1990, en tredelt **bildesamling** som består av luftbilder, landskapsbilder og kart primært fra Trøndelag og **"Spirit of the Vikings"**, som er en samling av radiosendinger fra andre verdenskrig, hentet fra Nasjonalbiblioteket.

Tjenester under Diglib består av avsluttede prosjekter som har utforsket delaspekter ved digitale biblioteker gjennom generering av prototyper og delsystemer. Av slike avsluttede ***prosjekter*** kan nevnes **"Trobib Search (TS)"**, et system for søking i samlingen "Trobib", **"Ididok"**, et system for lagring av, søking etter og gjenhenting av hovedfagsoppgaver ved IDI, NTNU. **"Libxml"**, en xml-lagringsmodell som bruker Versants ODBMS som lagringsmedium, "**Distributed Relation System"** et system for lagring av relasjoner realisert som pekere i Xlink og **"Xml-o2-matic**", et system for å bruke xml i o2, en objektorientert database. For Diglib er det også opprettet en Handle System navneserver (http://hdl.idi.ntnu.no) og en egen navneautoritet (1009).

Det er ikke definert et eget grensesnitt mot DIGLIB som digitalt bibliotek, men det er satt opp hjemmesider for prosjektet der mange av prototypene er lagt ut. Disse sidene finnes på http://fenris.idi.ntnu.no/

4 Om oppgaven

4.1 Oppgavedefinisjon

Oppgaven går ut på å finne fram til en systemarkitektonisk modell for digital rettighetshåndhevelse i digitale bibliotek. Mer spesifikt skal det finnes fram til en hensiktsmessig arkitektur for en adgangsprotokoll for å beskytte rettigheter til rettighetsbeskyttet materiale i DIGLIB som skal publiseres gjennom ADEPT.

4.2 ”The problem at hand”

Hvis Internet skal vokse til å bli en skikkelig distribusjonskanal for kvalitetssikret informasjon må de skapes systemer som kan styre distribusjonen av rettigheter over et materiale fra forfatter gjennom flere mellomledd til en sluttbruker. De samme systemene må også styre betalinger for overføring av rettigheter over et materiale samt sikre at bruken av materialet er i henhold til de rettigheter som er tildelt. En del slike systemer og standarder som støtter de finnes allerede i dag, men det er få som tar sikte på å beskytte innholdet i digitale bibliotek. Hvis Digitale Bibliotek skal kunne fungere som ”øyer” av kvalitetssikret informasjon i Internet i dag er det en forutsetning at det utvikles slike systemer som er tilpasset behovene til Digitale Bibliotek.

5 Design av arkitektur i et digitalt bibliotek

Digitale bibliotek er store komplekse systemer med mye og avansert funksjonalitet. Det vil ikke være mulig å holde oversikt over et slikt system hvis de ikke er bygd etter modeller og arkitekturer som lar utviklerne dele opp funksjonaliteten i håndterbare enheter etter et arkitektonisk paradigme. Dette vil også være viktig i utviklingen av REAP som er temaet for denne oppgava.

5.1 Hva er systemarkitektur?

Systemarkitektur er et begrep som beskriver den interne oppbygging av og den interne kommunikasjon i et datasystem som skal utføre en gitt oppgave, samt hvordan systemet kommuniserer med omverdenen. For store systemer er det viktig å komme fram til en hensiktsmessig, fornuftig og konseptuelt intuitiv oppbygging av systemet slik at systemet lett kan utvides eller bygges om ved behov, feil kan identifiseres og rettes opp uten å

introdusere nye feil, og ny funksjonalitet kan legges til basert på allerede implementert funksjonalitet.

Systemarkitektur består av alle opplysninger som er nødvendig for at vi skal kunne forstå og diskutere et system og dets funksjonalitet. Disse opplysningene kan da angi:

- hvilke komponenter som inngår i systemet
- hva disse komponentene gjør hver for seg
- hvordan komponentene kommuniserer mellom seg
- hvordan komponentene påvirker hverandres funksjonalitet
- Hvordan komponenter samarbeider for å løse en oppgave.
- Hvilke komponenter som kommuniserer med omverdenen og hvordan denne kommuniksjonen ivaretas.

Disse opplysningene kan brukes til å bygge en konseptuell modell over et system. Modeller er viktig for å bedre forståelsen for hvordan et system fungerer eller å spesifisere og formidle hvordan et system er ønsket å fungere. De abstaherer bort og forenkler kompliserte detaljer og framhever sentrale aspekter ved et system. En konseptuell modell kan i neste omgang brukes til å bygge en fysisk modell som angir komponenter, kommunikasjon og databehandling i systemet og kommunikasjon med omverdenen. En slik modell kan ses på som byggetegninger for systemet og danner grunnlaget for implementasjonen av systemet

5.2 Ideelle krav til arkitektur for digitale bibliotek

I artikkelen *Key Consepts in the Architecture of the Digital Library* **[Arms95]** angir forfatteren 8 aspekter som en arkitektur for digitale bibliotek bør ta hensyn til.

- Den tekniske løsningen eksisterer innenfor et juridisk og sosialt rammeverk. Et digitalt bibliotek må ta hensyn til åndsverkslovgivning og opphavsrettigheter, lover og retningslinjer ang. kommunikasjon over nettverk, hensyn til brukere og andres rett til privatliv og personvernlovgivningen. I og med at et digitalt bibliotek ikke er bundet av nasjonalstatlige grenser må man i mange tilfeller ta hensyn til at lovgivningen kan variere i forskjellige land. Det er viktig at en arkitektur etablerer klare grenser for hvem/hva som er ansvarlig for å opprettholde respekt for overholdelse av slike retningslinjer.

- Forståelsen av konsepter innen digitale biblioteker er hemmet av terminologi. Mange ord som det kunne vært naturlig å bruke i beskrivelsen av er systemarkitektur har så forskjellig betydning i forskjellige kontekster eller har så mange forskjellige konotasjoner at bruken av de i mange tilfeller vil føre til misforståelser. Mange av disse misforståelsene har så stor innflytelse på forståelsen av arkitekturen at ordene rett og slett bør unngås, eller i alle fall brukes med stor forsiktighet. For digitale biblitek er det etter hvert utviklet en slags egen

terminologi, og det anbefales at denne brukes for å unngå misforståelser. Se for øvrig avsnittet. Definisjon av begreper brukt i oppgaven.

- Den underliggende arkitektur bør skilles fra materialet som er lagret i biblioteket. Funksjonalitet i et digitalt bibliotek bør aldri basere seg på egenskaper ved materialet som lagres i et digitalt bibliotek. Slik kan man lage et system som kan lagre mange typer informasjon. Dessuten: Materialet lagret i et digitalt bibliotek kan være av mange forskjellige slag. Man bør da skille mellom funksjonalitet som er gyldig for alle slags materiale, og den funksjonalitet som er nødvendig for å støtte spesifikke typer materiale. Felles funksjonalitet kan da fungere som en plattform for videre utvidelser av systemet til støtte for flere materialtyper.

- Navn og identifikatorer er basis byggeklosser for digitale bibliotek. Man trenger man metoder for å referere til materiale som er lagret i et digitalt bibliotekssystem, koble sammen materiale som på en eller annen slags måte står i relasjon til hverandre og og knytter relasjoner mellom materiale og mennesker eller juridiske personer. Slike navn må være unike og persistente.

- Objekter i det digitale bibliotek er mer en bare en rekke med bits. Alle digitale objekter har en viss indre struktur der elementene står i forhold til hverandre. I tillegg til at deler av materiale kan stå i forhold til hverandre og for eksempel aggregere et større hele(Eksempel: Kapitler i en bok) kan også et digitalt bibliotek inneholde diverse typer beskrivende metadata. Ved å kombinere slike strukturer kan nesten alle strukturer for innhold dannes. Arkitekturen bør støtte en slik dynamisk generering av objekter for materialet det tar vare på.

- Objektet som blir brukt av brukeren er annerledes enn det objektet som er lagret. For å kunne lagre informasjon på en effektiv måte, er det som regel nødvendig å organisasere informasjonen på en måte som ikke nødvendigvis er intuitiv for brukeren. En lagringsmodell bør støtte dynamisk oppbygging av innhold som kan presenteres i mange alternative visninger basert på brukerens valg eller forutsetninger. Slik kan informasjon kombineres på nye måter og danne ny kunnskap og eller forståelse av materialet.

- Repositories må ta vare på materialet de lagrer. Dette inkluderer å ta vare på intellektuell eiendom, og administrere denne eiendommen innenfor økonomiske og sosiale rammer. Dette er spesielt viktig for informasjonsenheter som utgjør inntektsgrunnlag for opphavsmannen. Utfordringen er altså å distribuerere materiale samtidig som man hindrer misbruk og uatorisert bruk av materialet. Det betyr å sikre at kun tillatte operasjoner utføres, og at disse operasjonene kun utføres av brukere som skal ha lov til det. Den indre struktur for lagring av materiale bør derfor være intern og all kommunikasjon med repositoryet bør gå gjennom et definert grensesnitt.

- Brukere ønsker intellektuelle arbeider, ikke digitale objekter. Selv om det finnes mange måter et verk kan lagres på er det viktig at brukerne av det digitale

bibliotek får presentert verket på en måte som er i overenstemmelse med hva som fra brukerens side oppfattes som et verk og at denne representasjonen ikke blir influert av hvordan det digitale bibliotek lagrer og behandler verket internt.

5.3 Funksjonelle krav til arkitektur for digitale bibliotek

I tillegg til slike ideelle krav kan man også sette opp en rekke funksjonelle krav til arkitekturer i et digitalt bibliotek. Disse er nødvendige for å kunne skape et digitalt bibliotek som er funksjonelt, utvidbart og skalerbart.

- Må inneholde en modell for lagring av informasjon. Digitale bibliotek gir adgang til informasjon. Denne informasjonen kan være lagret i egenoppbygde samlinger eller kan befinne seg i samlinger som det digitale bibliotek gir tilgang til gjennom sitt system. Det digitale bibliotek må da gi tilgang til materialet på en slik måte at lokasjon og implementasjon ikke har innvirkning på hvordan materialet presenteres for brukeren. Dette krever at lagringssystemet bygges med et felles grensesnitt mot andre deler av systemet uavhengig av underliggende implementasjon og lokasjon.

- Må inneholde en modell for kartlegging av og organisering av informasjon vha metadata. Informasjonen som det digitale bibliotek gir tilgang til må beskrives vha metadata. Dette gir mulighet for avanserte søk og også en oversikt over hva slags materiale som finnes i samlingene og hvilke samlinger det digitale bibliotek gir tilgang til. Metadata gir også mulighet til å evaluere materiale uten å aksessere det. Bruk av metadata er et av de viktige kjennetegnene ved digitale biblioteker.

- Må inneholde en modell for søking etter informasjon. Informasjon må kunne gjenfinnes hurtig og enkelt uten å måtte aksessere informasjonen. Dette kan gjøres ved å søke i metadatakataloger for samlingene.

- Må inneholde en modell for gjenhenting av informasjon. Informasjonen må kunne gjenhentes fra et repository og presenteres for brukere basert på en unik identifikator for materialet.

- Må inneholde en modell for styring av tilgang til informasjon. IPR (Intellektual Property Rights) policies må kunne uttrykkes og håndheves slik at digitale rettigheter til materialet ikke korrumperes av bruken i det digitale bibliotek.

- Må inneholde en modell for presentasjon av informasjon til brukeren. Fordi informasjonen ofte er lagret på en måte som ikke er intuitiv for brukeren, må ofte informasjonen som er lagret behandles etter gitte regler slik at den kan presenteres for brukeren slik brukeren ønsker å få den presentert.

- Kan inneholde en modell for støtte for betalingstjenester. I mange digitale biblioteker vil det være naturlig at materiale som skal publiseres er av en slik art

at det kreves betaling for å laste den ned. I disse tilfeller er det naturlig at det digitale bibliotek inneholder en modell for hvordan betaling skal innhentes på vegne av de som eier materialet og hvordan denne betalingen skal viderebefordres til eieren.

- Må kunne presentere informasjon slik at brukeren kan nyttiggjøre seg informasjonen. Selv om digitale biblioteker lagrer digitale objekter som kan kombineres på forskjellige måter er det viktig å tenke på at brukeren ønsker å se innholdet på et høyere abstraktsjonsnivå. Det betyr at digitale biblioteker må innkorporere visningsapplikasjoner som kan presentere kombinasjonen av forskjellige informasjonsobjekter for brukeren som en logisk helhet.

6 Opphavsrett

6.1 Hva er opphavsrett?

Opphavsrett er en juridisk beskyttelse av de økonomiske og ideelle rettigheter som anerkjennes en skaper av et intellektuelt arbeid.
Opphavsretten reguleres i Norge av *"lov om opphavsrett til åndsverk m.v."* av 12. Mai 1961 **[Åndsverksloven]** og diverse overnasjonale avtaler som Norge har ratifisert. Dette gjelder i første rekke Verdenskonvensjonen om opphavsrett av 1971 og Bern-Konvensjonen av 1948 **[Bernkonvensjonen]** Store deler av Bernkonvensjonen er også oversatt til norsk. **[BernkonvensjonenNorsk]**

De internasjonale avtaler som Norge har ratifisert bygger i hovedsak på to prinsipper, nemlig "nasjonal behandling" og "materielle minimumsregler".
Nasjonal behandling betyr at opphavsmenn og verk fra andre land skal ha samme beskyttelse som landets egne opphavsmenn og verk, og materielle minimumsregler angir det minimum av vern som landet som har ratifisert avtalen må gi til opphavsmenn og verk. Det er viktig å huske at disse prinsippene bare gjelder for land som har ratifisert avtaleverket, selv om også land som ikke har ratifisert avtaleverket eller deltar i samarbeidet kan ha god lovgivning som beskytter opphavsrett innenfor sine landegrenser.

Opphavsretten oppstår i Norge idet verket er skapt. Man må altså ikke søke eksplisitt om beskyttelse av et verk. I de fleste tilfeller vil det være opphavsmannen som innehar alle rettigheter til et verk. Unntaket er hvis verket er skapt som del av et kontraktsfestet forhold hvor det på forhånd er avklart at opphavsretten til materialet tilhører en av partene i forholdet. Ansettelsesforhold i bedrifter og organisasjoner inneholder ofte klausuler om at verk som skapes i forholdet tilhører bedriften/organisasjonen.

For at et verk skal være beskyttet må det være av en viss verkshøyde, dvs at det må ligge en viss skapende innsats bak verket og verket må være av et visst omfang. Det er viktig å vite at opphavsretten ikke omfatter ideer og tanker bak et verk, bare verket selv.
Økonomiske rettigheter er i Norge begrenset til 70 år. Det vil si at etter 70 faller opphavsmannens økonomiske eneretter bort. Det vil likevel være slik at mange av

opphavsmannens ideelle eneretter ikke vil falle bort etter 70 år. Opphavsretter kan gå i arv etter videre betingelser angitt i åndsverksloven

Opphavsretten kan deles i ideelle rettigheter og økonomiske rettigheter. De økonomiske rettighetene er rettighetene til å råde over verket, mens de ideelle rettighetene gjelder retten til å bli navngitt og vern mot krenkende gjengivelse osv.

De økonomiske rettighetene kan igjen deles i *enerett til eksemplarfremstilling* og *enerett til å gjøre tilgjengelig for offentligheten.* Eksemplarframstilling defineres av Jon Bing som slik: ”Med eksemplarfremstilling siktes det til enhver varig representasjon av verket – typisk på papir eller lerret. Men også maskinlesbare representasjoner er eksemplarer” **[Bing].** Å gjøre tilgjengelig for offentligheten kan ifølge ånsverksloven gjøres på tre måter. Den første er *spredning*. Dvs at allerede fremstilte eksemplarer av et verk tilbys allmennheten. Eksempler på spredning kan være salg, utlån eller utleie av bøker eller filmer. Den andre er *fremføring.* Dette er typisk fremføring av sanger, dikt og musikkverk der artisten fungerer som et medium som fremfører et verk for et publikum. Men også fremvisning av verker som krever et ikke-menneskelig medium (for eksempel TV-skjermer, Radioer eller dataskjermer) regnes som fremføring. Den tredje måten er fremvisning. Her er det mer snakk om fremvisning av fysiske artifakter som for eksempel skulpturerer eller malerier i et kunstgalleri eller sjeldne fisker i et akvarium som er åpent for publikum. Forskjellen mellom framføring og framvisning er at en framvisning ikke krever et medium.

Ideelle rettigheter kan ikke overføres. Økonomiske rettigheter derimot kan enten i sin helhet eller delvis selges, lisensieres eller på annen måte overføres fra opphavsmannen til andre juridiske personer. Det mest vanlige er da at man overdrar deler av opphavsretten, eller visse beføyelser som det også kalles, enten for en periode eller for resten av opphavsrettens levetid (70 år).

Det finnes ikke krav til hvordan slike avtaler skal utformes, og enhver type kompensasjon innen lovens grenser kan derfor avtales for en overføring av opphavsrettslige beføyelser. Økonomisk kompensasjon er selvfølgelig mest vanlig. Slike avtaler er ifølge loven statiske. $39 i åndsverksloven sier at ”Har opphavsmannen overdratt rett til å bruke verket på en bestemt måte eller ved bestemte midler, har erververen ikker rett til å gjøre det på andre måter eller ved andre midler.” Så hvis noen for 20 år siden kjøpte retten til å fremstille et gitt verk i bokform og selge, betyr ikke det at de har rett til å publisere boka som e-book selv om denne teknologien ikke var kjent da overføringa av eksemplarfremstillingsretten ble overført. Overføring av opphavsrett eller opphavsrettslige beføyelser gir i all hovedsak ikke rett til å forandre det opprinnelige verket

Åndsverkslovgivningen i Norge og andre land forsøker å balansere to behov. På den ene siden skal åndsverksloven beskytte opphavsmenn og deres retter, men samtidig er hensynet til samfunnsopplysning, forskning og teknisk og sosial utvikling i mange tilfeller så viktig at det rettferdiggjør unntak i opphavsretten.

Åndsverkslovens kapittel 2 definerer en del slike begrensninger i opphavsretten. Dette gjelder blant annet enhvers rett til begrenset fremstilling av eksemplarer til privat bruk, eksemplarfremstilling og fremføring til spesielle formål (som for eksempel undervisning, institusjoner, gudstjenester, reportasjer osv) og eksemplarfremstilling for bruk som sikkerhetskopi eller fremstilling av eksemplarer som er nødvendige for den forutsatte bruk. I USA kalles dette for ”fair use” og er nedfelt i ”Copyright law of the United States of America” **[Copyright law]**. Det er ikke alltid så lett å vite hva som kan unntas opphavsretten under henvisning til kapittel 2 i åndsverksloven, og i mange grensetilfeller har det vært nødvendig å søke avklaring gjennom rettsapparatet. I mange tilfeller varierer det også sterkt over landegrensene hva som anses som rettferdiggjorte unntak fra opphavsretten.

I Norge kan offentlige og private institusjoner og firma som ønsker å bruke rettighetsbelagt materiale i en eller annen form tegne avtaler om dette med organisasjonen Kopinor **[Kopinor]** som igjen har avtaler med en lang rekke rettighetshavere. Institusjoner og firma betaler en avgift til Kopinor som på sin side distribuerer denne avgiften til rettighetshaverne. Kopinors søsterorganisasjoner finnes i de fleste land med en skikkelig beskyttelse av intellektuelle rettigheter over litterære og kunstneriske verk.

6.2 Hvordan overføres rettigheter?

Som nevt i avsnittet over kan økonomiske beføyelser av opphavsretten til et verk overføres mellom juridiske personer. Uten slik mulighet til å overføre rettigheter over opphavsrettseskyttede verker ville faktisk verkene i seg selv ha en begrenset verdi, da opphavsmannen i seg selv sjelden har mer enn en begrenset mulighet til å fremstille eksemplarer av verket og gjøre det tilgjengelig for omverdenen. Retten til slik eksemplarfremstilling og spredning må i praksis selges til personer eller organisasjoner som har den nødvendige kunnskap og finansielle styrke til å sette eksemplarfremstillingen inn i en produksjon og som råder over et distribusjonsapparat som kan gjøre bøker og andre arbeider tilgjengelig for offentligheten. De fleste som lager intellektuelle arbeider av en viss sort er jo opptatt av at det som skrives eller skapes skal nå ut til et publikum og at det gjennom det kanskje kan oppnå en viss økonomisk kompensasjon for sitt arbeid.

De fleste som skaper intellektuelle arbeider av tekstuell art vil for eksempel forsøke å selge, gi bort eller lisensiere en del beføyelser av sin opphavsrett til et forlag mot et vederlag. Vederlaget kan være av økonomisk art eller av mer ideell art som en lovnad om å utgi verket i bokform i hele Europa. Forlaget blir da en rettighetsholder til hele eller deler av opphavsretten som tidligere var forbeholdt opphavsmannen.
Forlaget kan så selge videre eller lisensiere retten til eksemplarfremstillelse til et trykkeri som lager bøker av verket. Når bøkene er ferdige kan forlaget selge eksemplarene til en distributør som selger eksemplarene til bokhandlere i hele Europa. Når vi går i en bokhandel eller handler bøker på Internet kjøper vi så retten til å lese boka.

Rettighetsoverføring skjer altså i en potensielt lang rekke av avtaler mellom juridiske personer. I alle ledd gjelder prinsippet om at en rettighetsholder kun kan overføre rettigheter til en tredjeperson som rettighetsholderen har fått tillatelse til å overføre gjennom avtale med rettighetsholderen over seg i kjeden. Avtaler vil da typisk inneholde to aspekter. Det ene er rettigheter over verket selv og det andre det jeg vil kalle metarettigheter over verket., dvs retten til å gi videre rettigheter eller metarettigheter over materialet til en tredjeperson.

6.3 Tradisjonell håndhevelse av rettigheter

[RosenblattTrippeMooney] deler rettigheter over et materiale i tre deler. *Juridiske rettigheter* er rettigheter som du som skaper av et intellektuelt arbeid mottar automatisk gjennom åndsverksloven og internasjonale avtaler eller som gis for eksempel i form av et patent etter å ha søkt om slik beskyttelse. Som nevnt i avsnittet om opphavsrett kan en del rettigheter overføres i en transaksjon mellom rettighetshavere. Rettigheter som er overført fra en juridisk person til en annen kaller vi *overførte rettigheter*. I tillegg til disse rettighetene definerer **[RosenblattTrippeMooney]** også noe de kaller *implisitte rettigheter*. Dette er rettigheter som er definert av mediet som er bærer av det intellektuelle arbeidet.

Et eksempel på implisitte rettigheter oppstår når en person kjøper en bok i en bokhandel. Mot at man betaler en viss sum får man en del rettigheter over boka. Man kan lese boka så mange ganger man vil, kaste den, brenne den, rive ut det kapitlet der helten blir drept eller gi den bort til en kamerat. Alt dette er eksempler på implisitte rettigheter. Men man har ikke rett til å lage hundre kopier av boka og selge de fra nærmeste gatahjørne. Dette er ikke bare ulovlig men også svært vanskelig. Man gis heller ingen rett til å forandre manuskriptet til boka, noe som jo er umulig uten å ha tilgang til originalverket. Man ser at det man gis rett til å gjøre med boka kontra det som ikke er lov i stor grad svarer til hva det er bekvemt å gjøre med en bok. Rettigheter som tildeles i egneskap av at de kan eksekveres med bekvemmelighet gitt et spesifikt medium er det som kalles implisitte rettigheter.

Det finnes lovgivining som regulerer bruken av rettighetsbelagt materiale, både på nasjonalt plan og internasjonalt gjennom internasjonale avtaleverk, men det har vist seg meget vanskelig å håndheve dette lovverket. Innehavere av rettigheter over et materiale har hatt få muligheter for å kontrollere om bruken av materialet utøves i overensstemmelse av de rettigheter som er knyttet til materialet etter at materialet er overlevert brukeren. Det har manglet systemer for å spore bruken av materiale etter at det har vært stilt til disposisjon for brukeren. Rettighetshavere har stort sett måttet stole på at de som overlates materialet har overholdt begrensninger i rettigheter over materialet i henhold til gitte retningslinjer og gjeldende lovverk, og at implisitte rettigheter har forhindret uønsket bruk av materialet. Bare i relativt ekstreme tilfeller har rettsvesenet vært koblet inn for å slå ned på brudd på tillatelser gitt til rettighetsbeskyttet materiale.

I mangel på systemer for å ivareta rettigheter i forhold til et materiale, har materialet tradisjonelt i en viss grad vært beskyttet i egenskap av å være tilgjengelig i fysiske eksemplarer (jfr implisitte rettigheter). For å kunne for eksempel lese, kopiere eller distribuere fra rettighetsbelagt materiale har man måttet være i besittelse av eller ha tilgang til en fysisk kopi av materialet. Til en viss grad har man kunnet styre og begrense hvem som har hatt tilgang til materialet gjennom utlånsregler i biblioteker og valg av distribusjonskanaler (for eksempel åpent salg kontra utsendelse til utvalgte bekjente). En styrt distribusjon kommer dog ofte i konflikt med ønsket å nå ut til et størst mulig publikum.

Iboende egneskaper ved fysiske eller analoge media (lydbånd, papir, mikrofilm osv) har også begrenset mulighetene til bruk av materialet som ikke er i henhold til de rettigheter som er knyttet til materialet (Jfr. implisitte rettigheter over). For eksempel har kopiering fra bøker alltid medført en forringelse av kvaliteten av kopien. Slik har mulighetene til å lage nye instanser/kopier av materialet som igjen kan brukes i uoverensstemmelse med rettigheter knyttet til det originale materialet blitt begrenset.

Både fysisk realisasjon, styrt distribusjon og iboende egenskaper ved materialet er en beskyttelse av rettigheter 'by accident'. Foruten lover og regler har det ikke eksistert effektive systemer for å kontrollere at materiale blir konsumert og brukt i henhold til de rettigheter som er knyttet til materialet etter at det er overlatt brukere av materialet.

6.4 Opphavsrett og Internet

"Copyright is dead."
(John Perry Barlow. Electronic Freedom Foundation)

Digital teknologi har ikke rettet opp svakhetene i tradisjonell rettighetshåndhevelse. Tvert imot har mulighetene til å kopiere, distribuere og bruke rettighetsbelagt materiale på nye måter blitt nærmest ubegrensede. Kopier av digitalt materiale kan lages praktisk talt uten kvalitetstap, og nettverksteknologien har gjort at man fra sin egen skrivepult kan publisere materiale, aksessere materiale som befinner seg på helt andre lokasjoner enn sin egen, og bruke andres materiale nærmest ubegrenset. Denne utviklinga har ført til at antall brudd på retningslinjer og lovverk har steget enormt. Fysisk distribusjon og iboende egenskaper ved mediene som er bærere av rettighetsbelagt materiale, som tidligere bidro til å redusere misbruket av rettighetsbeskyttet materiale, er nøytralisert av den digitale teknologien. Man kan si at "The Internet has made … implisit rights explisit" **[RosenblattTrippeMooney]**. Dette har ført til at mange rettighetshavere er nølende til å publisere sine verk i nettverksbaserte medier.

Ironisk nok er det nettopp digital teknologi og nettverksteknologi som kan gi oss mulighetene til å for første gang lage systemer som kan *håndheve* retningslinjer og lovverk knyttet til rettighetsbelagt materiale. Slike systemer kan for eksempel spore bruken av materialet som overleveres brukeren, håndtering av materialet, videre distribusjon av materialet og gjenbruk av materialet.

Utviklingen av slike systemer har gått tregt sett i forhold til de enorme mulighetene for distribusjon og salg som ligger innbakt i å ha en distribusjonskanal som internett. Grunnen til det kan muligens være at mye av tankegangen og drivkraften bak utviklingen av internett har vært nettopp mulighetene for enkel, uhindret aksess av informasjon og muligheten for alle til å publisere informasjon. ”Information wants to be free” har blitt et slagord for mange, og spesielt kanskje blant medlemmene i World Wide Web Consortium **[w3c]** som er ansvarlig for mange av de viktigste Internet-standarder i bruk i dag. Mange av disse er vel kanskje også redd for at store multinasjonale selskaper skal få en dominerende innflytelse på Internets framtidige utforming og at den ”anarkistiske samfunnsorganisering” som har lokket mange inn i Internet skal bli skadelidende.

Kvalitetssikring av ressurser og rettighetshåndhevelse har ikke vært i fokus i for utviklingen av Internet, men det ser ut til at utviklingen av Internet har kommet til et stadium der det er naturlig å begynne å se på disse aspektene ved informasjonsflyten over digitale nettverk. Dette må da kunne komme som et tillegg til de egenskapene ved Internet som har gjort det til et så overlegent medium, ellers vil kanskje Internet som kommunikasjons- og distribusjonskanal miste mye av sin tiltrekningskraft. De fleste som bygger systemer for rettighetsstyring i dag bygger da også slike systemer på toppen av eksisterende nettverksteknologi.

En annen grunn til at utviklingen av systemer for håndhevelse av opphavsrett og andre rettigheter på Internet har gått tregt er at Internet i sin natur er transnasjonalt og at ingen enkelt stats lovgivning derfor gjelder for Internet som helhet. Den gjeldende oppfatning er vel etter hvert at ”loven” gjelder på Internet, men det er mindre klart hvilken lov vi snakker om. Det i det siste har blitt undertegnet noen avtaler og konvensjoner som går i retning av en sterkere beskyttelse av intellektuelt arbeid i Internetsammenheng, spesielt i regi av WIPO **[WIPO]**. Selv om Bernkonvensjonen ikke var laget med Internet i tankene har det vist seg at også mange av bestemmelsene der kan føres direkte over på mye av den publikasjon som foregår i Internetsammenheng.

Det er et åpent spørsmål om man klarer å lage konvensjoner som garanterer beskyttelse av opphavsretten for digitalt materiale. Bernkonvensjonen bygger på at publikasjon av materiale stort sett foregår innenfor enkeltland. Ved å fastsette internasjonale retningslinjer som ratifiseres i enkeltland kan man definere en rettighetsbeskyttelse for materiale i de land som har ratifisert de internasjonale avtaler. Ved kun å gjøre materiale tilgjengelige i disse landene har man laget en rettighetsbeskyttelse av materiale som er basert på styrt distribusjon. Problemet er at publikasjon i Internet i sin natur er transnasjonal slik at publisering kan foregå fra servere og datamaskiner som er plassert i land som ikke har ratifisert et avtaleverk om opphavsrett og likevel være fullt ut tilgjengelig i alle land og deler av verden. Dette ser vi allerede eksempler på når servere som distribuerer stjålet software osv plasseres på utrangerte oljeinstallasjoner i internasjonalt farvann eller i land som har svak eller ingen lovgivning som sikrer opphavsrett og lignende.

I den analoge verden er det slik at håndhevelsen av rettighetslovgivning, både den som er fastsatt nasjonalt og den som er fastsatt gjennom internasjonalt avtaleverk er overlatt til nasjonalt politi og rettsvesen. Brudd på opphavsretten må rapporteres, etterforskes og sanksjoneres overfor i det land det oppstod i. Det er dessverre slik at mange land rundt omkring i verden ikke har kapasitet til eller tradisjon for å håndheve slik lovgivning. Et alternativ hadde kanskje vært å opprette et slags overnasjonalt ”Internet-politi og ”Internet-rettsvesen” men det er lite sannsynlig at suverene stater ville være villige til å overlate suverenitet til slike instanser.

Det har også vist seg at implementasjonen av overnasjonale avtaler i de enkelte lands lovgivning kan gi uante konsekvenser. I 1998 undertegnet daværende president i USA Bill Clinton en lov som kalles ”The Digital Millennium Copyright Act” **[DMCA]** som skulle beskytte rettighetsbelagt materiale. Section 1201 skulle implementere de nylig undertegnede WIPO avtaler som USA hadde forpliktet seg til å ratifisere. Dessverre gikk loven ut over det som var avtalt i det internasjonale avtaleverket, og har ført til en mulighet for industrien til å bruke loven til konkurransevridning og å nøytralisere viktige prinsipper som ”fair use” og personlovgivning og forskning. Problemet er ikke begrenset til USA. På grunn av Internets transnasjonale natur har loven ført til at mange datakonferanser må avholdes utenfor USA, ikke-amerikanske borgere nektes innreise til USA og både borgere av USA og andre land er etterlyst for brudd på loven som ellers skulle ha falt inn under fair use. **[DMCA_Cons]**.

6.5 Selge rettigheter eller materiale?

Rettigheter over et materiale kan altså overføres mellom juridiske personer. I et analogt miljø har man i praksis også vært nødt til å overføre en fysisk kopi av materialet for at den som får rettigheter over et verk for faktisk skulle kunne utøve rettigheten. Hvis for eksempel man skal gi bort retten til å lese en bok vil det være nødvendig å stille en kopi av boka til rådighet for den man gir denne retten til. I et analogt miljø kan man altså si at man selger et eksemplar av et verk og angir hva kjøperen kan gjøre med verket. Men vi har ingen garanti for at det brukeren velger å gjøre med materialet er det vi vil han skal kunne gjøre med verket. Digital teknologi gir oss muligheten til å ta utgangspunkt i rettigheter og handle med disse uavhengig av ressursen de gjelder. Brukere kjøper da en rettighet over et materiale og materialet stilles til rådighet for utøvelse av denne rettigheten. Forskjellen er at man i tilfellet der man handlet med bøker, begrenser man tilgangen til eksemplarer, mens alle muligheter for rettighetsutøvelse er gitt (selv om noen er vanskeliger å utøve enn andre), mens i det andre tilfelle begrenser man tilgangen til rettigheter. Forskjellen på disse to kan illustreres som følgende:

Gitt at Per har skrevet en bok om livet sitt. Han vil selge boka og kopierer opp fire eksemplarer. Boka planlegger han å selge for 100 kroner pr eksemplar. Per har ingen aning om hva kjøperne vil gjøre med bøkene etter at de har kjøpt de. Per vil altså ha solgt 4 eksemplarer av boka si, men vil ikke ha avgrenset skikkelig hva kjøperne kan gjøre med boka etter at de har kjøpt den. Men så finner Per på noe lurt. Han vil ikke selge boka i det hele tatt, men heller selge kun retten til å lese i boka. Han setter seg ned på Torget

og lar folk som er interesserte lese i boka som han holder opp foran de mot at de betaler 20 kroner. Nå selger Per rettigheter til materialet men gir ikke fra seg eksemplaret, og fungerer faktisk som et rettighetsstyringssystem

Å selge rettigheter over et materiale uten å selge materialet selv er et av de viktigste konseptene man forstå når man skal bygge systemer for rettighetsstyring for digitalt materiale i fremtiden.

7 Digital Rights Management

7.1 Hva er Digital Rights Management?

Tradisjonelt har økonomiske rettigheter over et verk blitt handlet med gjennom en lang rekke av avtaler fra opphavsmannen til sluttbrukeren av et materiale. DRM systemer prøver å bruke digital teknologi for å automatisere denne prosessen og forbedre den ved å tilby metoder og rutiner for å hindre at avtalene som utgjør verdikjeden mellom opphavsmann og sluttbruker ikke kompromitteres gjennom brudd på opphavsretter osv.

Termen ”Digital Rights Management” kan tolkes på to måter. Enkelte tolker DRM som ”digital rettighetshåndhevelse” mens andre tolker DRM som ”håndhevelse av digitale rettigheter”. Forskjellen er om det er rettighets*håndhevelsen* som er digital eller om det er *rettighetene selv* som er digitale.

Dr. Renato Ianella presiserer at ”..DRM is the ’digital management of rights’ and not the ’management of digital rights’” **[Ianella2001]**. Dette understøttes også av **[RosenblattTrippeMooney]** som sier at ”Digital rights management refers to *controlling* and *managing* rights to intellektual property.”

Det er altså ikke rettighetene til et materiale i seg selv som er digitale, men behandlingen av dem. Dette er viktig betraktning fordi den impliserer at det ikke finnes digitale rettigheter som i sin natur fraviker fra de rettigheter som finnes i den analoge verden. Derimot dreier digital rettighetshåndtering seg om å behandle de samme rettigheter som eksisterer i analoge sammenhenger i en ny kontekst og med nye midler.
Med utviklingen av digitale medier og digitale formater har måten vi kan distribuere og ikke minst konsumere materiale på blitt revolusjonert. Mange konsumeringsmåter som ikke er aktuelle i et analogt miljø er muliggjort av et digitalt medium. Dette er en styrke, men det innebærer også at rettighetshåndtering i et digitalt miljø må ha et bredere fokus enn rettighetshåndtering i et analogt miljø. På grunn av mulighetene for konsumering av materiale i den digitale verden er utvidet trenger vi nye metoder for å håndheve rettighetene på som er tilpasset dette nye mediet. Til dette trenger vi adgangskontrollsystemer og rettighetskontrollsystemer som kan styre hvem som har tilgang til materialet og hvilke rettigheter disse personene skal ha over et materiale. I følge tankegangen bak Open Digital Rights Language **[ODRL 1.0 Spec]** kan slike rettigheter deles i fire grupperinger:

Bruk: Hvilke rettigheter skal en bruker ha til å bruke materialet slik det er? Slik bruk kan innebære retten til å lese materialet, skrive ut materialet, spille av materialet og eksekvere materialet.

Gjenbruk: Hvilke rettigheter skal en bruker ha til å gjenbruke hele eller deler av materialet? Slik gjenbruk kan innebære å modifisere, lage utdrag fra, kommentere og samle opp materialet.

Overføring: Hvilke rettigheter skal en bruker ha til å gjøre materialet tilgjengelig for andre? Slik overføring av materialet kan innebære å selge, låne bort, gi bort, eller lease bort materialet.

Håndtering: Hvilken rettighet skal brukeren ha til å håndtere materialet? Slik håndtering kan innehære å flytte materialet, duplisere materialet, ta sikkerhetskopier av materialet, installere materialet, slette materialet, verifisere materialet, gjenopprette materialet, avinstallere materialet og lagre materialet.

DRM (Digital Rights Management) er en samlebetegnelse for systemer og teknologier som brukes til å styre distribusjonen av og beskytte rettigheter som er knyttet til et materiale. Systemer som ivaretar slike rettigheter finnes i mange forskjelliger versjoner og falsetter alt etter hvilke hensikter de tjener og hvilke forretningsmodeller de støtter (Mer om forretningsmodeller i kapittel 6.3). Derfor finnes det også mange definisjoner på hva Digital Rights Management er. Jeg vil bare nevne noen få her.

I spesifikasjonen til rettighetsspråket XrML finner vi denne definisjonen:

"The reality of the Internet and the need to contol the use of digital content and digital services has fueled the development of technologies that attempt to manage, secure, control, and automate the flow of content and the access of services. Digital Rights Management (DRM) is the common term associated with such tecnologies"
[XrML 2.0 Spec]

I innledningen til spesifikasjonen til rettighetsspråket Open Digital Rights Language finner vi denne definisjonen av DRM:

"Digital Rights Management (DRM) involves the description, layering, analysis, valuation, trading and monitoring of the rights over an entreprise's tangible and intangible assets" **[ODRL 1.0 Spec]**

Renato Ianella, som også er hovedarkitekten bak Open Digital Rights Language, sier at:

Digital Rights Management Systems *"..covers the description, identification, trading, protection, monitoring and tracking of all forms of rights usages over both tangible and intangible assets including management of rights holders relationships."* **[Ianella2001]**

Ut fra disse definisjonene kan man si at DRM systemer sikrer og distribuerer digitalt materiale i en form som beskytter og håndhever rettighetene til dette materialet gjennom bruk, gjenbruk, håndtering og videre distribusjon samtidig som de holder rede på rettighetsholdere og deres rettigheter gjennom slik distribussjon.

7.2 Når er bruken av DRM Systemer relevant?

Som sagt i avsnittet over har økonomiske rettigheter over et verk blitt handlet med gjennom en lang rekke av avtaler fra opphavsmannen til sluttbrukeren av et materiale. Dette gjøres fortsatt selv om prosessen ofte kan automatiseres vha DRM systemer.

Innholdet i mange slike avtaler som lages høyt oppe i verdikjeden vil være svært økonomisk viktig for partene i en potensiell avtale, og det vil derfor ikke være sannsynlig at de vil være villige til å overlate framforhandlingene av disse avtalene til et automatisert system. Fordi slike viktige avtaler også vil være resultat av lange forhandlinger mellom partene vil de også ofte inneholde svært spesialiserte klausuler som det vil være vanskelig eller kanskje umulig for et automatisert system å holde oversikt over eller håndheve.

Likevel vil det kanskje være naturlig å la DRM systemer ta seg av rettighetshåndteringen for materialet på et lavere nivå i verdikjeden, selv for verk der avtaler på et høyere nivå i verdikjeden ble inngått manuelt. På slike lavere nivå vil ofte mange av de spesialiserte klausulene falle bort og pga av at de enkelte transaksjonene ikke er like økonomisk viktig for partene vil de være mer tilbøyelig til å overlate rettighetsbehandlingen til automatiserte systemer, spesielt fordi avtalene som må inngås kan bli svært mange. For verk som ikke er like viktige økonomisk vil det kanksje være mest naturlig å la et DRM system ta seg av all rettighetshåndtering i forbindelse med verket i alle ledd fra opphavsmann til sluttbruker.

Men det finnes kanskje også tilfeller der det vil oppleves som for tungvint eller kostbart å laste verk og deres rettighetsbeskrivelser inn i et DRM system, slik at man heller velger å ikke bruke slike systemer for å håndheve rettigheter i forbindelse med slike verk. I mange slike tilfeller vil man heller enten bare legge verkene ut på en åpen webside eller kanskje ikke publisere materialet i det hele tatt. Det er likevel viktig å merke seg at det mest sannsynlig vil være flere verk som med økonomisk gevinst kan gjøres tilgjengelig for offentligheten gjennom automatiserte publiseringssystemer enn det som ellers ville blitt offentliggjort gjennom tradisjonelle media.

7.3 Forretningsmodeller

Når man skal velge en DRM arkitektur eller løsning er det viktig å sette dette i sammenheng med hva man skal bruke systemet til. De fleste som publiserer rettighetsbasert materiale over Internet i dag gjør dette i henhold til et eller flere

paradigmer eller forretningsmodeller. **[RosenblattTrippeMooney]** identifiserer 6 mulige forretningsmodeller som man kan bygge DRM systemer rundt**:**

Paid Downloads betyr at man betaler for en kopi av et materiale som lastes ned i kryptert form til en spesiell klient som dekrypterer innholdet og viser fram innholdet til brukeren. Man betaler da for å få tilgang til nøklene som skal til for å "låse opp"/dekryptere materialet.

Subscription betyr at man betaler for tilgang til et "members only" område på et nettsted og har fri og ubegrenset adgang til ressurser på medlemsområdet i en gitt periode.

Pay-per-view og pay-per-listen betyr at man betaler for antall ganger man aksesserer en ressurs. I motsetning til paid downloads er dette en forretningsmodell som egner seg best for "streaming media" altså en kontinuerlig strøm av lyd og eller bilde fra en server til en klient uten at nedlastingen fører til at det lages en kopi av materialet på klienten.

Usage metering betyr at brukeren registrerer seg på et nettsted som har ressurser som brukeren ønsker tilgang til. Nettstedet holder styr på hvor mange ganger brukeren aksesserer en ressurs og /eller hvor lenge brukeren har adgang til ressursen. For gitte tidsintervaller mottar brukeren en avregning som er i henhold til akkumulert aksesstid eller akkumulert antall forespørsler multiplisert med prisen for en aksess eller for adgang i et gitt tidsintervall.

Peer-to-peer og superdistribution. Superdistribusjon er en fler-trinns distribusjon der en ressurs blir levert videre i flere trinn. Et eksempel på slik superdistribusjon kan være når en forfatter selger rettigheter til et manuskript til en forlegger som selger rettigheter til en bokhandler som selger rettigheter til en leser. Dette kalles "multitiered superdistribution". Men vanligere i Internetsammenheng er peer-to-peer distribution. Peer-to-peer distribusjon er ikke lagdelt slik som multitiered superdistribution. Et eksempel kan være at en som laster ned en fil også gis retten til å gi eller selge videre fila til en eller mange tredjepersoner under et gitt sett av betingelser og begrensninger. Denne tredjepersonen kan så selge fila videre til en fjerde person osv. Det finnes mange variasjoner over peer-to-peer superdistribution og det vil ta for lang tid å dekke alle her.

Selling rights betyr i realiteten at man seller rettigheter til et materiale og distribuerer rettighetene over et materiale i stedet for selve materialet. Det er en tynn linje mellom dette og de andre forretningsmodellene, fordi det å selge en ressurs også betyr å selge visse rettigheter sammen med materialet. Typisk foregår slike transaksjoner mellom aktører på bedrifts- og organisasjonsnivå heller enn mellom sluttbrukere av et materiale, typisk når en del av et verk vil brukes i en ny bok eller en bok vil en forlegger ønsker å utgi en norsk oversettelse av en engelskspråklig bok.

Forståelsen av slike forretningsmodeller er viktig for å kunne lage et skikkelig DRM system. Et DRM system er ikke annet enn en implementasjon av en forretningsmodell på Internet.

7.4 Forretningsmodeller i Digitale Bibliotek

Tradisjonelle bibliotek har vært kjennetegnet ved at de låner ut sine ressurser til sine brukere mot at brukerne returnerer eksemplarene etter en avtalt tid. Dette er en forretningsmodell som er basert på at verk er fysisk distribuert i eksemplarer. Et bibliotek har kun ett eller en begrenset mengde eksemplarer av verket tilgjengelig for sine brukere, slik at hvis flere skal kunne ha glede av verket må brukerne tvinges til å levere tilbake verket til avtalt tid.

I et digitalt bibliotek derimot er et verk realisert som en digital representasjon lagret på en tjenermaskin. I stedet for å låne ut en begrenset mengde fysiske realisasjoner av verket som må leveres tilbake, kan man i et digitalt bibliotek fremstille kopier av originalressursen og gi til brukeren. Tilbakelevering er ikke nødvendig for at andre brukere skal kunne gis tilgang til verket fordi ”originalen” fortsatt befinner seg i det digitale bibliotek slik at nye kopier kan framstilles. Slik sett er lånemodellen en ikke nødvendig restriksjon på distribusjonen av materiale fra et digitalt bibliotek

Man kan likevel implementere en tradisjonell ”lånemodell” for et digitalt bibliotek ved å legge inn funksjoner som gjør kopier av materialet uleselig etter en hvis tid, og man kan sette begrensninger på hvor mange kopier det digitale biblioteket kan distribuere samtidig. Eksempelvis kan man si at hvis det digitale biblioteket bare skal kunne låne ut fire eksemplarer av et verk vil man kunne distribuere inntil fire kopier av verket. Femte bruker som forespør et eksemplar av verket vil bli nektet tilgang det så lenge de fire første distribuerte kopiene er leselige. Når en eller flere av de fire først distribuerte kopiene er blitt uleselige kan man distribuere nye kopier til nye brukere. Slik kan man implementere en en tradisjonell forretningsmodell for bibliotek i en digital kontekst.

Uansett om det digitale bibliotek kan distribuere et gitt eller ubegrenset antall kopier er dette en versjon av ”Paid Downloads” som ble presentrert i avsnittet om forretningsmodeller over. Forskjellen er at brukeren ikke får tilgang til ressursen til evig tid. Brukeren ”leverer tilbake” materialet når materialet ikke lenger er leselig. I realiteten får ikke en bruker eiendomsretten over en kopi av materialet, men en lisens som tillater brukeren tilgang til ressursen i en gitt periode. I tillegg til tidsbegrensningen kan man legge inn begrensninger på *hvem* som får ”låne” et eksemplar og *hva* disse får lov til å gjøre med et materiale. Implementasjonen av en slik forretningsmodell forutsetter DRM systemer.

7.5 Rettighetsmodeller

En rettighetsmodell defineres i boka ”Digital Rights Management Business and Technology **[RosenblattTrippeMooney2002]** som *”.. a spesification of the types of rights that the system can keep track of and what the system can do to or with those rights.”* Rettighetsmodeller beskriver typer av rettigheter og attributer til disse rettighetene som et gitt system, eller i noen tilfeller alle mulige typer systemer, skal eller bør kunne behandle.

Rettighetsmodeller er ikke spesifikt for digitale rettigheter. De finnes også for analoge medier som bøker, filmer osv. Men behovet for å uttrykke rettighetsmodeller i den analoge verden har ikke vært så stort som det er i den digitale verden fordi transaksjoner mellom rettighetshavere ofte har vært behandlet på individuelt grunnlag gjennom skriftelige kontrakter. Transaksjoner på lavere nivå har også gjerne vært regulert av implisitt rettigheter (se kapittel 5.3) som til en viss grad gjør kravet til å uttrykke rettighetsmodeller mindre. Hvis man ser for seg at man går i bokhandelen og kjøper en bok er stort sett rettighetsmodellen for dette gitt av implisitte rettigheter. Når man kjøper en bok får man retten til å lese boka så mange ganger man vil og når man vil. Man får også retten til å gi bort, brenne opp eller låne bort sitt eksemplar. Dette er rettigheter som ligger innebygd i formatet til boka. Prisen for å få disse rettighetene er at man betaler hva boka koster. I digitale medier legger ikke formatet begrensninger på hva man kan gjøre med ressursen, og derfor må dette uttrykkes eksplisitt slik at man kan lage systemer som kan beskytte disse rettighetene. For å kunne uttrykke slike rettigheter må vi lage en modell over de mulige rettigheter som finnes.

Den første kartlegginga av digitale rettigheter og deres egenskaper finner vi i artikkelen ”Letting Loose the Light: Igniting Commerce in Electronic Publication” av Dr Mark Stefik **[Stefik1]** fra Xerox PARC research labs. Jeg vil gjengi i grove trekk rettighetsmodellen fra denne boka for å illustrere hva en rettighetsmodell er, både fordi den er enkel og intuitiv, fordi den fortsatt er gyldig som referansemodell og fordi den danner mye av den itellektuelle bakgrunnen for rettighetsspråket eXtensible rights Markup Language **[XrML]** som Dr. Stefik og hans kolleger står bak og ”Open Digital Rights Language” **[ODRL]**, et annet språk for å uttrykke digitale rettigheter.
I **[Stefik1]** delte Stefik alle typer digitale rettigheter inn i tre kategorier som han kalte ”Render Rights”, ”Transport Rights” og ”Derivative Work Rights”.

”Render Rights” er retten til å gjengi et materiale eller presentere det i et eller annet medium slik at brukeren får tilgang til innholdet i materialet. ”Render Rights” kan deles inn i ”Print” som betyr å lage en permanent kopi i et fysisk medium, ”View” som betyr å vise fram på et dynamisk display som for eksempel en PC-skjerm, og ”Play” som betyr å gjengi i sekvens fra begynnelse til slutt slik at når man kommer til slutten av avspillingen er gjengivelsen over.

”Transport Rights” er rettigheter til å flytte eller kopiere innhold fra en plass til en annen. ”Transport Rights” kan deles inn i ”Copy”, ”Move” og ”Loan”. Man kan illustrere forskjellen mellom de ved å tenke seg at en bruker (Hans) i utgangspunktet har adgang til en ressurs og har tre måter han kan overføre denne adgangen til en bruker nummer 2 (Grete). ”Copy” betyr at Hans lager en kopi av materialet og gjør den tilgjengelig for Grete. Både Hans og Grete har nå tilgang til materialet. ”Move” betyr at Hans gir opp sin tilgang til materialet og overfører det til Grete slik at hun har tilgang. ”Loan” betyr at Hans midlertidig overfører materialet til Grete. Grete får tilgang til materialet, men må gi tilgangen tilbake til Hans etter en gitt periode.

”Derivative Work Rights” er retten til å manipulere med materialet for å lage nye avleda (derivative) materialer. Dette kan deles inn ”Extract Rights” som er retten til å trekke ut deler av materialet og bruke disse delene uavhengig av resten av materialet, ”Edit Rights” som er retten til å forandre på innholdet i materialet, og ”Embed Rights” som er retten til å ta en bit av materialet og inkludere denne i et annet materiale.

Selv om det er finnes andre rettigheter som ikke er nevnt her vil de fleste falle direkte inn under en av de tre grupperingene. Unntaket er rettigheter som kan kalles ”Utility Rights” som eksisterer ut fra et praktisk teknisk behov, men som ikke er viktig for å forklare en konseptuell modell over rettigheter. Slike rettigheter kan være retten til å lage temporære kopier for å gi raskere adgang til data i databaser eller nettverk, retten til å lage backup-kopier som følge av bruk av back-up rutiner på filservere og retten til å sette inn checksums, feilopprettingsdata og lignende for å sikre at data fra den originale ressursen ikke går tapt eller ødelegges.

For alle disse rettighetene kan det settes opp ”Considerations” som er attributter til en rettighet som angir hva brukeren må gjøre for å få tilgang til den gitte rettigheten. Dette kan være å betale en gitt sum, å registere sin epost adresse hos utgiveren av materialet, å gi tillatelse til at ens bruk av ressursen overvåkes av en sentral server eller andre forpliktelser.

For alle rettigheter kan det også settes opp attributter for ”Extent”som angir hvor ofte, hvor mange ganger, hvor lenge eller i hvilke kontekster en rettighet er gyldig, og det kan settes opp attributter for ”Types Of Users” som gir mulighet for å spesifisere forskjellige sett av rettigheter og attributter for forskjellige brukere eller grupper av brukere.

Man kan si at Dr. Stefiks rettighetsmodell er tredimensjonal fordi den for hver rettighet (eller også gruppe av rettigheter) kan angi attributter i tre dimensjoner. Man kan sette opp attributter for utstrekning en rettighet er gyldig for, hvilke brukere som kan eksekvere rettigheten innenfor denne utstrekningen og hvilke forutsetninger som må være oppfylt for at disse brukerne skal kunne eksekvere rettigheten innenfor utstrekningen

Ved hjelp av en slik modell kan man uttrykke alle rettigheter som skal kunne forstås av et system, og dette vil danne grunnlaget for implementasjonen av et Digital Rights Management (DRM) System.

7.6 Uttrykking av rettighetsmodeller

Rettighetene til et materiale må både identifiseres og beskrives/kodifiseres. Identifisering betyr å danne rettighetsmodeller som støtter en gitt forretningsmodell som skal fungere i en gitt kontekst. Kodifisering betyr å formalisere en rettighetsmodell i et språk eller sett av rutiner som gjør rettighetsmodellen forståelig for maskiner som skal håndheve rettighetene i en rettighetsmodell.

Kodifisering av rettighetsmodeller kan essensielt gjøres på to måter.

Den første er å lage programvarerutiner som innkorporerer en rettighetsmodell. Dette er en ikke anbefalt metode, da man lett kan komme til å blande rettighetsmodell-logikk med annen programvarelogikk. Dessuten blir også en slik løsning veldig raskt statisk med en relativt begrenset uttrykkskraft. Den andre er å lage rettighetsspråk som kan uttrykke rettighetsmodeller og som kan fungere som input til et programvaresystem som håndhever rettigheter definert i rettighetsspråket. Dette gir større mulighet til å skille mellom programlogikk og rettighetslogikk som er del av en rettighetsmodell, og gir mulighet for mer fleksibel progamvarelogikk. Iannella **[Iannella2001]** skiller mellom en funksjonell arkitektur som håndhever rettighetsproblematikken, og en informasjonsarkitektur som kodifiserer rettighetsproblematikken i et formelt språk. Den funksjonelle arkitekturen vil da bruke informasjonsarkitekturen som grunnlag og inndata i utøvelsen av rettighetsstyringen i forhold til et gitt materiale.

Det finnes flere formelle språk som har mulighet til å uttrykke rettigheter. De fleste ser ut til å bestå av et rammeverk som består av en ”Data Dictionary” og en syntaks. ”Data Dictionaries” definerer hvilke termer og begreper som brukes i et gitt rettighetsspråk og hva som er den semantiske betydningene av disse. Syntaks er en deklarasjon om hvordan slike termer og begreper kan kombineres for å danne kvalifiserte uttrykk om rettigheter som programvaresystemer/funksjonelle arkitekturer kan forstå og handle på grunnlag av.

Definisjoner av rettigheter over et materiale blir ofte veldig raskt kompliserte, spesielt når de skal støtte forretningsmodeller som tillater superdistribusjon, gjenbruk av materiale fra et verk i nye derivative verk og skal kunne uttrykke unntak i rettigheter og begrensninger av rettigheter som er basert på det som i USA er kalt ”fair use”, og som er nedfelt i kapittel 2 i åndsverksloven i Norge.

7.7 Hvilke muligheter åpner seg gjennom DRM?

Flere og flere som publiserer rettighetsbelagt materiale blir klare over at det i de nye forretningsmodellene som er definert over åpner seg nye muligheter for økt fortjeneste gjennom finere granulert distribusjon (muligheten til for eksempel å selge en enkelt artikkel i stedet for et årsabonnement i et tidsskrift), sekundær publikasjon (for eksempel å øke verdien av ressurser ved å tilby de til andre publisister som kan pakke de inn i større samlinger), og ikke minst gjennom å finne ut mer om sine kunder. DRM systemer kan finne fram til kjøpemønstre for sine kunder, bruksmønstre for en gitt ressurs, om kunden gir videre en ressurs til tredjeperson, nedlastingsfrekvens og mange andre opplysninger som man ikke ville hatt mulighet til å kontrollere i et fysisk distribusjonsparadigme. ”.. sometimes the information on content use is worth more than the content itself.” **[RosenblattTrippeMooney]**. Dette reiser mange prinsipielle spørsmål om hvilke opplysninger Internet aktører har lov til å samle inn om sine brukere, men det er et fagområde som er for stort til å ta med i denne oppgava.

I tillegg viser det seg at DRM systemer ofte støtter funksjonalitet som går ut over det som er anerkjent som beskyttelse av rettigheter. Det er ofte veldig vanskelig å definere hva som er beskyttelse av rettighetshaveres legitime krav til bekyttelse av rettigheter til

digitalt materiale og hva som er beskyttelse av kommersielle interesser forbundet med salg eller lisensiering av det samme materialet. I mange tilfeller vil store selskaper ha muligheter til å definere forretningslogikk inn i DRM systemer som ikke nødvendigvis er begrunnet i opphavsretten til det samme materialet. Dette kan selvfølgelig ses på både som en styrke og en fare ved DRM systemer alt ut fra ståsted.

DRM og elektronisk publisering gir også mulighet for småskala publisering av materiale som kan komme mange forfattere, musikere og kunstnere til gode. Ved at kostnadene ved å lansere for eksempel en cd vil være mye lavere ved Internetbasert distribusjon enn hvis det skulle trykkes opp masse eksemplarer for å sende rundt i verden skulle sjansen til å kunne bli publisert kunne være økt betraktelig.

7.8 Hva er svakhetene og farene ved bruk av DRM?

I første avsnitt av dette kapittelet fastslo vi at DRM omhandler håndhevelsen av rettighetene over et digitalt materiale. Det er håndhevelsen av rettigheter som er digital, ikke rettighetene selv. For mange vil dette være en lettelse, men det impliserer også en del problemer. Som tidligere nevnt er publisering i Internet i sin natur transnasjonal. Men opphavsretten er definert nasjonalt med variasjoner mellom diverse land, selv om den til en viss grad er harmonisert gjennom internasjonalt avtaleverk. Så det store spørsmålet er egentlig: Når vi lager systemer som skal håndheve rettigheter for transnasjonal distribusjon av rettighetsbelagt materiale, hvilke lands lovgivning skal vi normalisere systemet på? I mine øyne er dette et problem som er lite påaktet i utviklingen av DRM-systemer. Bakgrunnen for at dette problemet er så lite påaktet er kanskje at utviklingen av slike systemer stort sett foregår i USA og tar utgangspunkt i USAs lovgivning og de juridiske avveininger som blir tatt når det gjelder opphavsrett der.

Et annet problem er at slike systemer er lite egnet til å styre unntak fra opphavsretten som er begrunnet ut fra hensyn til samfunnsutvikling og sammfunnsopplysning. Dr. Stefik mener at dette ikke er et gyldig argument mot DRM systemer da dette bare er et spørsmål om å lage språk og systemer som er sofistikerte nok til å kunne definere unntak fra de generelle opphavsretter. For å lage slike språk er det en forutsetning at man kan definere presist hva som skal være unntatt opphavsrett. Dessverre er det veldig vanskelig, kanskje umulig, å lage en slik definisjon, da det i de fleste tilfeller vil være en vurderingssak hva som kvalifiserer for unntak fra opphavsretten, selv om det finnes en del retningslinjer nedfelt i åndsverkslovens kapittel 2. Dessuten vil vurderingene av hva som kan regnes som berettighet for unntak fra opphavsretten variere sterkt fra land til land og også utvikle seg over tid.

En annen ting er at DRM systemer kan underminere noen av kreftene som står bak den naturlige utviklingen av opphavsrettens unntak. En viktig pådriver i denne utviklingsprosessen vil være at noen mener at en gitt bruksmetode eller et gitt bruksområde for et materiale kvalifiserer til unntak fra de generelle opphavsregler. Hvis rettighetsholderen ikke er enig kommer saken opp ofte opp for en domstol som skal vurdere om dette er et brudd på opphavsretten eller om det kvalifiserer som unntak. Det

er i slike tilfeller at grensene for hva som kan unntas fra generelle opphavsrettighetsbestemelser gås opp. Hvis man lager systemer som begrenser eller eliminerer muligheten til å utfordre de regler som rettighetsholdere setter for bruk av sitt materiale tar man også bort en viktig utfordring til opphavsrettslovgivningen. **[FairUse&DRM]**

Dess mer kompliserte systemer vi lager for rettighetsstyring, dess mer brukerfientlig vil også systemene ofte bli, selv om mye av kompleksiteten i slike systemer kan skjules for brukerne. Generelt vil brukere ofte velge den raske og enkle måten å aksessere informasjon på, delvis uavhengig av pris og kvalitet. Brukere vil sky unna systemer som krever en ”omvei” for å få tilgang til materiale. I mange tilfeller viser det seg at det ikke er den økonomiske kompensasjonen som må betales som utgjør tungen på vektskålen når man velger informasjonskilder. Hvis det virker for tungvint å få tak i noe man vet er kvalitetskontrollert vil brukere ofte velge en alternativ, fri og enkel nedlasting som ikke krever at man for eksempel registrerer seg på et nettsted før man får tilgang til et materiale. Dette gjelder også for systemer som brukerne oppfatter som en begrensning på hva de kan gjøre med materialet. Men det siste er jo faktisk litt av vitsen med et DRM system.

Mange har også hevdet at tankegangen med at man skal kunne bruke DRM systemer for å automatisere avtaler om overføring av rettigheter ikke er mulig å omsette i praksis. Hele vitsen med forhandlinger er at man ut fra to ståsteder kommer fram til en middelvei som begge parter kan godta. I grunnen for alle forhandlinger ligger altså en tovegs kommunikasjon. Men i grunnen for DRM systemer ligger derimot har en slags ”take it or leave it” tankegang. Rettighetsholdere fremsetter et tilbud til brukere om hvilke rettigheter de kan få tilgang til når de legger materialet inn i et slikt system, men det er ikke mulig for brukere å påvirke dette tilbudet. Brukerne kan akseptere tilbudet, evt ta imot deler av tilbudet, men de kan ikke påvirke selve tilbudet til sin egen fordel. Slik DRM systemer fungerer i dag er det derfor ikke mulig å forhandle om en avtale i tradisjonell forstand. Kanskje kommer det etter hvert systemer som lar brukeren forhandle om rettighetsoverføring i tradisjonell forstand, men i dagens situasjon må man vel si at muligheten til å forhandle er relativt ”begrenset”, selv om mange nok ser på dette som en mer filosofisk betraktning.

7.9 Hvorfor er DRM viktig i digitale bibliotek?

Brukere ønsker i utgangspunktet å ha tilgang til mest mulig informasjon uten å måtte forplikte seg til vederlag i form av penger eller handlinger. Men de ønsker å gjøre dette på en måte som ivaretar deres behov for sikkerhet og uten at andre har innsyn i hva de foretar seg. Hva de leser og aksesserer er en privatsak så lenge det er innen juridiske rammer, og er en viktig del av et menneskes demokratiske rettigheter.

Rettighetshaverne skal ofte leve av å publisere materialet de produserer. I slike tilfeller er de avhengig av at materialet som publiseres bare aksesseres av brukere som betaler for å få tilgang til det. I mange tilfeller utgjør også materiale verdier som man ikke vil at andre

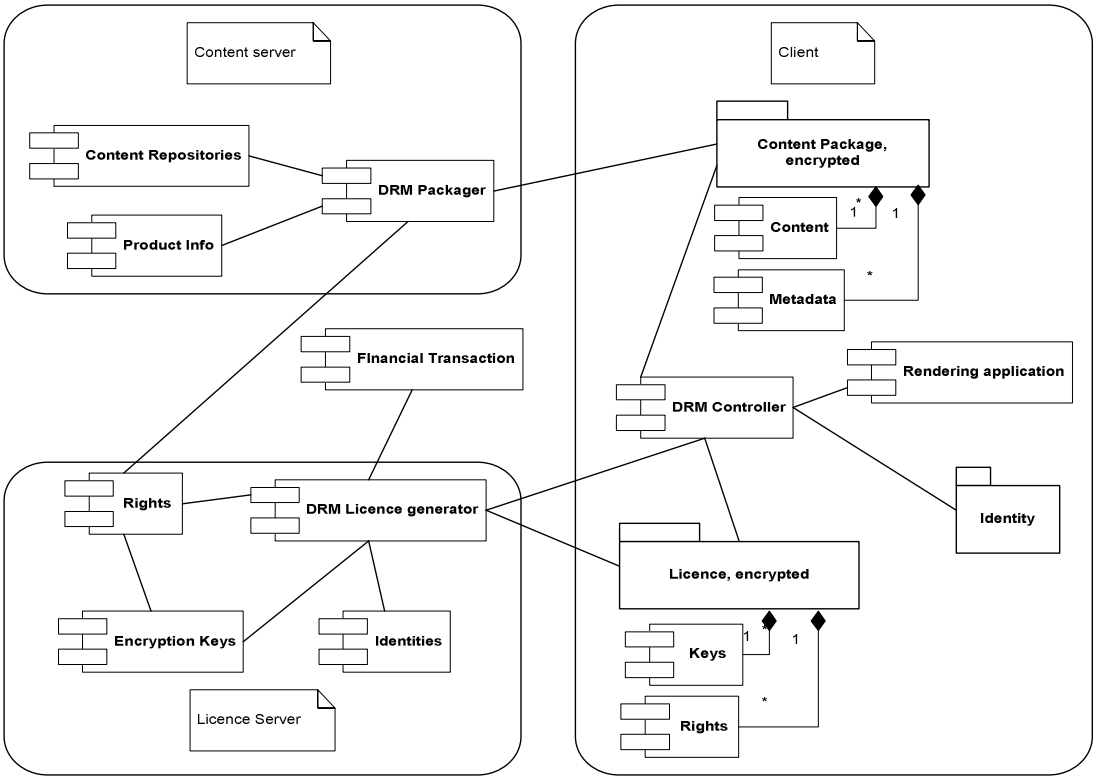
skal ha tilgang til i det hele tatt, eller som man bare vil at en utvalgt gruppe av brukere skal ha tilgang til, for eksempel brukere innen samme bedrift hvis det gjelder bedriftsinterne dokumenter osv. De er altså interessert i at tilgangen til deres informasjon styres til en gitt bruker etter kriterier de setter opp uten at andre brukere får tilgang til materialet, samtidig som de gjerne vil ha betalt for at andre skal lese deres materiale.

I bunn og grunn er digitale bibliotekers behov en kombinasjon av brukernes og rettighetshavernes behov. Det digitale bibliotek må i størst mulig grad tilfredsstille begge grupper for å kunne overleve. På den ene side vil liten adgang til distribuert materiale ikke være særlig attraktivt for brukeren da han ikke får den varen han er interessert i. På den annen side vil man ikke finne mennesker som er villige til å publisere materiale gjennom det digitale bibliotek dersom ikke betingelsene det gjøres under og systemet som håndhever betingelsene oppleves som adekvat av de som skal publisere materialet.

Det er derfor i det digitale biblioteks interesse å kreve kompensasjon fra brukeren når publisert materiale aksesseres, sørge for at rettighetshavere har et visst innsyn i hvordan deres materiale tas i bruk, forhindre uønsket bruk og misbruk av materiale i det digitale bibliotek, og å styre tilgangen til materialet til enkelte personer. Det er også viktig at det digitale bibliotek tar hensyn til personvernet til brukere og eiere av materiale. Godt gjennomtenkte og adekvate DRM systemer kan sørge for en god balanse mellom de forskjellige gruppers behov.

7.10 En referansearkitektur for DRM systemer

I boka ”Digital Rights Management Business and Technology” **[RosenblattTrippeMooney]** mener forfatterene å ha funnet at de fleste DRM systemer har en del fellestrekk. Ut fra disse fellestrekkene har de laget en referansemodell for DRM Systemer. Selv om ingen DRM systemer i dag implementerer referansemodellen, er likevel en slik referansemodell viktig for en konseptuell forståelse av DRM systemer generelt og kan brukes som referanse for å sammenligne forskjellige løsninger og å finne fram til nye løsninger for nye forretningsområder. Referansemodellen er gjengitt under.



Figur 7-1En referansearkitektur for DRM systemer

Referansemodellen består av tre deler: ”**Content Server**”, ”**License Server**” og ”**Client**”. I tillegg finnes en komponent som kalles ”financial transaction” som angir en betalingstjeneste. Da de fleste DRM systemer outsourcer slike tjenester gjennom samarbeid med etablerte betalingstjenester tilgjengelig fra for eksempel MasterCard eller VISA er dette holdt utenfor selve referansemodellen.

En ”**Content Server**” består av ”Content Repository”, ”DRM Packager” og ofte ”Produkt Info”. Selve materialet som skal distribueres og som brukeren søker adgang til befinner seg i et ”Content Repositor” i en form som er klargjort for distribusjon eller som muliggjør klargjøring for distribusjon idet materialet etterspørres. ”Content repositiory” er i praksis en database eller filsystem som i tillegg til selve materialet inneholder en del metadata om materialet som skal distribueres.

En ”**DRM Packager**” er en programvarepakke som klargjør et materiale for distribusjon. Den krypterer materialet som skal sendes til brukeren sammen med en rettighetsbeskrivelse for materialet. Distribusjonen inneholder også en identifikator for

materialet og ofte en del deskriptive metadata. En slik distribusjon kalles en ”Content Package” Nøklene for å dekryptere materialet lagres i ”Licence Server” som beskrives under. Rettighetsbeskrivelser kan være lagret sammen med materialet som skal distribueres i ”Content Repository” og må da sendes til ”Licence Server” av ”DRM Packager” eller kan være lagret uavhengig av materialet i ”License Server” i hvilket tilfelle ”DRM Packager” må hente rettighetsbeskrivelsene før den kan lage en ”Content Package”. ”Product info” er deskriptive metadata om ressursene som befinner seg i ”Content Repository” og kan brukes til å bygge søkesystemer og markedsføringssystemer.

En ”**Licence Server**” inneholder rettighetsbeskrivelser for et materiale angitt av en identifikator, informasjon om brukere som vil utøve rettigheter over et materiale, og nøkler som trengs for å dekryptere ”Content Packages” etter at en ”DRM Packager” har dannet slike ”Content Packages” En ”License Server” inneholder også en programvarepakke kalt en ”DRM Licence Generator” som invokeres av av en klient for å få tilgang til dekrypteringsnøkler osv fra ”Licence Server” Før vi beskriver denne programpakken videre tar vi en kikk på klienten i systemet.

”**Client**” er klienten som brukeren benytter for å få tilgang til materialet i en ”Content Package”. Klienten består av en ”Rendering Application” og en ”DRM Controller”. En ”Rendering Application” er et program som kan gi tilgang til å for eksempel vise fram, kopiere eller skrive ut materialet alt etter hva brukeren har lov til å gjøre med materialet. Hvilke muligheter en slik rendering application gir tilgang til bestemmes av en ”DRM Controller” som samarbeider med en ”licence Server” En ”DRM Controller mottar en brukers ønske om å utøve rettigheter over en content package, henter inn relevante opplysninger fra en bruker og basert på disse opplysningene henter den en lisens fra ”DRM License Generator” som angir hva brukeren kan gjøre med materialet og nøklene som skal til for å dekryptere ”Content Package”. Etter at materialet er dekryptert sendes det til ”Rendering Application” sammen med en beskrivelse av hvilke rettigheter brukeren har over materialet. Denne angivelsen av rettigheter bestemmer for eksempel hvilke menysystemer brukeren skal ha tilgang til eller om det skal være mulig å høyreklikke på materialet hvis brukeren sitter på en PC. Utøvelsen av rettigheter kan også rapporteres tilbake til ”DRM Licence Server” gjennom ”DRM Controller” etter hvert som de eksekveres.

8 Forutsetninger for Digital Rights management

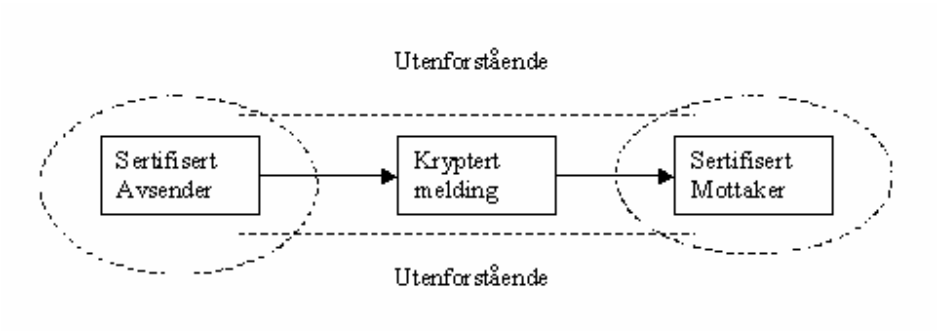
I dette kapittelet vil jeg ta for meg noen aspekter som er viktige forutsetninger for å kunne lage velfungerende digitale bibliotek som inkluderer Digital Rights Management, men som ikke egentlig er del av fagfeltet Digital Rettighetsstyring.

8.1 Sikker kommunikasjon og identifikasjon

"...Internet communication is much like anonymous postcards answered by anonymous recipients. However, these postcards, open for anyone to read—and even write in them—must carry messages between specific endpoints in a secure and private way."
[E.Gerck98]

IP-pakker som sendes over nettverk må ofte gå gjennom mange noder på veien fra avsender til mottaker. Alle disse nodene kan uten store anstrengelser aksessere innholdet i en pakke før den sendes videre, enten for å lete etter sensitiv informasjon eller for å sabotere kommunikasjonen. Enhver av disse nodene kan også settes opp til å svare på henvendelser over Internet på en slik måte at det vil være umulig for den andre part å skjønne at han ikke kommuniserer med den som var ment å være mottaker av informasjon, men en svindler som er ute etter å avsløre sensitiv informasjon.

For at digitale biblioteker som kommuniserer over Internett skal kunne fungere trenger vi metoder for å forsikre oss om at innholdet i kommunikasjonen mellom bibliotekssystemet og brukerne ikke kan snappes opp eller ødelegges på vegen mellom de to, og at deltakerne i en slik kommunikasjon er positivt identifisert. Det er vanlig å si at vi både må sikre avsender og mottaker og kommunikasjonskanalene mellom dem. Den mest vanlige løsningen på problemet er å bruke **kryptering** av meldinger (for å sikre "fortrolighet" og "integritet") og **sertifisering** (for å sikre at kommunikasjon foregår mellom de ønskede endepunkter i form av avsender og mottaker). Figuren under illustrerer beskyttelse av avsender og mottaker og kanalen mellom de.



Figur 8-1 Sikring av deltakere og kommunikasjonskanalen mellom dem

8.2 Postiv identifikasjon av deltakere

Et digitalt sertifikat er internets equivalent til et ID kort. På samme måte som vi fysisk gjerne viser fram et id-kort når vi skal identifisere oss for eksempel når vi leier en bil eller lignende finnes det digitale certifikater som kan brukes av servere og klienter for at de sikkert skal kunne vise at de er den de gir seg ut for å være på Internet. Potensielle deltakere i en sikker kommunikasjon som ønsker å garantere sin identitet til andre kan søke om et digitalt certifikat fra en Certificate Authority (CA) som utsteder

digitale sertifikater. Dette er organisasjoner som tar betalt for å verfisere hvem du er gjennom diverse foretaksregistre, personnr-identifikasjon, personlig konsultasjoner osv. Når en slik Certifiserings Autoritet føler seg trygg på din identitet utstedere den et kryptert digitalt certificat. Digitale certifikater baserer seg på RSA algoritmen **[RSA]** og er kryptert med utstederens (CA) private nøkkel slik at autentiteten til certifikatet kan verifiseres. Digitale certifikater inneholder søkerens offentlige nøkkel og en mengde annen identifiseringsinformasjon for søkeren for eks navn, certifikatets serienummer, gyldig-til-dato, utstederens offentlige nøkkel osv. Den private nøkkelen for certifikatet gis direkte til søkeren. Denne skal holdes hemmelig.
Digitale certifikater fungerer som et vedlegg til en elektronisk melding som verifiserer positivt hvem avsender er av meldingen er.
Når noen mottar en melding med et kryptert digitalt certifikat kan vedkommende forsøke å dekryptere certifikatet med den offentlige nøkkelen til organisasjonen som utstedte certifikatet. Hvis dette lykkes vet mottakeren av meldingen at certifikatet er autentisk fordi det bare kunne ha vært kryptert med utstederens (CA) private nøkkel og dermed at avsenderen er den han utgir seg for å være. Dessverre er det relativt få privatpersoner som har digitale certifikater.

8.3 Sikring av kommunikasjonskanaler

For kryptering av meldinger mellom en server og en klient er det vanlig å bruke Secure Socket Layer.(SSL) **[SSL]**. SSL er en protokoll for å kryptere meldinger som sendes over et åpent nettverk opprinnelig utviklet av Netscape som nå finnes i versjon 3**.** Selv om SSL er en standard utviklet av Netscape er det en såkalt åpen protokoll og er i dag de facto industristandard.
SSL tilbyr kryptert kommunikasjon mellom sockets plassert i maskinene til deltakerne i en toveis kommunikasjon basert på en offentlig og en privat nøkkel.
Krypteringsalgoritmen for SSL er RSA algoritmen. **[RSA]**
SSL inneholder også støtte for å utveksle krypterte digitale certifikater slik at klienter og servere kan positivt identifisere hverandre og forsikre seg om at den andre er den vedkommende utgir seg for å være. Se digitale certifikater over.
Dessuten kan en mottaker vha av SSL sjekke om en melding fra en avsender har kommet fram uforandret. Dette gjøres ved hjelp av checksums som settes inn i meldingene.

9 State of the art i DRM

For å kunne lage fungerende DRM systemer må man ha en måte å uttrykke de rettigheter som systemet skal handle på grunnlag av. Behandling av intellektuelle rettigheter (engelsk: Intellektual Property) i et maskinbasert miljø krever at rettighetene som skal prosesseres må uttrykkes i et språk som er klart, utvetydig, og er lesbart for maskiner. Det er flere måter å gjøre dette på, men de fleste standarder bruker XML som et metaspråk for å definere språk som kan uttrykke rettigheter over et materiale.

Digital Rights management er et veldig stort felt som det kan være vanskelig å avgrense. Ofte blander standarder som er i bruk i dag sammen språk for å uttrykke rettigheter med andre ting som for eksempel kommunikasjonsprotokoller og filoverføringsprotokoller. Dette er tilfellet med for eksempel standarden The Information and Content Exchange **[ICE]** som er en standard for å utveksle data mellom informasjonstilbydere som aviser og radio- og tvstasjoner. I denne redegjørelsen for state of the art vil jeg bare ta med standarder og språk som på som tar sikte på å uttrykke rettigheter på generell basis. Det finnes ingen standarder som er spesifikt rettet mot digitale bibliotek, men etter min mening vil slike generelle språk kunne tjene behovene for digitale bibliotek.
Dette vil på ingen måte være en uttømmende liste av språk som eksisterer idag, men vil presentere noen av de initiativer som jeg mener er mest relevante ut fra et digitalt biblioteks ståsted.

9.1 Rettighetsspråk

9.1.1 Open Digital Rights Language (ODRL)

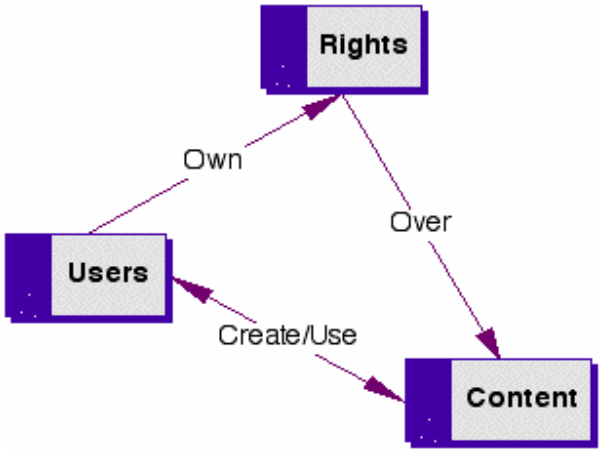
Open Digital Rights Managenent Language (ODRL) **[ODRL]** er et rettighetsspråk realisert i xml. Språket komplementerer eksisterende analoge standarder for rettighetshåndtering ved å tilby digitale ekvivalenter til disse systemene og ved å støtte et utvidbart sett av nye tjenester som er muliggjort av utviklingen av nettverksteknologi, programvare og standarder for behandling av data.

Open Digital Rights Language er i stor grad en videreføring av mye av tankegodset fra <indecs> **[INDECS]** prosjektet, og foreligger for tiden i versjon 2.0. Denne redegjørelsen av språket tar utgangspunkt i versjon 1.0 som som ble publisert 21. november 2001 av IPR Systems Pty Ltd.

IPR Systems Pty Ltd. har implementert et subsett av standarden selv og det samme har Nokia og Panasonic gjort. Standarden støttes også av mange tunge kommersielle og ikke-komersielle aktører. Dessuten er standarden sendt over til ISO/IEC MEGP standards body som respons på dennes etterlysning av forslag til ”..a Data Dictionary and Rights Expression Language for the MPEG 21 Framework”. Likevel mener **[RosenblattTrippeMooney]** at standarden ikke har fått så mye støtte utenfor Australia.

ODRL består av en modell og en data dictionary for språket som inneholder syntaksen/strukturen og semantikken for språket. Dette språket gir muligheten til å lage kvalifiserte uttrykk i xml som uttrykker rettigheter og overføring av rettigheter over både fysiske og digitale ressurser for behandling i et digitalt miljø. Språket uttrykker rettighetsmetadata i XML for alle typer materiale. Dette muliggjør kjøp og salg av rettigheter og kan brukes for å bestemme hvordan en bruker kan konsumere et innhold.

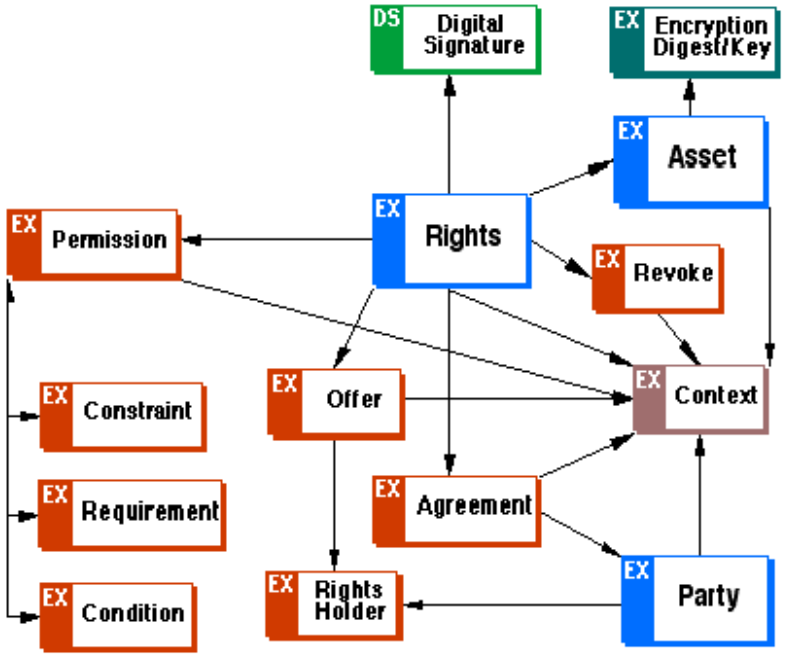
Grunntanken bak modellen er at problemdomenet identifiserer 3 kjerneentiteter og sammenhengen mellom dem. Dette er framstilt i modellen under. Figuren er hentet fra **[Ianella2001]**



Figur 9-1 Kjerneentiteter i tankegangen bak Open Digital Rights Management

Brukere kan være alle slags brukere fra de som er eiere av materialet via diverse rettighetsholdere til sluttbrukere. Innhold kan være enhver type innhold av enhver type på ethvert aggregeringsnivå. (jfr IFLA modellen) som er gitt en identifikator. Rettigheter er de uttrykte rettigheter som er gitt bruker til å råde over et gitt innhold. Dette kalles ofte IPR-Statements (IPR = Intellectual Property Rights)

Hele modellen som ligger til grunn for standarden kan illustreres som i figur 8-2 under. Her er semantikken gjort om i forhold til figur 8-1. Party er nogenlunde det samme som Users og Asset er det samme som content i den forrige figuren.



Figur 9-2 Konseptuell modell for Open Digital Rights Language

Rettigheter beskrives i form av tilllatelser (permissions) som beskriver hva som er tillatte handlinger for dette innholdet i den gitte konteksten (context). Slike tillatelser kan for eksempel være tillatelsen til å spille av en lydfil eller tillatelsen til å gi en tillatelse videre til en tredjeperson

For hver tillatelse kan det defineres begrensninger(constraints) for hva som kan gjøres med materialet evt tillatelsene i forhold til materialet, og forpliktelser (requirements) som må oppfylles for at disse tillatelsene kan gis. En begrensning kan bestå i at lydfilen kun får spilles av 3 ganger, og en forpliktelse kan bestå i at brukeren må betale for å spille av fila. Dessuten kan det defineres betingelser (conditions) som fungerer som triggere for å markere en tillatelse som ikke lenger gyldig.

Parties kan deles i de to gruppene sluttbruker og rettighetsholdere. Rettighetsholderen (Rightsholder) er en bruker som er anerkjent som den rettmessige eier av et sett med rettigheter knyttet til et innhold. Disse kan lage tilbud (Offers) til andre brukere om overføring av rettigheter til et materiale i form av permissions. Disse tilbudene kan så brukes som utgangspunkt for å lage avtaler (Agreements) mellom rettighetsholderen og brukere om overføring av et subsett av rettighetsholderens rettigheter til sluttbrukeren. Sluttbrukeren er etter å ha inngått en slik avtale selv en rettighetsholder

Kontekst (Context) er en viktig, men ikke obligatorisk separator i språket og kan brukes til flere formål. De fleste andre entiteter og sammenhenger mellom entiteter kan beskrives ytterliger vha en context entity. For eksempel kan en tillatelse defineres som gyldig kun innen en gitt kontekst, for eksempel at brukeren befinner seg innen USA. En agreement kan linkes til det opprinnelige tilbudet eller en brukers rolle kan spesifiseres vha en kontekst.

Språket angir også muligheter for å trekke tilbake ettighetsbedkrivelser ved å angi at de ikke lengre er gyldige. Dette kan gjøres på to måter Den ene er å bruke språkets revoke entitet hvor de som har laget offers kan legge inn et ekstra dokument som, når det er tilstede annullerer både offers og agreements inngått. Den andre er å legge inn conditions i rettighetsbeskrivelsene. Dette er en angivelse av tilstander som, hvis de inntreffer, gjør at en gitt rettighetsbeskrivelse automatisk må anses som ikke lengre gyldig.

I tillegg til funksjonaliteten beskrevet over inneholder også språket mulighet for å beskrive kryptering av innhold og signering av rettigheter. Dette kalles språkets sikkerhetsmodell.

ODRL er en konseptuelt intuitiv modell og transformerer enkelt til en xml modell som er relativt enkelt å forstå samtidig som den er kraftig nok til å la brukere danne et vidt spekter av IPR-statements. I en del tilfeller er det likevel min oppfatning at ODRL ikke er et helt komplett språk for uttrykking av rettigheter fordi spesifikasjonen av en del konstruksjoner befinner seg på et såpass høyt nivå at det vil være opp til brukere av språket å angi hvilke datatyper og opplysninger som kan settes inn i de forskjellige konstruksjoner. Dette kan i mange tilfeller føre til at det ikke vil være mulig å overføre

rettighetsbeskrivelser fra ett system til et annet selv om begge hevder å støtte standarden. Et begrenset subset av ODRL-standarden vil bli brukt i min oppgave for å uttrykke IPR-statements.

9.1.2 eXtensible Rights Markup Language (XrML)

XrML (eXtensible Rights Markup Language) **[XrML]** er et XML-basert språk for å spesifisere rettigheter og betingelser for å kontrollere adgang til digitalt materiale og tjenester. Det har sitt utspring i Xerox Palo Alto research Center som laget det de i 1996 kalte ”Digital Property Rights Language” (DPRL) basert på Dr Mark Stefiks rettighetsmodell. (Jfr avsnittet om rettighetsmodeller). Standarden skiftet navn da metaspråket som brukes for å konstruere språket ble lagt om fra LISP til XML i 1999.

Ved å bruke XrML kan alle som eier eller distribuerer digitale ressurser (som for eksempel digitalt materiale, tjenester eller software) identifisere hvilke brukere som har lov til å bruke ressursene, rettighetene som er tilgjengelig for disse brukerne og betingelser og forutsetninger som gjelder for utøvelsen av disse rettighetene.

XrML tar sikte på:

> *Å være en åpen standard der tunge industrikrefter kan bidra med deres ekspertise og innfluere på utviklingen av språket.*
> *Å være til nytte i enhver Forretningsmodell inkludert multi-tier modeller*
> *Å tilby interoperabilitet mellom syntakser, semantikker og systemer for å kunne brukes som del av større systemer og inneholder også støtte for sikkerhet.*
> *Å være utvidbart ved at nye termer kan legges til etter som industrien utvikler seg.*

I tillegg har XrML laget det de kaller en ”Companion Language Software Development Kit” som inneholder biblioteker for å støtte utviklingen av software som kan forstå og manipulere uttrykk om rettigheter i språket.

Standarden eies og vedlikeholdes av ContentGuard Inc. Firmaet, som ble opprettet i april 2000, eies av Xerox Corporation (som det utgikk fra) mens Microsoft Corporation eier en minoritetsandel. Microsoft har sagt at de vil bruke standarden i fremtidige produkter, og har faktisk begynt å bruke den i produkter som Windows Media Player som er Microsofts fremvisningsapplikasjon for lyd og bilde og i Microsoft Reader som er klienten i Microsofts eBook system (.LIT formatet).

ContentGuard har som intensjon å overføre standarden til ”OASIS, the XML interoperability standards consortium” **[OASIS].** Dette for å åpne opp for en bredere deltakelse fra industrien i utviklingen av standarden. Det foreligger planer om å foreslå standarden til standardiseringsorganisasjoner som etterspør et Rettighetsspråk, og standarden har allerede blitt sendt inn til MPEG-21 og TV-Anytime for vurdering. For å bruke XrML må man få tildelt en Lisens av ContentGuard Inc.

Mange har hevdet at at standarden ikke er laget i open source tradisjonen/paradigme og mistenker Microsoft for å lage en standard som ikke vil være kompatibel med annen programvare. Microsoft er da også de eneste som har implementert standarden.

9.1.3 eXtensible Media Commerce Language (XMCL)

eXtensible Media Commerce Language **[XMCL]** er en relativt enkel, men fungerende standard utviklet av REALNetworks. Standarden bruker XML for å spesifisere rettighetsmetadata om en ressurs i henhold til en rettighetsmodell som beskrevet tidligere i oppgava.

Det viktigste konseptet i språket er en såkalt ”licence” som er en angivelse av ordinære metadata om ressursen, en angivelse av hvor lenge lisensen er gyldig og en angivelse av tilgjengelige rettigheter over ressursen. Språket uttrykker altså en tradisjonell rettighetsmodell.

Selv om standarden ser ut til å være skrevet med REALNetworks egen programvare i tankene, har mange store selskaper annonsert sin støtte til standarden. Iflg **[RosenblattTrippeMooney]** virker det likevel som om denne støtten mer har vært motivert av et ønske om å uttrykke støtte til enhver standard som kan være et alternativ til Microsofts XrML enn av faktiske planer om å bruke standarden i egne produkter. De mener også at standarden/språket er et alternativ på vei ut da det ikke virker som om standarden og bruken av standarden utvikles videre.

9.2 Viktige teknologier

9.2.1 Kryptering

Krypteringsteknologi er en av de viktigste byggestener i DRM systemer. Kryptering er prosessen med og teknologier i forbindelse prosessen med å konvertere data til en form som ofte kalles siffertekst slik at dataene ikke kan leses eller med stor vanskelighet kan leses av mennesker som ikke skal ha tilgang til dataene. Dekryptering er prosessen med å konvertere dataene tilbake til sin opprinnelige form slik at dataene igjen er lesbare. For å kunne konvertere data til siffertekst trenger man en algoritme og muligens nøkler som fungerer som inndata til algoritmene avhengig av hvilke algoritmer som brukes i konversjonen. For å kunne konvertere dataene tilbake til sin originale form må man ha tilgang til det riktige sett med algoritmer og nøkler.

Kryptering benyttes i DRM systemer hovedsakelig til å sørge for at data ikke er tilgjengelig unntatt når systemene eksplisitt vil gi tilgang til dataene. Dette gjøres ved at materiale stort sett distribueres i kryptert form og ved at nøklene som skal til for å konvertere sifferteksten til lesbare data som gir mening for brukere og applikasjoner distribueres kun til de som skal ha tilgang til dataene. Dessuten brukes kryptering til å sørge for at data som skal overføres i en transaksjon ikke kompromitteres i overføringen

ved at krypteringsteknologier inneholder metoder for å sjekke at dataene som resulterer av dekrypteringsprosessen svarer nøyaktig til dataene som ble kryptert ved hjelp av såkalte digest-verdier.

9.2.2 Vannmerking

Vannmerking er tradisjonelt forbundet med papir. Produsenter av papir som skal brukes til spesielle formål lager merker i papiret sitt som fungerer garanti for at papiret stammer fra en gitt kilde. Disse merkene er laget slik at de vanskelig lar seg reprodusere ved for eksempel kopiering. Derfor kan man for eksempel med rimelig sikkerhet garantere at pengesedler er ekte ved at bare legitime seddelprodusenter har tilgang til å kjøpe papir med det aktuelle vannmerket.

Vannmerker skal overbringe informasjon om et dokument på en slik måte at vannmerket ikke kommer i konflikt med resten av innholdet i et dokument og slik at det ikke er ikke er mulig å fjerne vannmerket fra dokumentet og dermed ta vekk garantien for at dokumentet er ekte.

Disse ideene har blitt overført til digitalverdenen. Begrepet vannmerking, som henviser til teknologien som brukes for å merke papir før det trykkes på det brukes i digitalverdenen som et substitutt for mer presise uttrykk som ”informasjonsskjuling”, ”datainkludering” og stenografi fordi det konseptuelt er veldig forklarende for hva disse teknologiene består av. Vannmerking er teknologier for å inkludere metadata om en ressurs i ressursen slik at metadataene blir del av ressursen selv heller enn å eksistere ved siden av dataene.

For at et vannmerke skal kunne fungere etter hensikten må en del forutsetninger være oppfylt:

- Vannmerket må være inkludert i en ressurs på en slik måte at det ikke er mulig skille det ut fra den visuelle eller audielle opplevelsen av presentasjonen av ressursen.
- Vannmerket bør være robust nok til å overleve en konvertering til formater med lavere oppløsning eller analoge formater og tilbake igjen.
- Vannmerket bør kunne ha mulighet til å ha lagre så mye informasjon som mulig.
- Vannmerket må være umulig eller i alle fall veldig vanskelig å fjerne fra ressursen eller forandre informasjonen i uten også å ødelegge selve ressursen. Dette gjøres ofte ved hjelp av krypteringsteknologier.
- Innsatsen for å legge inn informasjon i et vannmerke eller lese av informasjonen i et vannmerke må ikke være større enn at det oppleves som hensiktsmessig å bruke vannmerker som bærere av metadata.

Kryptering kan også brukes til å binde metadata til en ressurs ved at de to entitetene krypteres sammen til en kryptert enhet. Forskjellen mellom å bruke kryptering til å binde sammen metadata og ressurser og å bruke vannmerking er at for å vise fram et innhold

som er kryptert må man ha en eller annen slags dekrypteringsfunksjonalitet i programvaren som skal vise fram ressursen. Det trenger man ikke hvis man binder sammen metadata og innhold vha vannmerking. Dessuten binder vannmerking metadata permanent til innholdet de beskriver i motsetning til kryptering som ved dekryptering reproduserer metadata og data som selvstendige entiteter.

Noen filtyper er mer egnet til å inkludere vannmerker i enn andre. I bilder kan det velges en viss region der gitte pikslers farge- og lysintensitet kan varieres på en slik måte at de blir bærere av metadatainformasjon uten at forandringen blir synlig for det menneskelige øye. I lydfiler kan man legge til metadata i veldig lave frekvenser eller ultrasoniske frekvenser som ikke er hørbare for mennesker. Tekstfiler er det verre å vannmerke fordi mennesker oppfatter hver distinkt karakter i en tekstfil.

Vannmerking kan brukes til forskjellige ting innen DRM systemer.

- Man kan fastsette opphavet til en fil for å verifisere at man behandler den originale versjonen av fila og ikke en illegal reproduksjon eller forfalskning.
- Man kan inkludere rettighetsinformasjon i fila som applikasjoner kan bruke for å sikre at ressursen ikke brukes i strid med opphavsrett osv.
- Man kan spore bruk av ressursen ved at applikasjoner leser metadataene i vannmerket og rapporterer hva den gjør med fila til en sentral server angitt i vannmerket.
- Servere som publiserer data kan samle inn informasjon om maskinvare eller programvare som brukes når ressursen lastes ned før den sender fila til klienten. Ved å inkludere slik informasjon i vannmerker idet ressursen leveres kan man begrense bruken av fila til den maskina eller instansen av programvaren som var brukt til nedlasting.

Vannmerking brukes ofte i sammenheng med kryptering ved at vannmerket i en fil krypteres uten at ressursen selv krypteres og eller ved at fila som inneholder vannmerket krypteres før den sendes til en bruker.

9.2.3 Digitale signaturer

For å kunne sikre seg at en inngått avtale er mellom to parter er juridisk bindende har det vært vanlig å kreve en underskrift skrevet med blå eller svart penn på et dokument. Ekvivalenten til dette i nettverkssammenheng er digitale signaturerer. I Norge er digitale signaturer likestilt med håndskrevne signaturer etter innføringen av LOV 2001-06-15 nr 81: Lov om elektronisk signatur **[ElektroniskSignaturLov]** som trådte i kraft 1. juli 2001. Denne loven inneholder også angivelse av hvilke digitale signaturer som godtas som juridisk bindende i Norge. Også andre land i Europa og ellers i verden lages det lover som sidestiller digitale signaturer med håndskrevne. I USA er for eksempel digitale signaturer juridisk likestilt med håndskrevne signaturer på et ark etter innføringen av ”Electronic Signature in Global and National Commerce Act”. Slik lovgivning gjør at avtaler inngått over Internet kan gjøres juridisk bindende på samme måte som avtaler inngått og underskrevet på tradisjonell måte.

Digitale signaturer identifiserer entydig personen som signerer noe. Dessuten kan den digitale signaturen garantere at innholdet av det som er signert ikke er forandret etter at det var signert. Det sier seg selv at for DRM systemer som skal kunne auomatisere prosessen med å gjøre avtaler om overføring av rettigheter mellom juridiske personer, vil jevnstillingen mellom håndskrevne og digitale signaturer være av stor viktighet. Digitale signaturer er basert på krypteringsteknologi.

10 Mitt Digitale Bibliotek

10.1 Visjon

Målet med Mitt Digitale Bibliotek er å vise at man kan lage systemer som tar vare på rettigheter over et materiale som publiseres gjennom et digitalt bibliotek, gjennom å foreslå en modell for publisering av materiale som inkorporerer digital rettighetshåndtering (DRM).

Gjennom å bygge en prototype på en adgangsprotokoll for ressurser i et digitalt bibliotek vil jeg vise at man kan beskytte rettigheter i forbindelse med publisering og bruk av digitale ressurser. Beskyttelse av rettigheter og adgangskontroll vil i fremtiden være et viktig kompetanseområde for digitale bibliotek da spørsmålet om kontroll av adgang til, og styring av bruk av, digitale ressurser i et nettverksbasert miljø vil være avgjørende for hva slags materiale som vil bli distribuert i et digitalt bibliotek. Prototypen kalles REAP som er en forkortelse for ”Rights Enforcing Access Protocol”

10.2 DRM i DigLib. En behovsanalyse.

10.2.1 Forretningsmodell for REAP

For REAP kan man sette opp en del konkrete krav til hvordan materialet i DIGLIBS samlinger skal kunne distribueres, hvilke betingelser en slik distribusjon skal kunne gjennomføres under og hvordan man skal kunne håndheve disse betingelsene. Til sammen utgjør dette forretningsmodellen for DigLib/REAP.

Forretningsmodellen til REAP kan sammenfattes i de følgende punkter

- DigLib skal kunne distribuere alle typer materiale
- Et ubegrenset antall brukere har tilgang til materialet samtidig. Det settes ingen øvre grense for hvor mange kopier av materialet som skal kunne distribueres.
- Brukerne har forskjellig tilgang ut fra en rettighetbeskrivelse for materialet og dennes mapping til de opplysninger som er registrert om brukeren og andre opplysninger systemet kan hente ut fra brukerens tilknytning til Internet. Når

brukeren første gang ønsker tilgang til en ressurs vil systemet forsøke å lage en avtale mellom det digitale bibliotek og brukeren, basert på rettighetsbeskrivelsen for materialet og opplysningene som er registrert om brukeren. Basert på om brukeren oppfyller alle, ingen eller et subset av kravene som stilles for å få tilgang til materialet, lager systemet en avtale mellom brukeren og det digitale biblioteket som gir tilgang til alle, ingen eller et subsett av de mulige rettigheter over materialet som er definert i rettighetsbeskrivelsen for materialet.

- Materialet forlater aldri biblioteket. Materialet kan bare vises fram i dedikerte applikasjoner som er del av det digitale bibliotek, men som kan kjøres på brukerens datamaskin.

10.2.2 Rettighetsmodell for DIGLIB/REAP

Rettighetsmodellen for REAP må gjenspeile forretningsmodellen for REAP. Rettighetsmodellen må angi hvilke rettigheter som skal tildeles brukere og hvilke begrensninger som legges på de tildelte rettigheter, samt hvilke forutsetninger som må være oppfylt for at brukeren skal få tilgang til rettigheter over et materiale

For REAP vil det kun være aktuelt å modellere rettigheter som gjelder bruk av materialet. REAP skal distribuere materiale til sluttbrukere som ikke skal kunne gis tillatelse til å distribuere materialet videre. Derfor skal rettighetsmodellen for REAP bare modellere rettigheter som gjelder å lese, skrive ut, spille av og eksekvere et materiale.

Rettighetsmodellen for REAP er gitt av rettighetsspråket til REAP og blir behandlet i detalj under avsnittet om rettighetsspråk for REAP.

10.3 Kravsspesifikasjon for REAP

10.3.1 Overordnet mål for prototypen

Prototypen skal kunne fungere som en adgangsprotokoll som leverer materiale til brukere på en slik måte at brukere bare kan konsumere materialet på den for brukeren avtalte måte. Prototypen skal kunne uttrykke rettigheter knyttet til materiale som metadata av det samme materialet i et definert språk, og den skal kunne vise hvordan denne kodifikasjonen av rettigheter kan brukes til å håndheve de rettigheter som er knyttet til materialet.

10.3.2 Hva skal bygges?

Prototypen skal bestå av en ”Rights Enforcing Access Protocol” (REAP) som håndhever rettigheter knyttet til digitalt materiale. Denne skal kunne brukes sammen med ADEPT noden ved IDI NTNU (modersystem), men skal samtidig utformes slik at den også kan

brukes sammen med andre systemer, eller være en modell for andre som ønsker å utvikle lignende systemer. Kodifisering av rettigheter gjøres vha rettighetsspråk som er definert i denne oppgava. Arkitekturen for prototypen skal ta utgangspunkt i referansearkitekturen for DRM systemer som ble presentert tidligere i oppgava.

10.3.3 Avgrensning

Protokollen skal kun ta seg av rettighetshåndtering i forbindelse med gjenhenting av materiale. Alle andre funksjoner i digitale bibliotek skal tilligge komponenter og systemer som ligger utenfor prototypen.

Det tas utgangspunkt i at for alt materiale som publiseres i systemet har systemet allerede fått rett til å videreformidle materialet. Hvordan denne retten er ervervet er for prototypen irrelevant. Prototypen skal kun befatte seg med overføringen av rettigheter mellom det digitale bibliotek og en registrert bruker. Superdistribusjon skal ikke støttes.

10.3.4 Forutsetninger

Avgrensningen som ble gjort i første del av forrige avsnitt tilsier at vi må sette visse krav til metadata for materiale som skal publiseres gjennom prototypen. Navngiving av ressursen og informasjon om hvordan ressursen kan gjenhentes må kodifiseres inn i metadatformatene for ressurser som skal beskyttes. Søkesystemer må presentere denne informasjonen som del av sitt resultatsett som en URL til systemet. Identifikasjon av ressursen (identifikator for ressursen og samlingen den befinner seg) må angis som parameter til URL til systemet slik at brukere kan aksessere ressursen ved å aksessere en slik URL. Eksempel på en slik URL kan da være:

http://www.fenris.idi.tnu.no:8080/REAP/get_Document?collectionId=4?ItemId=2

10.3.5 Funksjonelle systemkrav

Funksjonelle systemkrav angir hva systemet skal gjøre. For REAP kan disse oppsummmeres som følgende:

- Holde oversikt over registrerte brukere og sentrale opplysninger om disse. Registerte brukere skal tildeles en individuell profil som inneholder sentrale og nødvendige opplysninger om brukerne. Denne profilen skal lagres i systemet slik at den kan aksesseres når brukeren logger på, og slik at den er tilgjengelig for systemet som helhet.
- Kodifisering av rettigheter som metadata. Systemet skal kunne **lage, transformere** og **ta vare** på, kvalifiserte uttrykk som kodifiserer rettigheter knyttet til digitalt publiserbart materiale ved hjelp av et standardisert språk, slik at

disse uttrykkene kan brukes til å håndheve de samme rettighetene i et digitalt miljø.

- Begrensing av overføring av rettigheter. Systemet må kunne gjenkjenne brukere og kontekster, og bare overføre rettigheter til personer som oppfyller kriteriene som er satt for rettigheten og som befinner seg i en kontekst som er definert for rettigheten.
- Håndheving av begrensninger av rettigheter. For et materiale vil et gitt sett av rettigheter eller ingen rettigheter være gitt til en gitt bruker. Systemet må sørge for at brukere ikke konsumerer materiale på en måte som går ut over de rettigheter som er gitt.
- Håndheving av forpliktelser i forbindelse med overføring av rettigheter. Systemet må kunne håndheve forpliktelser som brukeren må oppfylle før en rettighet til et materiale kan gis. Dette kan inkludere ting som betaling, registrering etc.
- Holde oversikt over rettighetshavere i det digitale bibliotek For at rettighetsholdere skal kunne krediteres for nedlasting av materiale er det viktig at systemet kan registrere opplysninger om disse som muliggjør slik kreditering
- Holde oversikt over gjennomførte rettighetstransaksjoner. For hvert materiale må det føres en logg over hvilke rettigheter som er gitt og i hvilket antall de er gitt slik at opphavsmannens krav til for eksempel betaling kan overholdes.
- Holde oversikt over en brukers tildelte rettigheter. Når en sluttbruker har fått tildelt en rettighet knyttet til et materiale må denne rettigheten tas vare på i brukerens profil, slik at rettigheten er tilgjengelig for brukeren ved senere påloggingger.

10.3.6 Ikke-funksjonelle systemkrav

- Systemavhengighet. Systemet skal bygges slik at det er mest mulig uavhengig av modersystem og underliggende samlinger samt navngivingsskjemaer, slik at det kan brukes av andre systemer eller brukes som modell for slike systemer.
- Systemarkitektonisk paradigme. Systemet skal bygges som samarbeidende moduler som hver for seg utfører en selvstendig, konkret oppgave. Ved å samarbeide på definerte måter skal de til sammen utføre en større oppgave

10.3.7 Modersystem

Fordi ikke alle deler og aspekter av et digitalt bibliotek er utviklet ved NTNU og fordi IDI NTNU er inne i et samarbeidsprosjekt med Alexandria Digital Library Project **[Alexandria]** vil deres system ADEPT brukes som modersystem. All søking og gjenfinning av informasjon foregår gjennom vanlig bruk av ADEPT via en ”standard” ADEPT klient. ADEPT tilfredsstiller forutsetninger som angitt over angående hvordan gjenhentingsinformasjon presenteres som del av resultatsett.

10.3.8 Kommunikasjon og samhandling med modersystemet

Avhengighet til modersystemet skal gjøres så liten som ovehodet mulig, slik at systemet senere kan brukes, eller brukes som modell ved andre utviklingsarbeider som bruker andre systemer/komponeneter for indeksering av, søking etter og navngiving av informasjonsressurser. Det lages et definert grensesnitt mot modersystemet som andre systemer kan ta ibruk.

10.3.9 Kommunikasjon og samhandling med samlinger.

Systemet må være i stand til å hente faktiske informasjonsressurser fra samlingene som inngår i systemet og presentere disse for brukerne i et html-grensesnitt.
Systemet skal gjøres så uavhengig av underliggende samlingers implementasjon som mulig.

10.3.10 Grensenitt mot brukere

Brukere skal kun være avhengig av en standard nettleser for å kunne kommunisere med systemet. Utveksling av sensitiv og rettighetsbelagt informasjon skal krypteres og brukere skal positivt identifiseres slik at det digitale bibliotek er rimelig sikker på at det kommuniserer med riktig vedkommende. Brukeren skal kunne logge seg inn på systemet en gang og så ha tilgang til all funksjonalitet i systemet helt til vedkommende logger seg av systemet eller er inaktiv over en gitt periode. I denne innloggingsperioden skal systemet holde kontroll over alle relevante opplysninger og profiler for kunden, og relevante opplysninger skal lagres i et kundearkiv før brukeren logges av.

10.3.11 Krav til samlinger

Hver ressurs må presentere en identifikator til adept som systemet kan forstå og som systemet kan bruke til å gjenhente ressursen. Dette kan være en lokal identifikator eller en mer persistent identifikator som for eksempel en handle.
Hver ressurs må ha metadata knyttet til seg som uttrykker rettigheter som tilbys for denne ressursen.

10.4 Arkitektur for REAP

10.4.1 Paradigme som systemet er bygd på

Arkitekturen tar utgangspunkt i referansearkitekturen for DRM systemer presentert tidligere i oppgava i kapittel 6.10. Dessuten bygger arkitekturen på en modell for hvordan

man kan gi tilgang til rettighetsbelagt materiale på en slik måte at opphavsrett med mer ikke krenkes.

Denne modellen består i at de som publiserer materiale gjennom REAP genererer et såkalt ”offer” eller rettighetstilbud som angir hvilke rettigheter de ønsker å tilby brukerne av det digitale bibliotek, hvilke forutsetninger som må være tilfredsstilt for å at rettighetene kan tildeles og hvilke begrensninger som gjelder for utøvelsen av disse rettighetene etter at de er tildelt. Disse rettighetstilbudene uttrykkes vha rettighetsspråket for REAP. For å få tilgang til materiale må brukere først inngå en avtale med systemet om tilgang, og ut fra denne avtalen kan det genereres såkalte tickets som lar brukeren eksekvere en rettighet en gang. Systemet genererer slike avtaler basert på rettighetsspråket for REAP.

Når en bruker forespør en ressurs første gang vil systemet forsøke å lage en avtale mellom brukeren og rettighetsholderen som avgjør hvilke rettigheter brukeren kan gis over materialet. I denne prosessen vil REAP fungere som en representant for rettighetsholderen(e) og grunnlaget for avtalen vil være det offer som rettighetsholderen har lagt inn for materialet og de opplysninger som brukeren angir om seg selv. Et offer vil typisk inneholde en oversikt over hvilke rettigheter rettighetsholderen tilbyr en bruker, hvilke begrensninger som gjelder hver enkelt rettighet og hvilke forutsetninger som må oppfylles for at brukeren skal gis tilgang til rettigheten.

Når systemet genererer en avtale vil det se på hvilke forutsetninger som er satt for å få tilgang til en gitt rettighet og sammenholde det med hvilke av disse forutsetningene en brukeren har valgt å oppfylle. Rettigheten gis i avtalen dersom det er en overenstemmelse mellom hvilken forutsetning brukeren har oppfylt og hvilken forutsetning som kreves i et offer. En slik avtale kan gi adgang til alle eller et subset av de rettigheter som eieren av materialet tilbyr ut fra hvilke rettigheter brukeren ønsker tilgang til og er villig til å oppfylle forutsetningene for. I prosessen med å generere en avtale lages det også en rapport om den inngåtte avtalen som skrives til en tekstfil på systemet. Denne tekstfila kan periodisk evalueres for for eksempel å kunne betale ut godtgjørelse til eiere av ressurser basert på hvor mye penger som er generert som følge av at brukere har inngått avtaler om overføring av rettigheter.

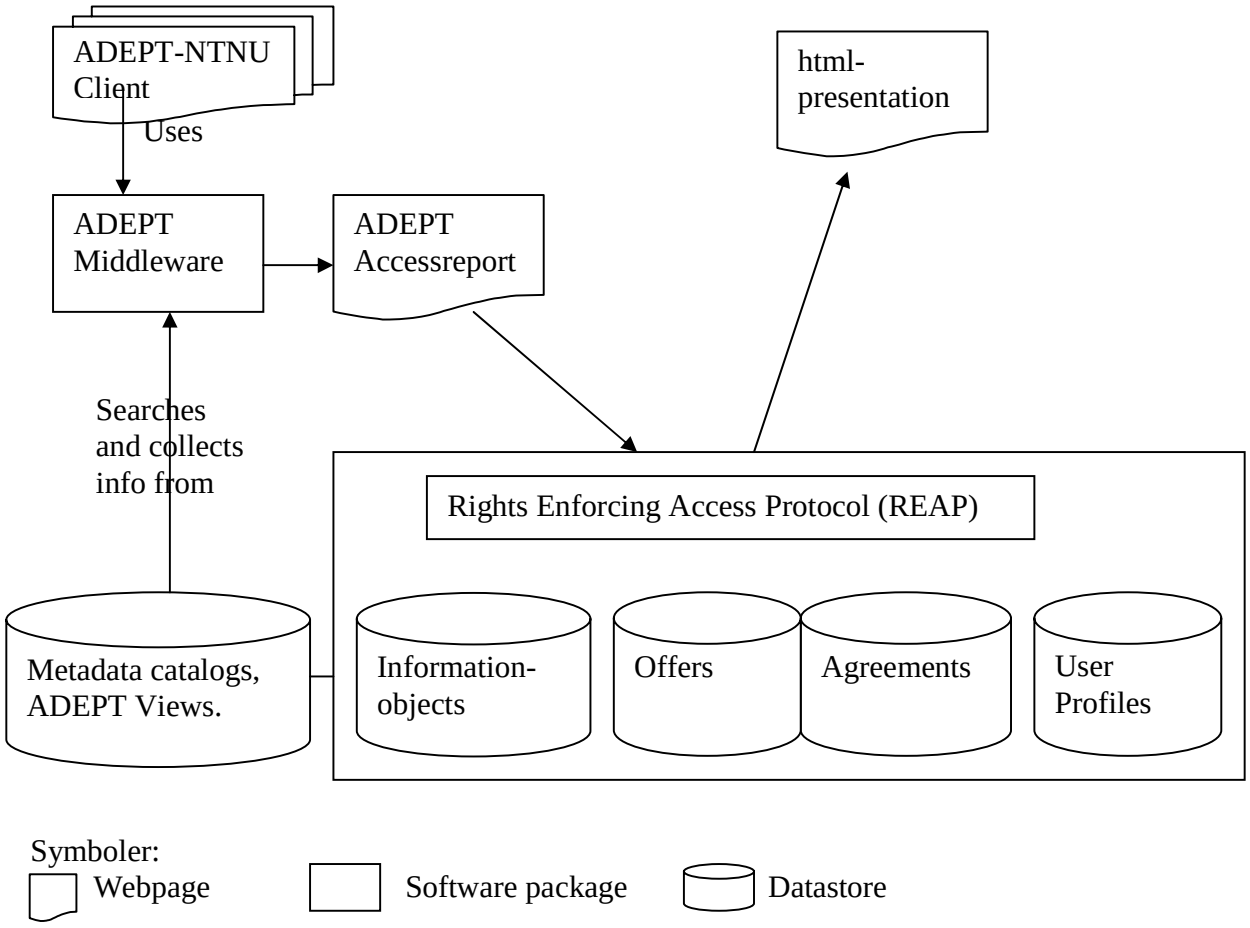
Når en slik avtale er generert kan brukeren aksessere materialet og utøve rettigheter over det i henhold til denne avtalen. Hver forespørsel om for eksempel å skrive ut materialet krever da at fremvisningsapplikasjonen som brukeren benytter først kontakter systemet for å få ut en såkalt ”ticket” for utøvelse av denne rettigheten. En ”ticket” er en tillatelse til å aksessere en rettighet en gang. Genereringen av slike tickets foregår ved at systemet samler informasjon om brukeren og evaluerer avtalen opp mot denne informasjonen for å se om brukeren krenker de begrensninger som er satt for utøvelsen av den rettigheten som brukeren forespør.

Hvis brukeren ikke krenker de begrensninger som er satt for utøvelsen av denne rettigheten iflg sin avtale, gis brukeren en slik ”ticket” til rettigheten, ellers gis det en

feilmelding (kan ses på som en ugyldig ticket) tilbake og brukeren kan ikke aksessere rettigheten.

10.4.2 REAP og ADEPT

Figur 9-1 under viser hvordan REAP er tenkt å fungere sammen med ADEPT. Utgangspunktet er at brukeren søker etter informasjon i ADEPT nettverket vha en ADEPT klient. Som resultat av dette søket får brukeren ut en ADEPT Access report som presenterer søkeresultatet for brukeren. En slik access report må da inneholde en url for å aksessere materialet som beskrevet i kapittel 10.3.4 Forutsetninger. Når brukeren aksesserer denne url'en vil REAP kontaktes og vil forsøke å lage en avtale med brukeren som vil avgjøre hva brukeren kan gjøre med materialet. Denne avtalen er basert på de Offers som rettighetsholderne/eierne av et materiale har lagt inn i databasen og opplysninger om brukeren. Etter at brukeren har fått generert en avtale vil REAP lage en html presentasjon av materialet for brukeren som gjenspeiler de rettigheter over materialet som brukeren har fått gjennom sin avtale med REAP.

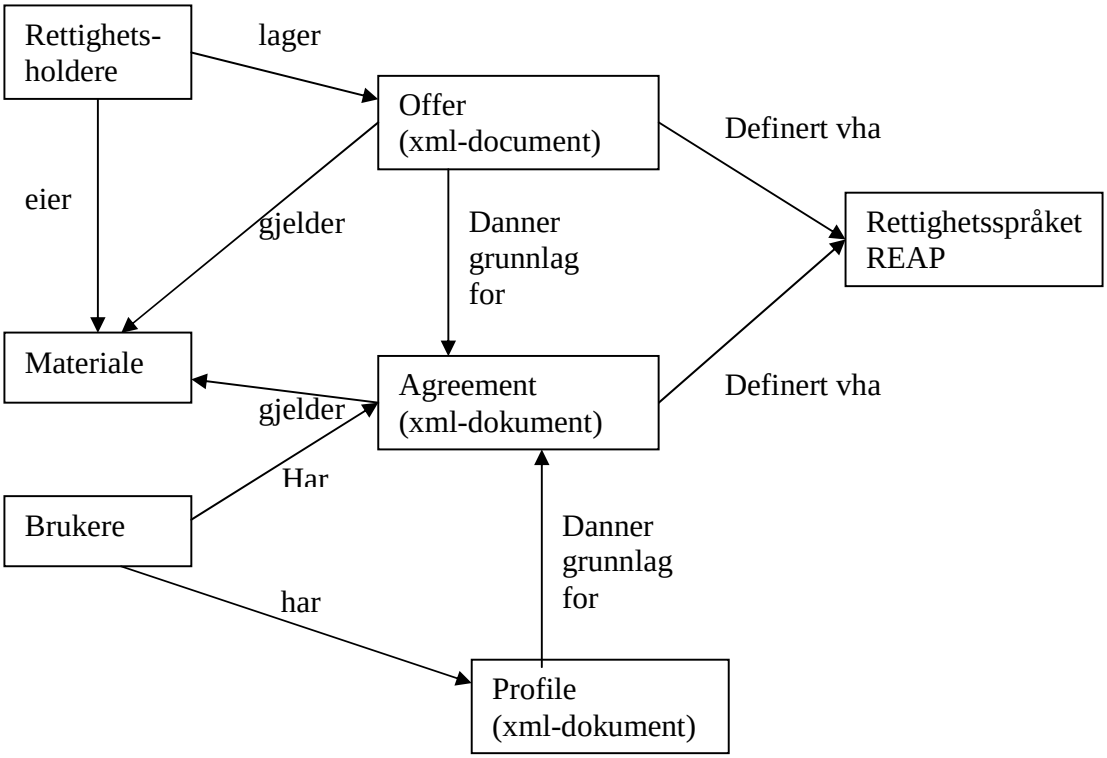


Figur 10-1 Sammenhengen mellom REAP og ADEPT

10.4.3 Rettighetsmodell for REAP

Prototypen skal bruke et eget språk for å lage kvalifiserte uttrykk om brukeres rettigheter over materialet og rettigheter som kan tilbys brukere av systemet over gitte ressurser. Dette språket skal kunne uttrykke både offers om rettigheter som kan tildeles brukere, og agreements som styrer hvordan en enkelt bruker kan bruke materialet. I tillegg skal systemet lage et enkelt dokument som representerer brukerens profil. Denne inneholder en del opplysinger om brukeren som kan være viktig å holde rede på.

Strukturelt kan vi framstille sammenhengen mellom et materiale, dets rettighetsholdere, brukere og rettighetbeskrivelser som angitt av figuren under



Figur 10-2 Strukturell sammenhengen mellom materiale, rettighetsholdere, brukere og rettighetbeskrivelser

Alt materiale som publiseres gjennom REAP skal tilknyttes et REAP Offer som uttrykker et tilbud om hvilke rettigheter som *kan* tildeles en bruker. Dette tilbudet lages av rettighetsholdere som eier materialet. Dessuten skal det til hver bruker knyttes en profile som inneholder alle stabile opplysninger om brukeren samt de avtaler som brukeren har inngått om konsumering av materiale.

10.4.4 Rettighetsspråk for REAP

Om valget av rettighetsspråk

Når man utvikler et Digital Rights Management system må man fastlegge hvordan systemet skal kunne fastslå hvilke rettigheter som skal kunne tildeles en bruker når denne forespør adgang til et materiale. Den vanligste måten å gjøre dette på er å definere et rettighetsspråk som kan uttrykke rettighetsbeskrivelser på en slik måte at programvare kan forstå og handle på grunnlag av det.

Ut fra ønsket om å basere REAP på et xml-språk som kan uttrykke rettigheter og gitt Open Digital Rights Language som utgangspunkt stod undertegnede overfor valget om å lage et system som kunne ta alle rettighetsbeskrivelser uttrykt vha ODRL som input eller å definere et eget subspråk basert på ODRL som kunne fungere som et applikasjonsspesifikt språk for REAP.

Fordelen med å bruke ODRL (eller et annet rettighetsspråk) uten forandringer ville vært at alle som kjente standarden kunne laget rettighetsbeskrivelser (eller programmer som laget rettighetsbeskrivelser) som REAP kunne forstå. Dermed kunne rettighetsbeskrivelsene brukes til å trade med rettigheter over et materiale i hele livssyklusen til et materiale helt fra fødsel til bruk og gjenbruk. REAP ville da bare vært et av mange ledd i kjeden av rettighetsholdere som handlet med materialet.

Selv om dette hadde vært en ideell situasjon ville man da stå overfor valget om å gjøre REAP til en avansert programvare som kunne forstå hele ODRL standarden (noe som så vidt jeg vet ikke er gjort enda, selv om mobiltelefonprodusenten Erickson har en delvis implementasjon) eller å la REAP bare forstå deler av standarden og ikke trade med rettigheter der man ikke klarer å verifisere at alle forutsetninger for trading er oppfyllt eller at man ikke klarer å verifisere at alle begrensninger blir overholdt. (Dette er også nevnt som en mulighet i beskrivelsen av ODRL)

Fordelen med å lage et forenklet applikasjonsspesifikt språk er at man kan begrense uttrykksmulighetene i språket så mye at man kan lage en programmvare som forstår alle aspekter ved språket. Dernest vil det være lettere å sette seg inn i et slikt redusert språk og man trenger ikke ha full kunnskap til ODRL standarden for å kunne lage rettighetsbeskrivelse for REAP. Bakdelen er at for materiale som skal publiseres og som allerede har en rettighetsbeskrivelse uttrykt ved ODRL knyttet til seg må denne konverteres til REAP sitt applikasjonsspesifikke språk. Det betyr at kjeden av rettighetstrading som ODRL er ment å kunne støtte ved å uttrykke alle aspekter ved den blir brutt.

Rettighetsspråk for REAP

Rettighetsspråket for REAP er gjengitt i appendix A i form av et XML Schema Dokument. Det kan i enkle vendinger uttrykke hvilke brukere som skal kunne utøve hvilke rettigheter over et materiale, hvilke betingelser som må oppfylles for at disse

brukerne skal kunne få tilgang til disse rettighetene og begrensninger i utøvelsen av rettighetene. Dette rettighetsspråket danner den grunnleggende logikken for REAP og prototypen er bygd opp for å forstå, manipulere med og handle på grunnlag av uttrykk laget ved hjelp av dette språket.

Rettighetsspråket for REAP er inspirert av Open Digital Rights Language **[ODRL]**, men har ikke på langt nær den uttrykkskraft som ODRL har. Rettighetsspråket for REAP ble utviklet med tanke på at det skulle være et subset av ODRL, på en slik måte at ingen nye elementer skulle legges til, men i arbeidet med dette subsettet viste det seg at heller ikke ODRL på alle områder var helt modent, og på noen områder ikke helt tilpasset mine behov. Dessuten fantes det enkelte aspekter ved ODRL som ikke var helt i samsvar med andre standarder, og som dermed forvansket bruken av ODRL som rettighetsspråk for REAP. (Disse ble rapportert inn til Dr. Renato Ianella og er forhåpentligvis rettet i den nye versjonen av ODRL som ble sluppet i august 2002.) Derfor utviklet rettighetsspråket for REAP seg til et eget språk, men kan fortsatt anses som et subsett av ODRL med noen tillegg og lokale tilpasninger.

Hva kan rettighetsspråket for REAP uttrykke??

Dette er en gjennomgang av rettighetsspråket REAP. Alle figurer og xml-snutter er hentet fra en behandling av rettighetsspråket i xml-editoren XML Spy versjon 4.4.

Rettighetsspråket for REAP kan uttrykke både tilbud om rettigheter over et materiale som kan tildeles brukere av systemet, og avtaler om overføring av slike rettigheter som brukere inngår basert på tilbud om rettighetsoverføring. Tilbud om rettighetsoverføring er typisk de tilbud om rettighetsoverføring som opphavsmenn eller andre rettighetsholdere tilbyr brukerne av systemet, mens avtaler inngås mellom brukeren av systemet og systemet som på sin side handler som en representant for rettighetsholderen/opphavsmannen for materialet.

Felles for både avtaler og tilbud er at de har et rot-element kalt rights som kan ett enkelt subelement som angir om det aktuelle xml dokumentet er et tilbud om rettighetsoverføring eller en inngått avtale om en rettighetsoverføring. Dette kan illustreres slik:

Figur 10-3 Rettighetsspråk for REAP. Offers og Agreements

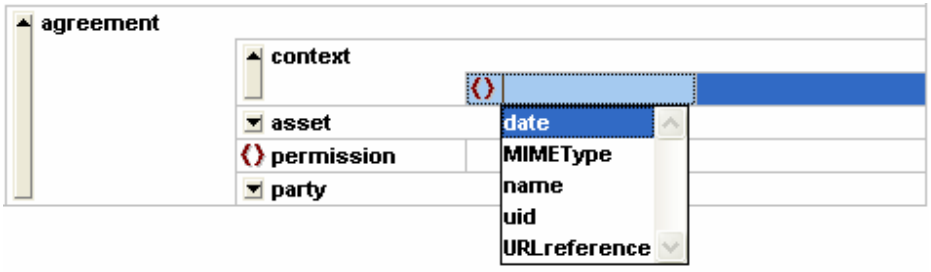
Både tilbud/offer og avtaler/agreement består av følgende elementer:

- context. Dette elementet brukes for å beskrive avtalen eller tilbudet

- asset. Dette elementet brukes for å beskrive materialet som avtalen eller tilbudet gjelder
- permission. Dette elementet brukes for å angi hvilke rettigheter brukeren tilbys eller har inngått avtale om
- party. Dette elementet brukes for å beskrive partene i en avtale eller et tilbud.

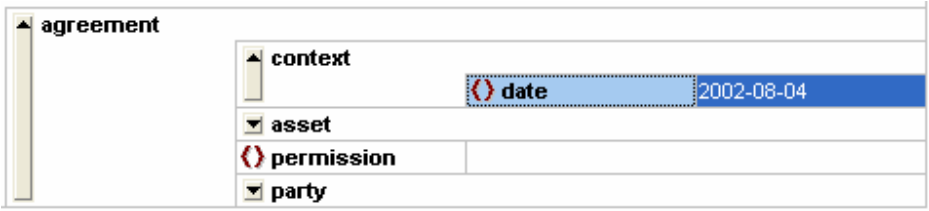
Figur 10-4 Rettighetsspråk for REAP. Oppbygging av offers og agreements

Elementet context brukes for å beskrive elementer på mange nivåer i språket og har mange forskjellige subelementer, men i denne sammenhengen er det meningen å bruke context til å angi en dato for når dokumentet var opprettet.



Figur 10-5 Rettighetsspråk for REAP. Beskrivelse av offer/agreement vha context.

Alle datoangivelser må være i overensstemmelse med ISO-standarden for datoer **[ISO8601]**, dvs på formen yyyy-mm-dd. En gyldig context for tilbud eller avtaler som beskriver datoen for opprettelse ville derfor kunne se slik ut:

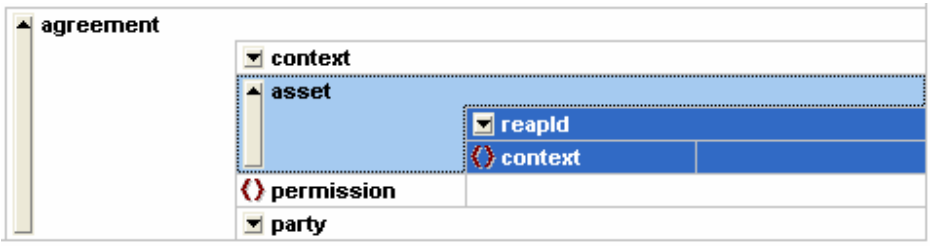


Figur 10-6 Rettighetsspråk for REAP. Et eksempel på beskrivelse av offer/agreement vha context

Elementet asset brukes for å beskrive materialet som tilbudet eller avtalen gjelder. Det består av følgende elementer

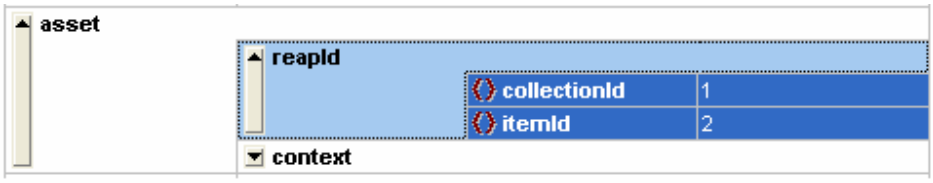
- reapId. Dette er et lokalt navngivingsskjema for REAP som brukes for å angi hvilket materiale i REAPs samlinger det er snakk om

- context. Dette er et element som angir noen generelle beskrivelser av materialet



Figur 10-7 Rettighetsspråk for REAP. Asset beskrives av reapId og context.

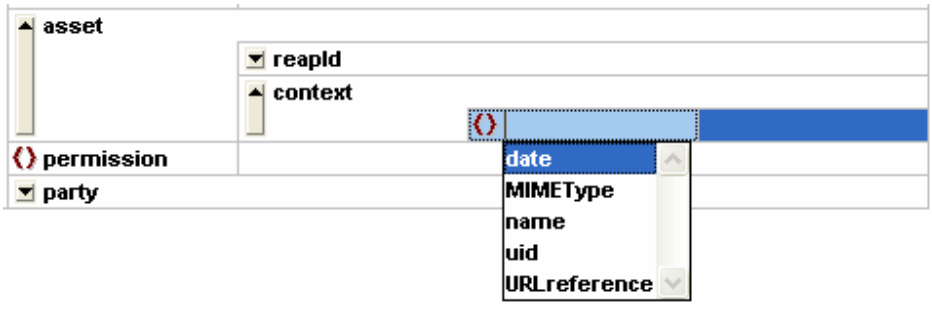
Elementet reapId brukes for å angi en lokalt entydig referanse til materialet som består av en collectionId og en itemId. collectionId angir i hvilken samling materialet befinner seg og itemId angir hvilket materiale i den aktuelle samlingen det er snakk om. Begge verdiene må være en positiv heltallsverdi. Defineringen av et lokalt navneskjema som ikke forstås av andre systemer er gjort fordi denne informasjonen er synlig for brukeren når brukeren forespør materialet, og det er viktig at denne informasjonen ikke kan brukes for å gjenhente ressursen utenfor REAP. REAP systemet inneholder funksjonalitet for å oversette dette navneskjemaet til faktisk lokasjon for materialet. Denne informasjonen vil aldri være synlig for brukeren. Et eksempel på angivelse av en slik reapId kan være



Figur 10-8 Rettighetsspråk for REAP. Eksempel på beskrivelse av Asset vha reapId.

Elementet context brukes for asset til å beskrive ressursen avtalen/tilbudet gjelder videre. Aktuelle subelementer her er:

- MIMEType. Kan brukes til å angi hvilken type fil det er snakk om, for eksempel ”image/jpg”
- name. Kan brukes til å angi filnavnet for ressursen
- uid. Kan brukes for å angi en globalt unik identifikator for ressursen.
- URLreference. Kan brukes for å angi en url til et dokument som beskriver ressursen videre. Det er her viktig at denne url ikke angir en alternativ nedlastingslokasjon. Det vil undergrave hele hensikten med systemet.



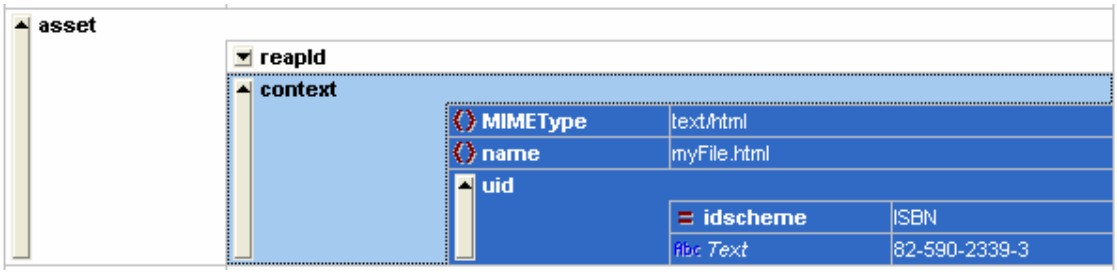
Figur 10-9 Rettighetsspråk for REAP. Beskrivelse av Asset vha context.

Elementet uid kan brukes til å angi en globalt unik identifikator for materialet. Dette elementet baserer seg på eksisterende standarder for globalt unike standarder. Elementet uid har et attributt kalt idscheme som angir hvilket identifikatorskjema som brukes og en verdi som angir den aktuelle identifikatoren. Attributtets verdi er begrenset til en gitt mengde verdier av identifikatorskjemaer:

- URI **[URI]**
- DOI **[DOI]**
- x500 **[x500]**
- ISBN **[ISBN]**
- ISTC **[ISTC]**
- ISO3166 **[ISO3166]**
- FPI **[FPI]**
- GUID **[GUID]**
- mpeg_ID **[mpeg_ID]**
- ISAN **[ISAN]**
- ISRC **[ISRC]**
- ISSN **[ISSN]**
- ISWC **[ISWC]**
- IDDN **[IDDN]**
- UUID **[UUID]**

Av disse er kun de som identifiserer materiale aktuelle for beskrivelse av asset.. Identifikatorskjemaer som for eksempel Unique User Identifikation (UUID) brukes selvfølgelig ikke for å entydig identifisere et verk for eksempel. Her er det også viktig at det ikke angis verdier som kan brukes for å angi alternative nedlastingslokasjoner, da dette ville undergrave hele hensikten med systemet.

Et gyldig eksempel på beskrivelse av et materiale (asset) ved hjelp av context kan da se slik ut:



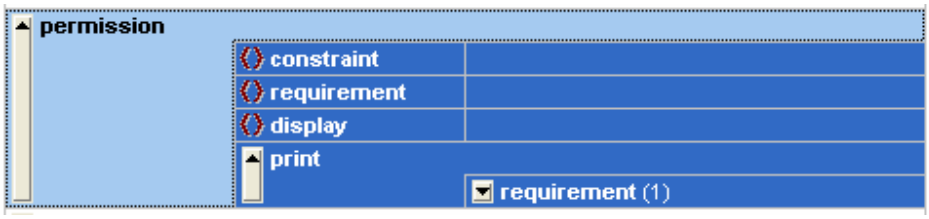
Figur 10-10 Rettighetsspråk for REAP. Eksempel på beskrivelse av asset vha context.

Elementene offer og agreement inneholder en beskrivelse av hvilke rettigheter som kan tildeles brukeren og hvilke betingelser og begrensninger som gjelder for disse. Betingelser (requirements) må innfris av brukeren før det kan genereres en avtale mellom systemet (som representerer eieren/rettighetsholderen for materialet i prosessen med å generere en avtale) og brukeren vedrørende bruk av materialet. Constraints er begrensninger i utøvelsen av rettigheten etter at det er inngått en avtale. Rettigheter kan i rettighetsspråket for REAP være av fire typer:

- **display**. Betyr retten til å lage en flyktig representasjon av ressursen på et dertil egnet apparat, for eksempel en dataskjerm.
- **print**: Betyr å lage en permanent kopi av ressursen, for eksempel på papir eller lysark
- **execute**: Betyr å eksekvere materialet dersom det er av eksekverbar art (programvare).
- **play**: Betyr å lage en flyktig presentasjon av ressursen på et dertil egnet apparat, for eksempel en dataskjerm på en slik måte at når presentasjonen er ferdig er ikke lengre ressursen tilgjengelig. Typisk brukt for streaming media som lyd og levende bilde (video)

Slike rettigheter er alltid deklarert direkte under elementet permission.

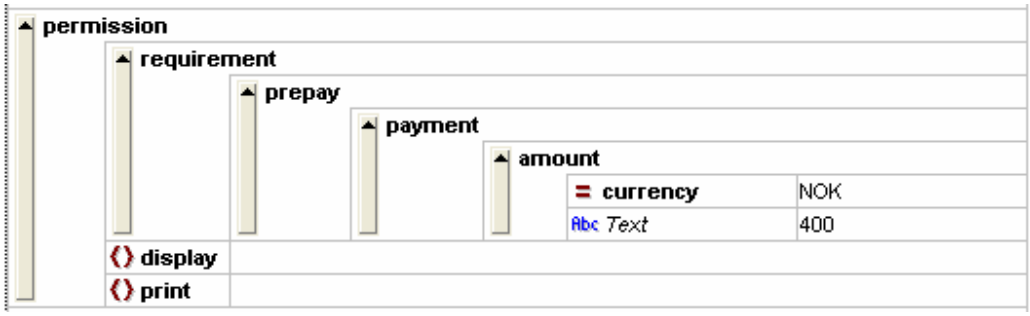
Constraints og requirements kan deklareres direkte under elementet permissions eller under en av rettighetene display, print, execute eller play. Hvis constraints eller requirements er deklarert på samme nivå som rettighetene betyr det at de gjelder for alle deklarerte rettigheter. Hvis constraints eller requirements er deklarert under en enkelt rettighet betyr det at disse constraints eller requirements kun gjelder for rettigheten de er deklarert under. Dette kan illustreres slik:



Figur 10-11 Rettighetsspråket for REAP. Deklarasjon av rettigheter og lokale og globale constraints og requirements.

Her er det først deklarert et constraint element og et requirement element. Disse befinner seg på samme nivå som rettighetene display og print og gjelder for begge rettighetene. For rettigheten print er det i tillegg definert en forutsetning (requirement) direkte under rettigheten print. Denne betingelsen gjelder bare for rettigheten print.

Forutsetninger (requirements) i REAP er for enkelhets skyld begrenset til å kreve at brukeren betaler for adgang til rettigheter over ressursen før det kan inngås en avtale om tildeling av slike rettigheter over ressursen. Derfor har requirement bare et subelement; prepay. Denne modellen kunne utvides med flere elementer som for eksempel postpay, som ville antyde at brukeren skulle betale etter å ha inngått avtalen for eksempel ved hjelp av sjekk eller overføring av penger på et senere tidspunkt eller peruse som ville svart til at brukeren betalte for hver gang vedkommende utøvet en rettighet over materialet. Dette ville satt større krav til systemet som skulle forstå, og handle på grunnlag av utttrykk laget i språket. Elementet prepay bruker en gjenbrukbar konstruksjon kalt payment som sitt subelement. Denne konstruksjonen har kun et subelement, nemlig amount. For amount angis en valutatype som attributt og et beløp som skal betales som verdi. Illustrasjonen under angir da at for å få tilgang til rettighetene display og print må brukeren først betale den nette sum av 400 norske kroner.

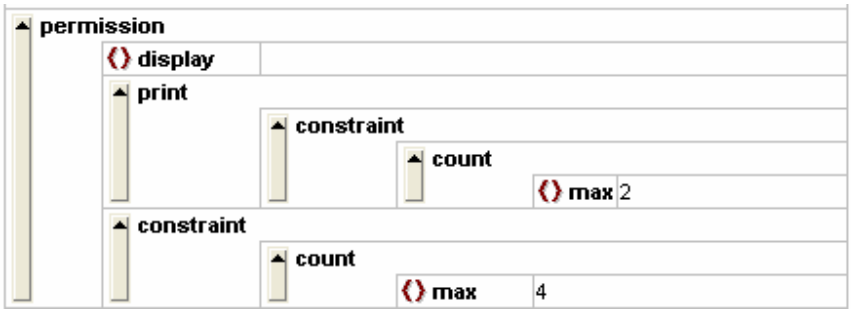


Figur 10-12 Rettighetsspråk for REAP. Deklarering av requirements.

Begrensninger (constraints) som kan defineres i rettighetsspråket for REAP inkluderer

- **count**. Angir hvor mange ganger en rettighet over et materiale kan eksekveres.
- **dateRange**. Angir en start og sluttdato for når ressursen kan eksekveres
- **individual**. Angir at utøvelsen av rettigheten over et materiale er begrenset til en angitt person.
- **Network**. Angir at rettigheten over materialet bare kan utøves fra datamaskiner som er del av et gitt nettverk

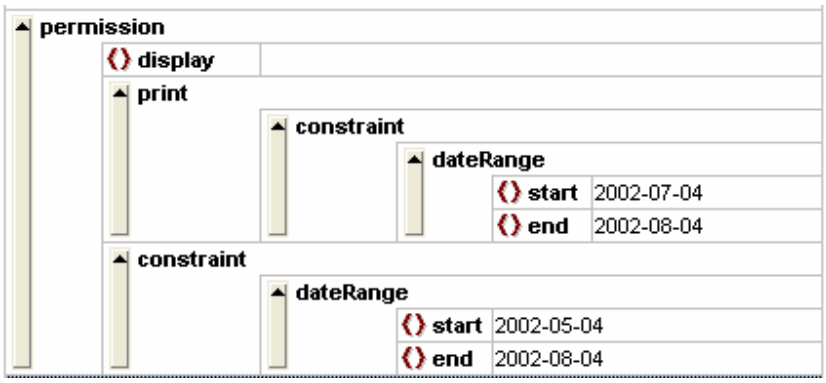
Begrensningen count kan defineres både på samme nivå som de enkelte rettigheter (globalt) eller for en enkelt rettighet (lokalt). Hvis det er konflikt mellom global og lokal deklarasjon av count vil den spesifikke rettigheten bare kunne eksekveres det antall ganger som er gitt av den minste verdien av count deklarert globalt og lokalt



Figur 10-13 Rettighetsspråk for REAP. Lokal og global deklarasjon av constraint count.

Her ser vi at det er en global deklarasjon av count der verdien av max er satt til 4. Altså kan alle rettigheter over dette materialet eksekveres max 4 ganger. Men for print er det satt en ytterligere begrensning av dette. Her er det satt en begrensning som bare gjelder for rettigheten print som sier at rettigheten print bare kan eksekveres max 2 ganger. Fordi denne verdien er lavere enn verdien av count som er deklarert globalt, vil systemet se på den lokale verdien når det skal avgjøre hvor mange ganger rettigheten print kan eksekveres. I dette tilfelle betyr det at brukeren kan vise fram materialet på skjermen sin max 4 ganger og skrive ut materialet max 2 ganger.

Begrensningen dateRange kan også deklareres både globalt og lokalt:

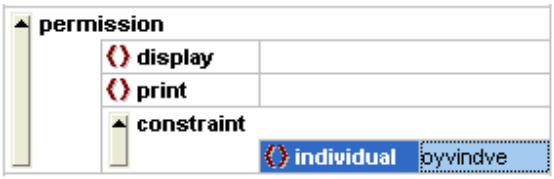


Figur 10-14 Rettighetsspråk for REAP. Lokal og global deklarasjon av constraint dateRange.

Denne deklarasjonen vil si at alle rettigheter i utgangspunktet kan eksekveres innenfor en tidsramme som er angitt av start: 4. mai 2002 og end: 4. august 2002. Men fordi det for print er deklarert et kortere tidsintervall (intervallet trenger ikke overlappe i det hele tatt) kan denne spesifikke rettigheten bare eksekveres innenfor en tidsramme fra 4. juli til 4. august 2002. Rettigheten display kan likevel eksekveres fra 4. mai til 4. august 2002.

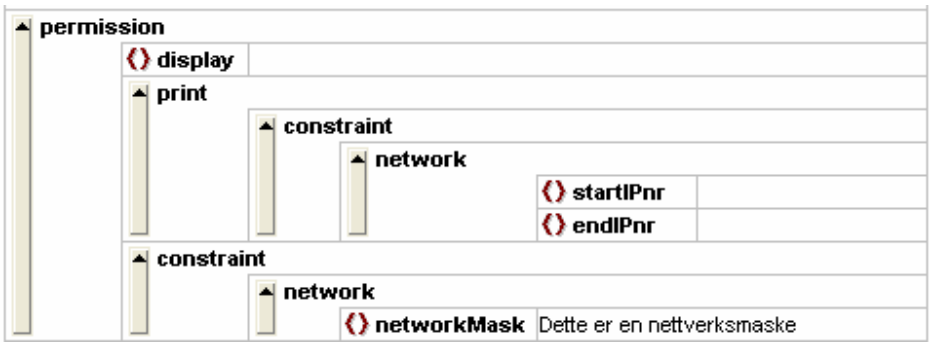
Begrensningen individual sier at ekseveringen av rettigheten bare kan utføres av en identifisert person. Begrensningen kan defineres både globalt, i hvilket tilfelle den gjelder for all deklarte rettigheter og lokalt, i hvilket tilfelle den gjelder kun for den rettigheten

den er deklarert direkte under. Under er et eksempel på deklarasjon der rettighetene display og print er begrenset til brukeren oyvindve.



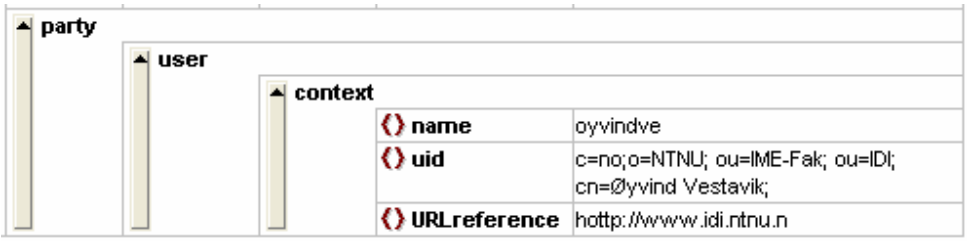
Figur 10-15 Rettighetsspråk for REAP. Deklarasjon av constraint individual.

Begrensningen network sier noe om hvilket nettverk maskina du bruker må være innenfor når du skal aksessere en rettighet over et materiale dersom rettigheten skal kunne gis. Nettverk kan identifiseres vha et spenn av IP adresser eller en nettverksmaske, og kan deklareres både globalt og lokalt. Et eksempel på dette er gitt under hvor det for alle rettigheter er definert en nettverksmaske som du må være innenfor for å få tilgang til rettigheter generelt og et spenn av IP adresser som du må være innenfor for å få tilgang til print rettigheten print.



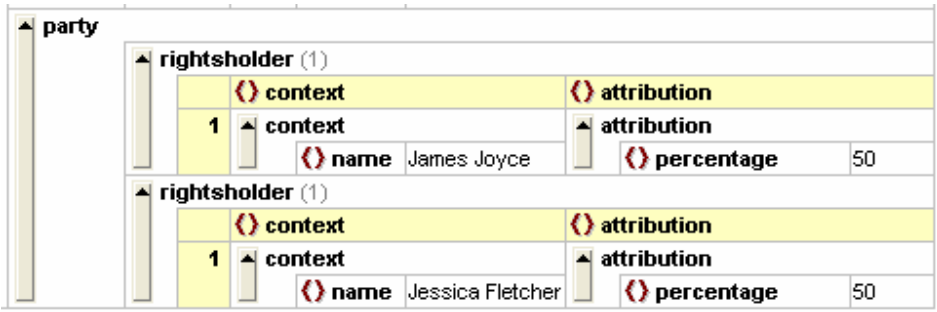
Figur 10-16 Rettighetsspråk for REAP. To metoder for å deklarere constraint network

Den siste hoveddelen som offers og agreements i REAP består av er **party**. Dette angir for offers rettighetsholderne som har laget tilbudet, og for agreements, partene som har inngått avtalen. Elementet party har to mulige subelementer; rightsholder og user. **User** er brukeren som har laget en avtale med systemet om adgang til rettigheter over materialet. User elementet beskrives videre av context og her kan man bruke forskjellige elementer til å beskrive en bruker. Både name, uid og URLreference kan brukes for å identifisere en bruker, enten alene eller i kombinasjon. Det beste er å bruke uid og velge et skjema for unik navngiving for eksempel UUID, men det utelukker ikke bruk av andre i tillegg. Et eksempel på bruk av slik identifisering er gitt under.



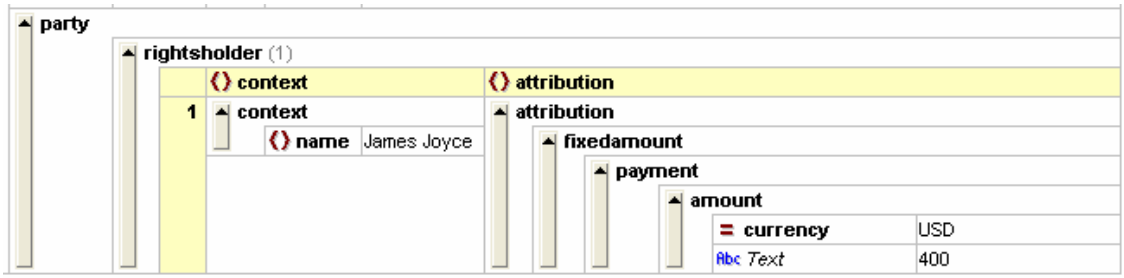
Figur 10-17 Rettighetsspråk for REAP. Beskrivelse av user vha context.

Rightsholder er den eller de som laget tilbudet om rettighetsoverføring. I en evt avtale blir de den ene parten i avtalen der brukeren er den andre. Rettighetsholdere beskrives på samme måte som user vha context, men i tillegg er det for rettighetsholdere definert et subelement som kalles attribution. Dette angir hvilken kompensasjon som skal ytes til rettighetsholdere når det inngås en avtale med brukere om overføring av rettigheter over et materiale. Dette er typisk penger som skal genereres til rettighetsholderne etter at REAP har fått inn pengene fra brukerne etter at en avtale har blitt generert. Rettighetsholdere skal enten ha et fast beløp hver gang en avtale inngås eller en prosentandel av den summen brukeren betaler for å få tilgang til rettigheter. Det kan være flere rettighetshaver og de kan dele det innkomne pengebeløpet mellom seg. Under er gjengitt et eksempel på 2 rettighetsholdere som deler den genererte betalingen mellom seg i like andeler:



Figur 10-18 Rettighetsspråk for REAP. Beskrivelse av rettighetsholdere vha context og angivelse av attribution vha percentage.

Det er også mulig at rettighetsholderne skal ha et fast beløp når en avtale inngås. Et eksempel på en enkelt rettighetsholder som skal ha et fast beløp hver gang en avtale genereres er gjengitt under.



Figur 10-19 Rettighetsspråk for REAP. Beskrivelse av rettighetsholdere vha context og angivelse av attribution vha fixedamount

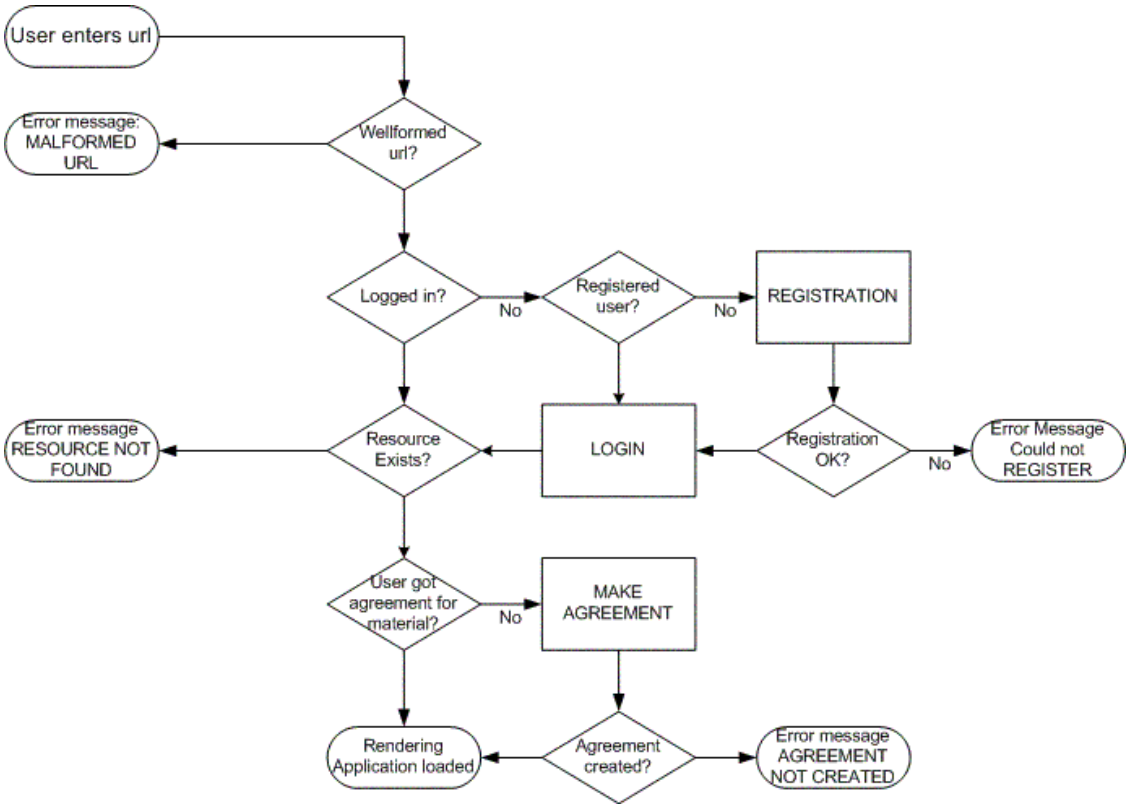
Ved hjelp av dette ganske begrensede språket kan man altså uttrykke hvem som skal ha tilgang til rettigheter til et materiale, hvilke betingelser som må være oppfyllt før det kan gis tilgang til retttighetene over materialet, og hvordan man definere begrensninger i utøvelsen av rettighetene etter at de er tildelt samt hvordan man deler økonomisk vederlag mellom rettighetsholderne.

10.4.5 Oversikt over REAP

Dette delkapittelet inneholder skisser og tegninger som viser både den arkitektoniske og konseptuelle oppbygginga av REAP. Hensikten er ikke å komme med utfyllende UML diagrammer som skal kunne fungere som grunnlag for implementasjonen av REAP men heller å illustrere tankegangen bak implementasjonen. Som nevnt tidligere er arkitekturen basert på referansearkitekturen for DRM systemer presentert tidligere i oppgava og leseren oppfordres til å ha denne i tankene ved lesing av dette kapittelet.

10.4.5.1 Eksekveringsstier i REAP

Dette er et enkelt Flowchart som viser gangen i REAP fra en bruker forspør et materiale til materialet er presentert for brukeren. Rektangler angir arbeidsprosesser og avrundede rektangler angir startpunkter eller endepunkter i eksekveringen. Rektangler satt på høykant angir valg mellom forskjellige eksekveringsstier hvor det er aktuelt



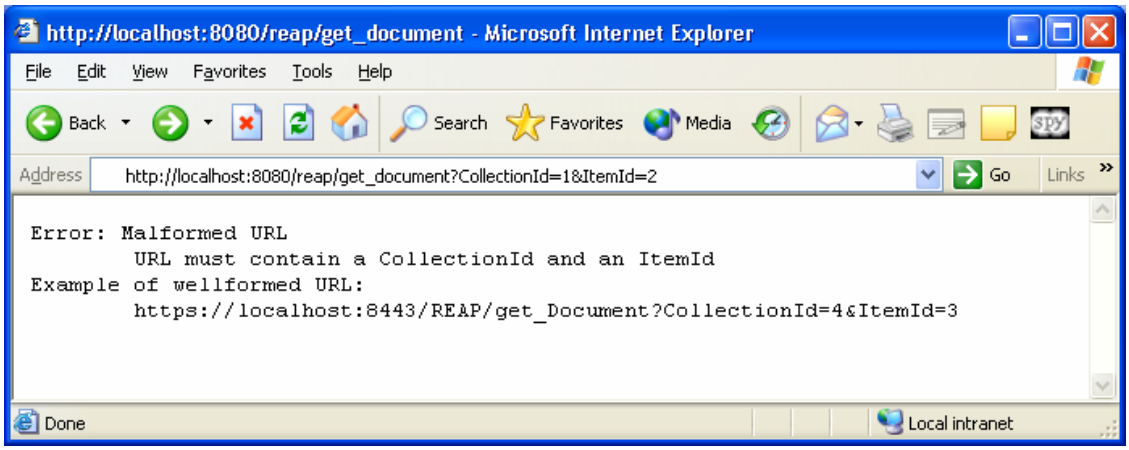
Figur 10-20 Eksekveringsstier gjennom REAP

10.4.5.2 Gjennomgang med skjermbilder og forklaringer

Denne gjennomgangen tar utgangspunkt i at brukeren har søkt etter informasjon, for eksempel gjennom ADEPT eller lignende systemer og har fått ut en URL som del av resultatet som skal brukes til å aksessere materialet. En slik url må da inneholde 2 parametre, nemlig collectionId og itemId. Eksempel på en slik url kan være http://localhost:8080/reap/get_document?CollectionId=1&ItemId=2

Steg 1:

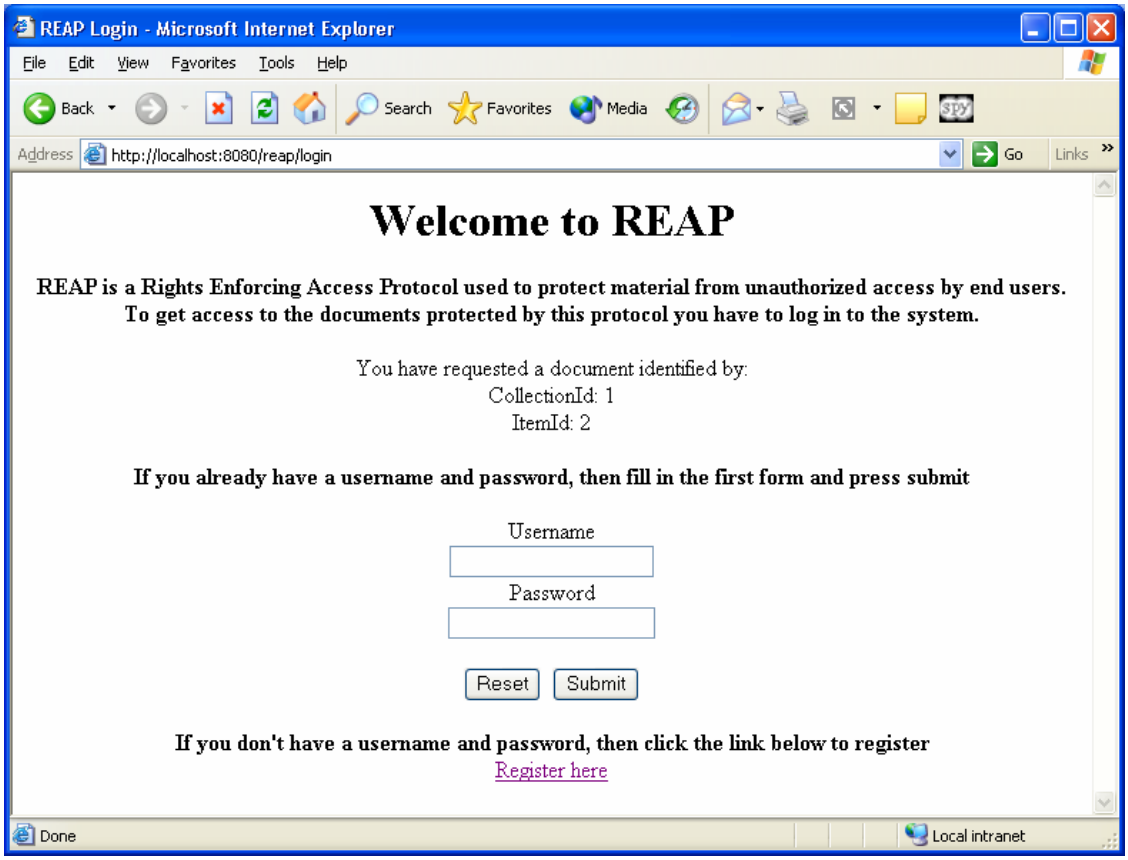
Ved å følge URL'en som brukeren fikk ut fra søkingen kontakter brukeren systemet og forteller at han er interessert i tilgang til et materiale som er identifisert av kombinasjonen av av CollectionId og ItemId. Hvis systemet blir kontaktet uten at forespørselen inneholder CollectionId og ItemId vil systemet vise frem en feilmelding:



Figur 10-21 Skjermbilder. Steg 1: Malformed URL

Steg 2:

Hvis brukeren har brukt en url som inneholder en gyldig url vil han, hvis han ikke allerede er logget inn bli presentert for en innloggingsside:

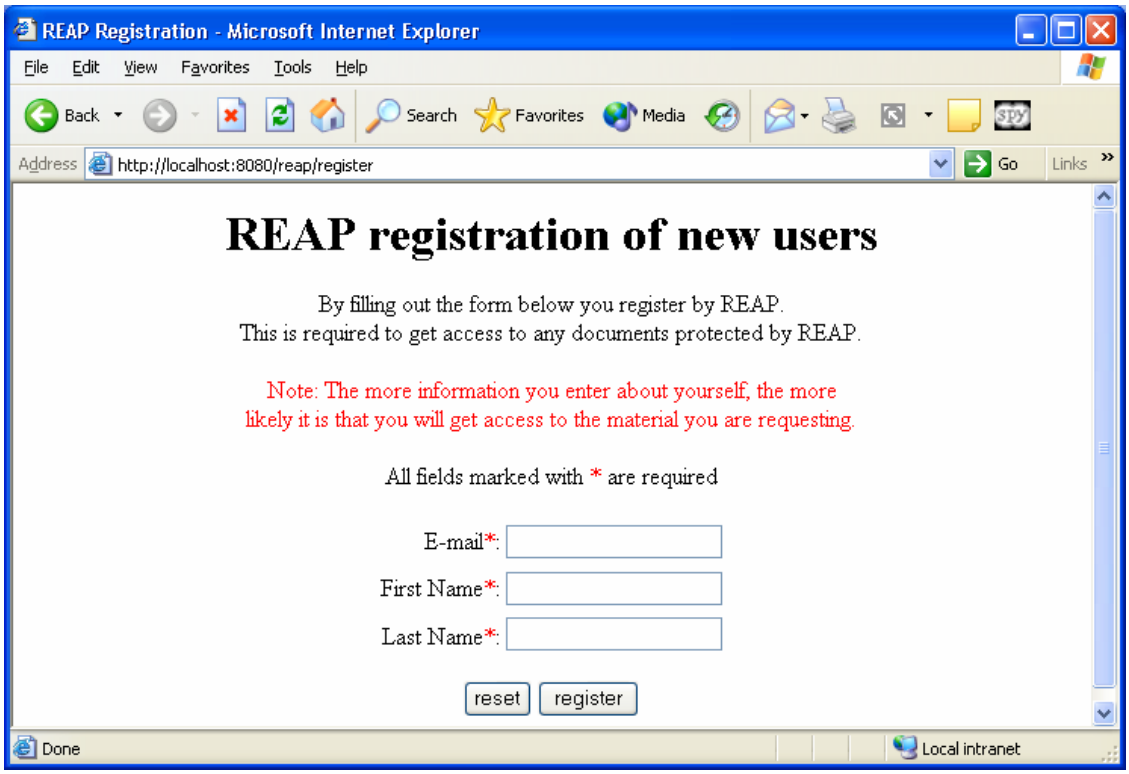


Figur 10-22 Skjermbilder. Steg 2: Innloggingsside

Hvis brukeren har brukernavn og passord kan han logge inn, og gå videre. Hvis han ikke har brukernavn og passord må han først registrere seg med REAP:

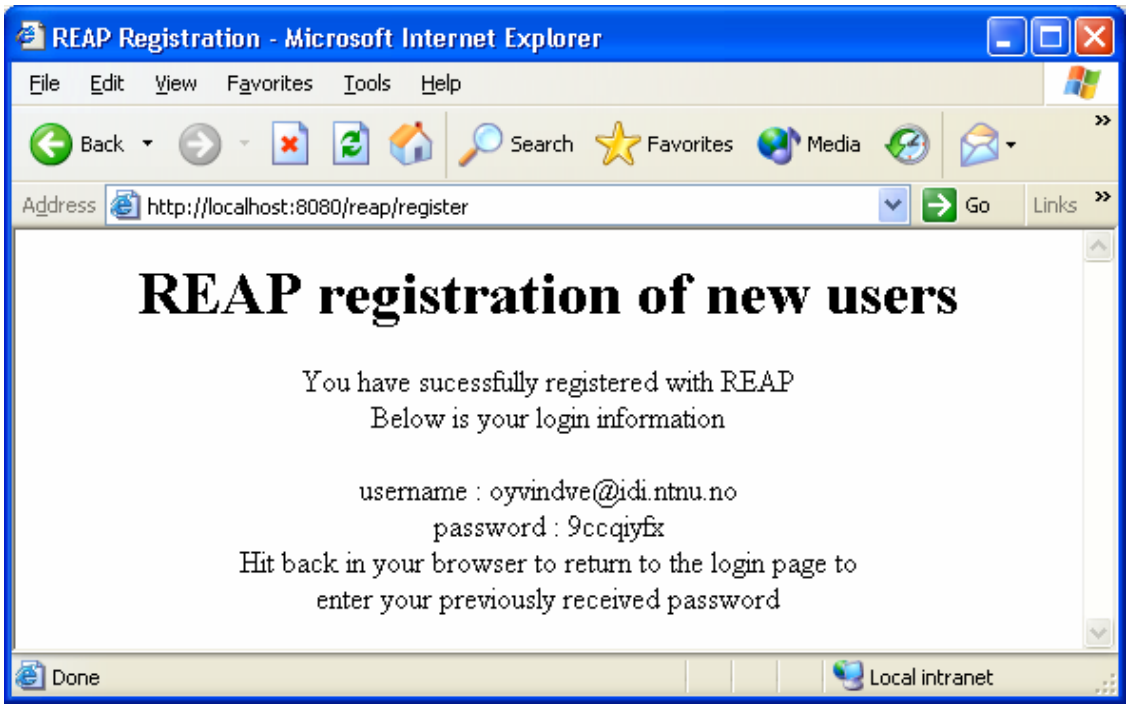
Steg 3:

Hvis brukeren ikke har brukernavn og passord må han registere seg og opprette en profil for sin bruker hos REAP først:



Figur 10-23 Skjermbilder. Steg 3 Registreringsside

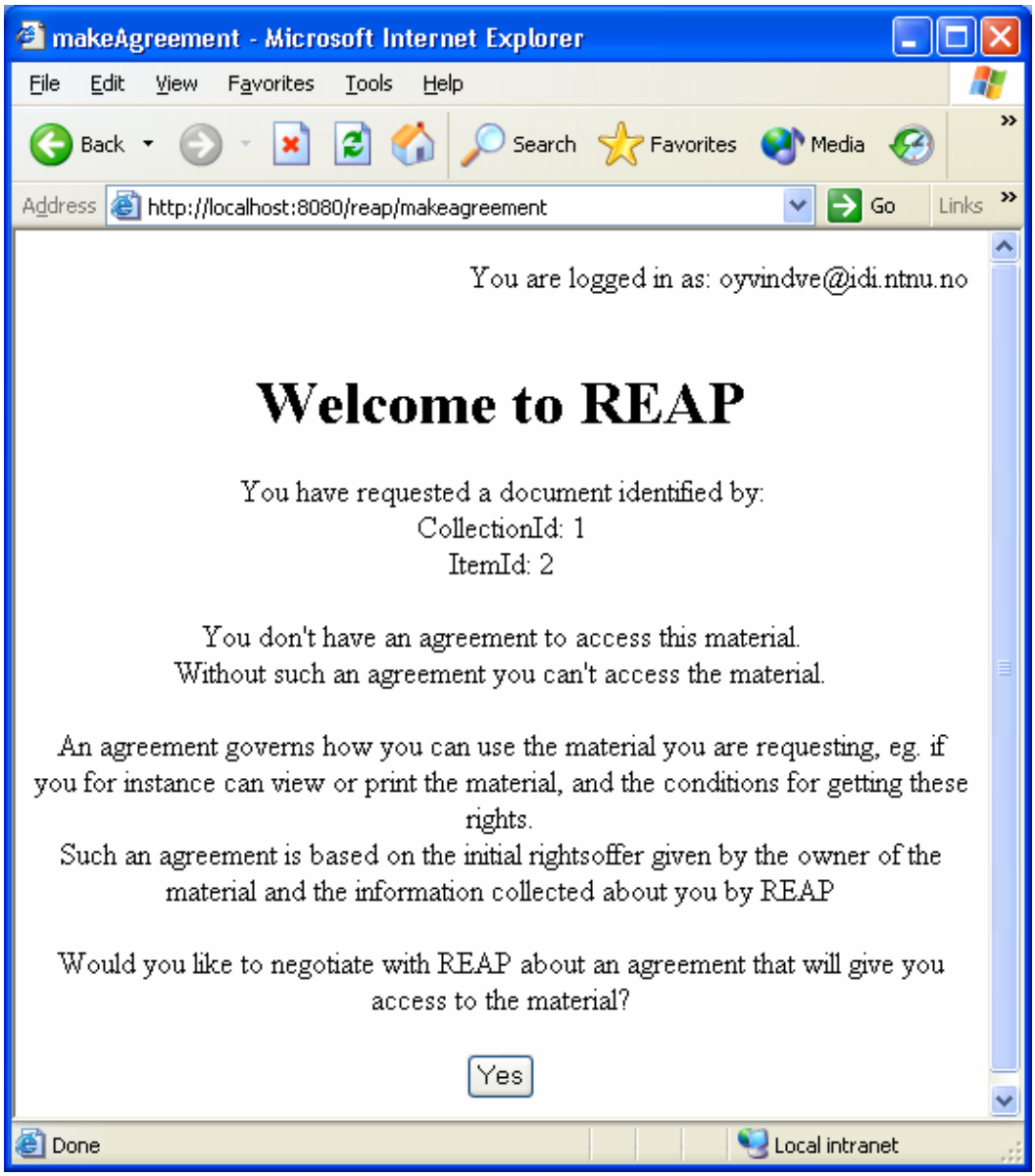
Når man registrerer seg genererer systemet et passord som består av en kombinasjon av store bokstaver og små bokstaver og tall. Dette passordet lagres i brukerens profil sammen med de andre opplysningene om brukeren i en database. Denne informasjonen presenteres så til brukeren slik at brukeren kan gå tilbake og logge seg inn:



Figur 10-24 Skjermbilder. Steg 3: Presentasjon av registrert brukernavn og passord

Steg 4:

Etter at brukeren har logget seg inn med sitt brukernavn og passord presenteres brukeren hvis han ikke allerede har en avtale om adgang til dette materialet en side som informerer om at han ikke har en avtale og systemet spør om han vil lage en slik avtale.



Figur 10-25 Skjermbilder. Steg 3: Systemet spør om brukeren vil fremforhandle en avtale/agreement

Steg 5

Hvis brukeren velger å lage en avtale viser systemet frem en side som presenterer rettighetsholderens tilbud om overføring av rettigheter der brukeren kan velge hvilke requirements han vil oppfylle: Fordi dette skjermbildet er veldig langt og ikke lar seg kopiere til oppgaven direkte, har jeg gjengitt innholdet og bare laget bilder som antyder øvre og nedre del av vinduet.

You are logged in as: oyvindve@idi.ntnu.no

Welcome to REAP

You have requested a document identified by:
CollectionId: 1
ItemId: 2

This is the offer from the publisher for how you can access the material identified by CollectionId 1and ItemId 2

If you want to make use of the offer you have to make an agreement by choosing which requirements you are willing to fullfill. Together with the information in your profile and the nature of your Internet connection, this will govern how you can use this material.

Information about the offer:
date:2002-08-03
Information about the resource
MIMEType:Image/jpg
name:Winter.jpg
uid (idscheme=ISBN):alkdjfaøkljdf

Global constraints
These constraints applies to all permissions defined below
Any individual permission can be excercised maximum 10 times if not further restricted under an individual permission
Global requirements
To get access to any of the permissions defined below these requirements all have to be fullfilled
To get access to any permissions you have to pay NOK 400.
☐ Yes, I want access to permissions.

Permissions offered:
Display
Constraints for the Display permission
No constraints defined
Requirements for the Display permission
To get access to this permission you have to pay NOK 200.
☐ Yes, I want access to this permission.
Print
Execute

Reset | Make agreement

Figur 10-26 Skjermbilder. Steg 5. Systemet presenterer rettighetsholderens offer for brukeren

Nå kan brukeren velge hvilke requirements han vil oppfylle og så trykke knappen merket "Make agreement"

Steg 6

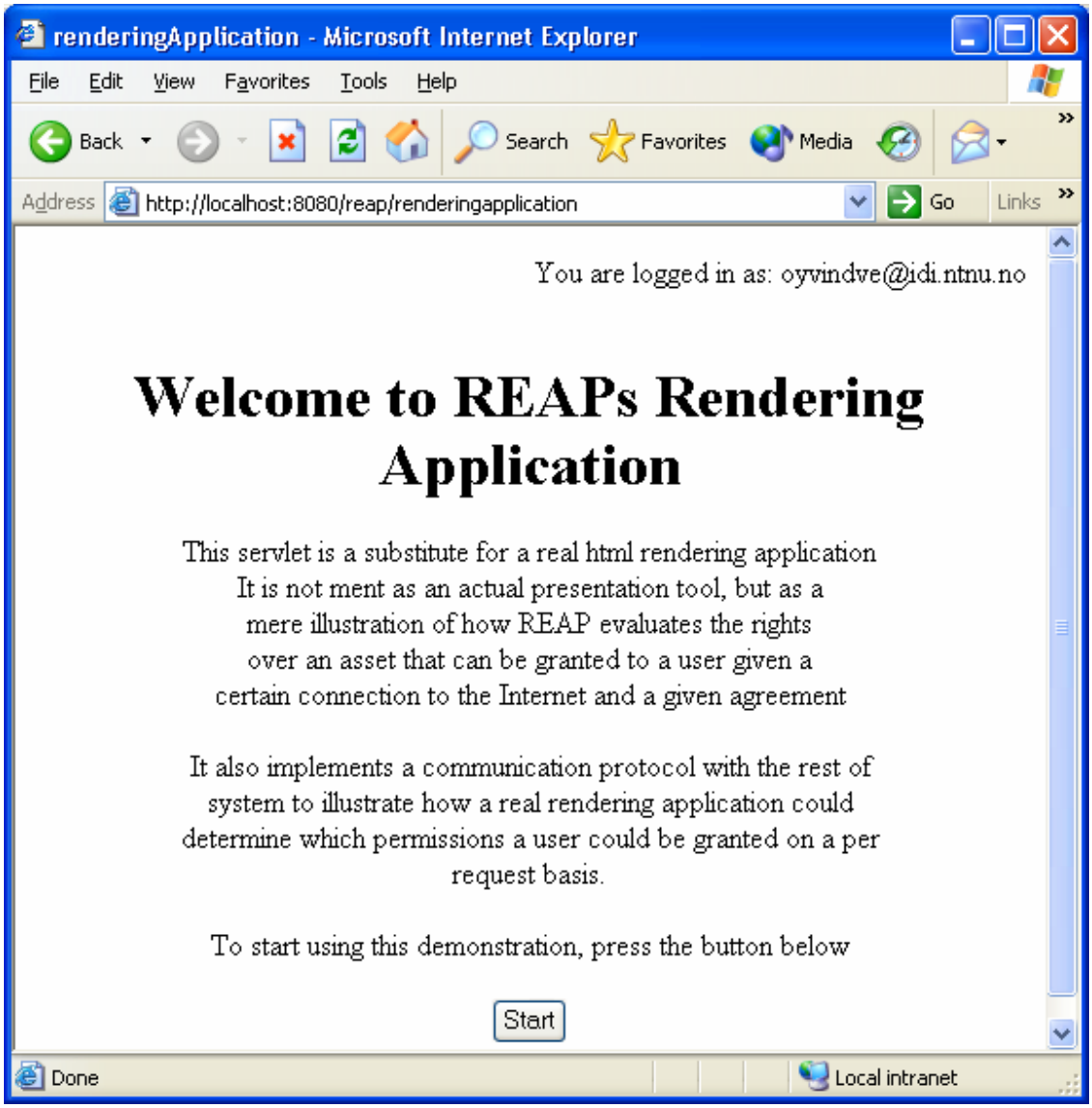
Brukeren presenteres nå for en side som i denne applikasjonen representerer en betalingsside der brukeren blir bedt om å gjennomføre en betalingstransasksjon som tilsvarer de requirements som brukeren krysset av for i forrige side:

Figur 10-27 Skjermbilder. Steg 6. Systemet viser en betalingsside slik at brukeren kan betale for rettighetene han han vil ha tilgang til.

I realiteten trenger ikke brukeren skrive inn et kontonummer og systemet krever bare at han trykker på knappen merket ”Pay and Generate Agreement”. I denne prototypen regnes et trykk på denne knappen som om brukeren har betalt det avtalte beløpet.

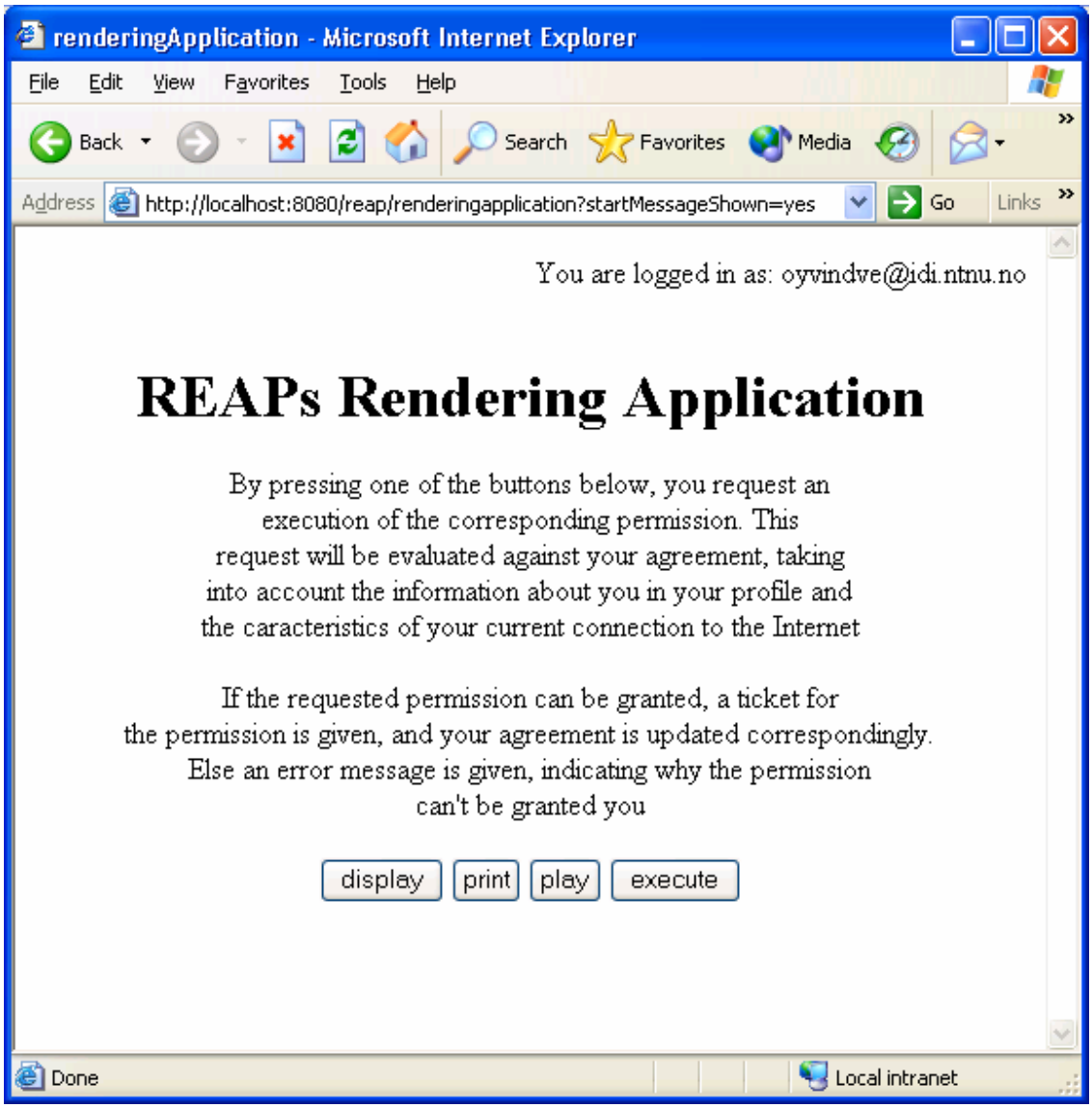
Steg 7.

Brukeren har nå en avtale, er logget inn og kan begynne å aksessere materialet. Det gjør at Systemet kan åpne en applikasjon som brukeren kan bruke for å aksessere rettigheter over materiale/forespørre tickets:



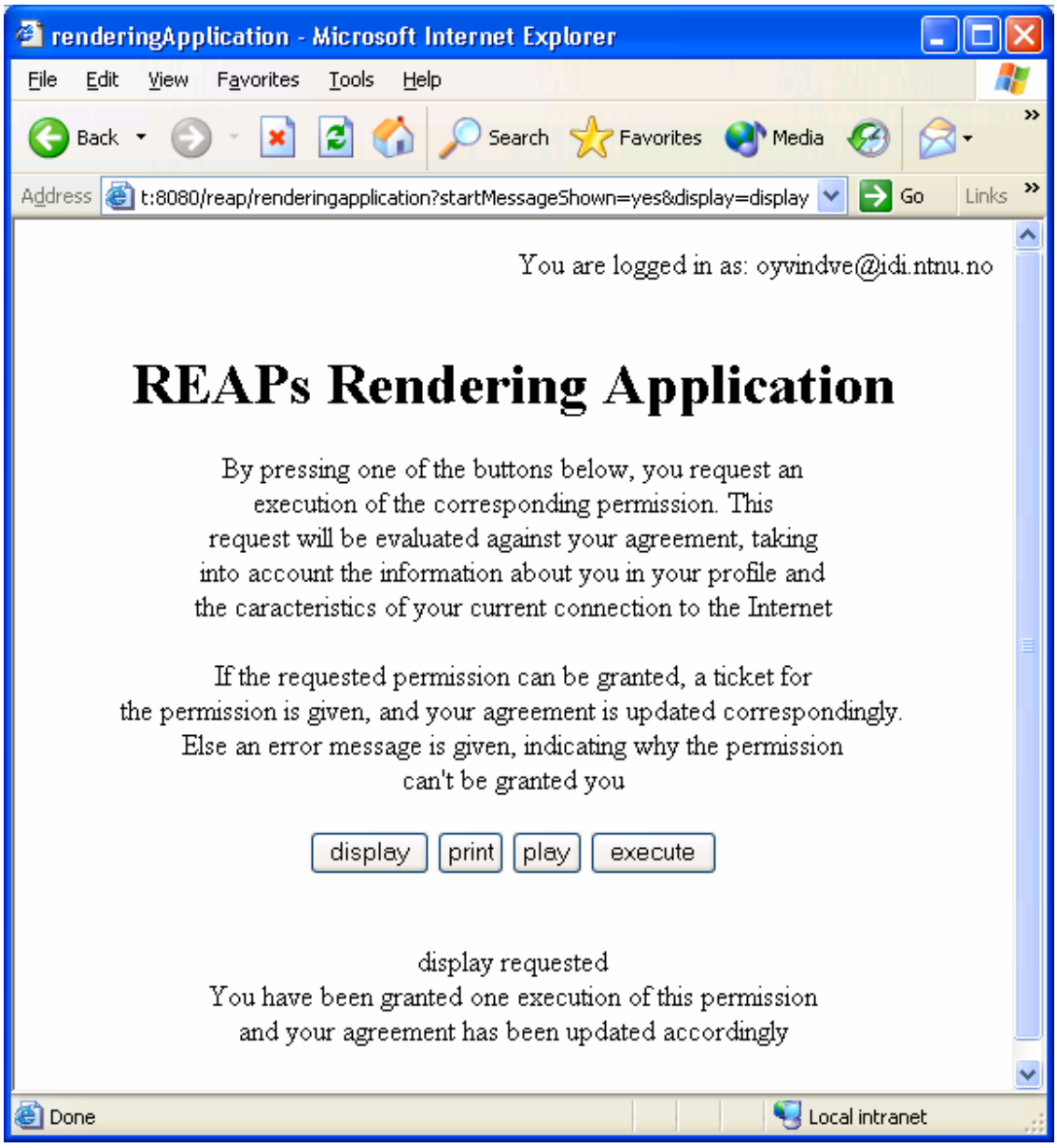
Figur 10-28 Skjermbilder. Steg 7. Systemet viser fram en velkomstmelding for fremvisningsapplikasjonen

Systemet inneholder ingen funksjonalitet for å la brukeren faktisk utføre de handlinger han ønsker. Et trykk på en av knappene i skjermbildet under fungerer som en forespørsel om en ticket til å utføre handlingen som knappen antyder og systemet vil da gi melding tilbake om denne ticket kunne gis eller ikke og en feilmelding som sier hvorfor ikke hvis brukeren ikke har tillatelse til å aksessere denne funksjonaliteten ut fra sin avtale. I en virkelig fremvisningsapplikasjon ville applikasjonen i stedet for å vise fram en melding om at du har blitt tildelt retten til å utføre handlingen en faktisk utført handlingen.



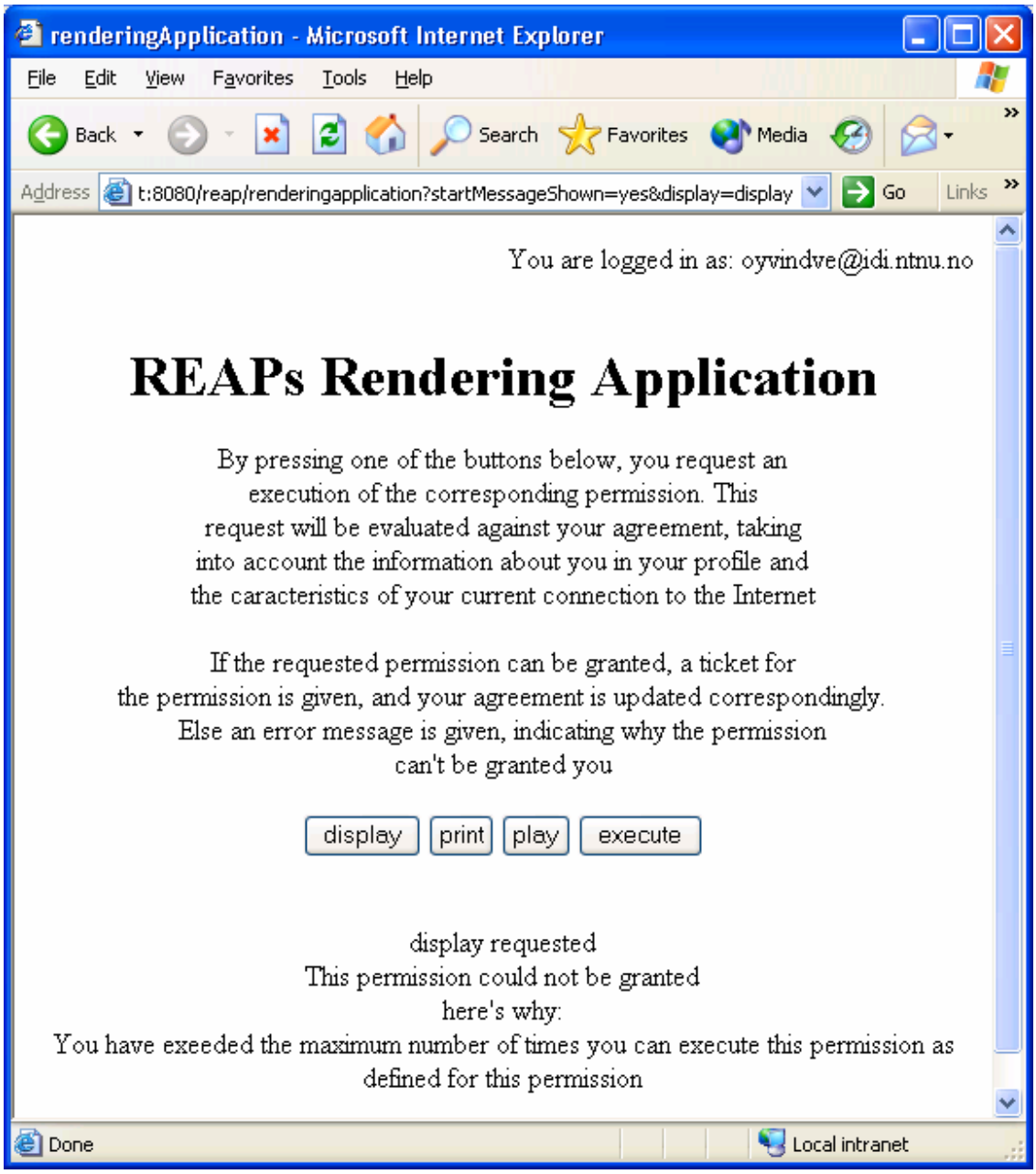
Figur 10-29 Skjermbilder. Steg 7: Fremvisningsapplikasjonens grensesnitt

Ifølge den avtalen brukeren akkurat har generert, kan han for eksempel aksessere permission display 10 ganger. De ti første gangene brukeren trykker på denne knappen vil han da få ut følgende som er et substitutt for å faktisk vise fram innholdet i fila brukeren forespør:



Figur 10-30 Eksempel på resultat av vellykket forespørsel om ticket/tilgang til en rettighet.

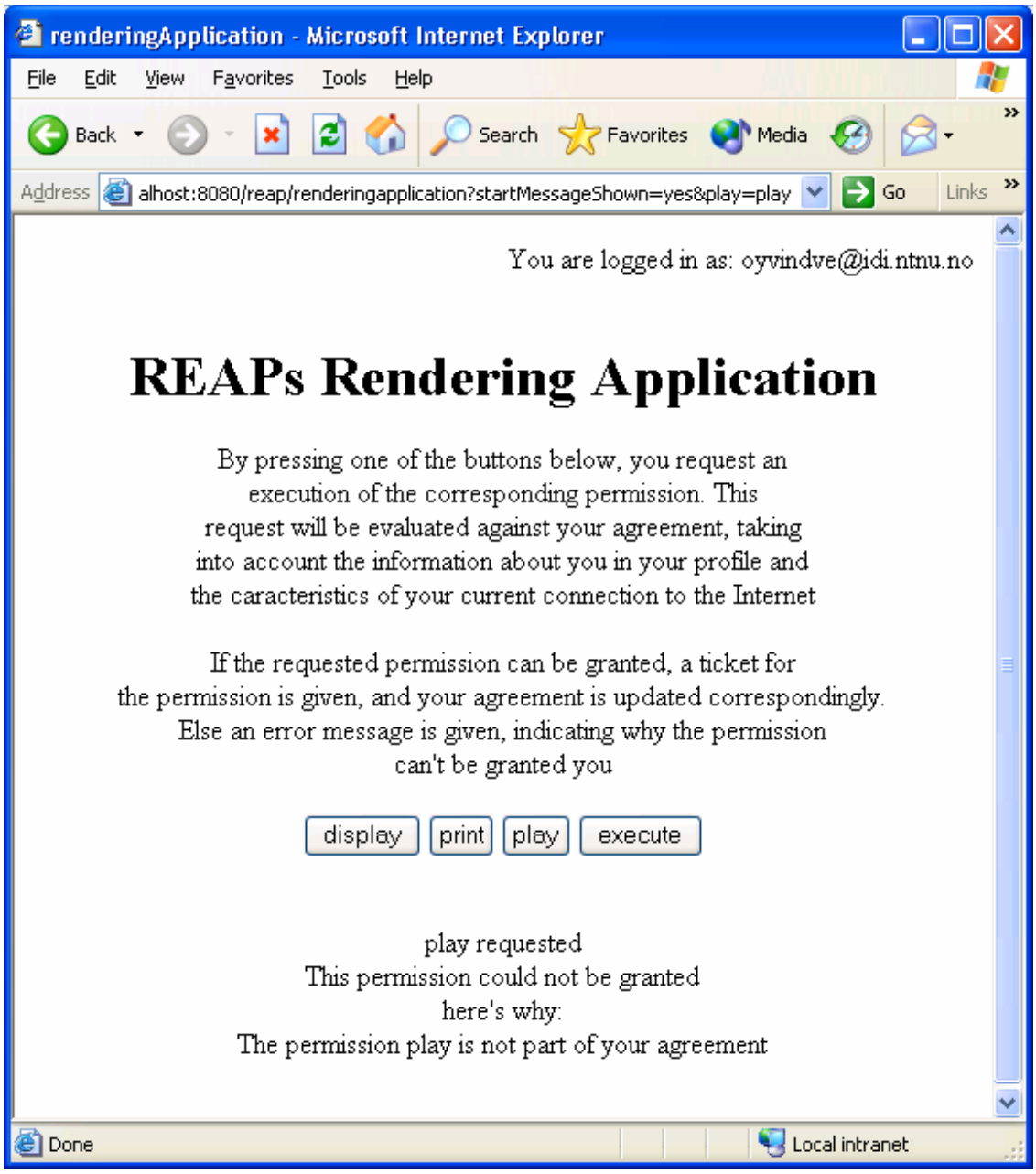
Hvis brukeren prøver å aksessere denne tillatelsen flere ganger enn det som er avtalt i avtalen til brukeren vil systemet utstede en ticket som sier at tillatelsen ikke kan aksesseres og brukeren vil få ut dette skjermbildet



Figur 10-31 Skjermbilder. Steg 7. Eksempel på ikke tildelt ticket/adgang til rettighet.

.

Hvis brukeren for eksempel prøver å aksessere en rettighet han ikke har tilgang til iflg sin avtale vil han få ut et skjermbilde som informerer om dette:



Figur 10-32 Skjermbilder. Eksempel på svar på systemet når brukeren forsøker å aksessere en rettighet han ikke har tilgang til.

Selv om brukerapplikasjonen ikke kan utføre handlinger som å vise fram materiale, skrive ut materiale osv, gir den et godt bilde av når og under hvilke forutsetninger en bruker ville blitt gitt tilgang til å utføre handlinger. Derfor er denne applikasjonen tilstrekkelig for å vise at slike systemer faktisk kan fungere i som adgangsprotokoll for digitale bibliotek.

10.5 Implementasjon

10.5.1 Redegjørelse for viktig teknologi brukt i implementasjonen

Java er brukt som programmeringsspråk for systemet som helhet. All bakenforliggende funksjonalitet for REAP er skrevet i dette språket. Som grensesnitt mot brukeren er det brukt Java Servlets. Servlets er Javas svar på CGI.(Common Gateway Interface). Denne teknologien regnes for å være mer effektivt enn CGI, og kan lettere kommunisere med et bakenforliggende system, særlig hvis dette er implementert i java.

Rettighetsspråket for REAP er definert i et XML Schema **[XMLSchema]**. Offers agreements og userprofiles lagres som XML dokumenter **[XML_Rec]**.

Selve applikasjonen som utgjør REAP er utviklet som en såkalt web-applikasjon. Dette betyr at all kode pakkes inn i en spesiell pakkestruktur og komprimeres i en enkelt fil av formatet war. Hele systemet kan distribueres ved å distribuere denne fila. Webtjenere pakker ut og installerer slike filer samt konfigurerer seg selv automatisk ved oppstart/restart. Databaser som brukes er ikke del av denne warfila, og må installeres utenom. Se for øvrig kapittel 9.5.3 under.

10.5.2 Redegjørelse for viktige tekniske komponenter i prototypen

Som webserverløsning brukes Jakarta Tomcat fra Apache Software Foundation. Jakarta Tomcat versjon 4 er den offisielle referanseimplementasjonen for webservere som støtter Java Servlets og JavaServer Pages. **[JakartaTomcat]**. Men siden prototypen er laget som en web-app, kan man bruke en hvilken som helst webserver som støtter Java Servlets og JSP.

Som lagringsmedium for XML dokumenter brukes xml-databasen ”Apache Xindice” fra Apache Software Foundation. **[Xindice]**
Da REAP er en prototype for å teste styring av rettigheter har databaser og lagringsmedia for digitale ressurser vært et perifert aspekt ved oppgaven. Likevel er det nødvendig å ha noen få ressurser som det kan testes på. Derfor har det vært lagt inn noen få bilder og enkle filer i systemet slik at systemet har kunnet gjenhente disse. Lagring av disse filene har vært gjort i et vanlig filsystem på maskina som REAP kjører på. Systemet er laget slik at kun en liten del av systemet må reprogrammeres dersom det skal brukes andre datalagringsalternativer for systemet.

10.5.3 Databasegrensesnitt i programmvaren

Klassene som aksesserer databasene i programvaren er spesifikke for den databaseløsningen som er valgt. Koden kobler seg opp mot databasene som prototypen

bruker. Hvis man senere vil bruke andre databaser må denne koden skrives på nytt. Evt nye implementasjoner av disse klassene må da implementere alle metoder som kalles fra andre klasser i systemet støttes.

Xml databasen (Xindice) lagrer opplysninger om registrerte brukere, rettighetstilbud ang et materiale og avtaler ang bruk av et material. Samlingene i databasen er ordnet hierrarkisk. Top-level samlingen i databasen er ”db” og under denne er det lagt til en samling som heter ”REAP” som inneholder alle subcollections som brukes av REAP. Samlingene som brukes av prototypen kan da refereres til slik:

/db/REAP/Offers

Lagrer xml dokumenter som uttrykker tilbud om rettighetsoverføring som er knyttet til et materiale. Alle dokumenter lagres her med en nøkkel a la ”colId 1 itemId 2”. Det er viktig at alle som publiserer materialet lagrer tilbudet for materialet i henhold til dette slik at systemet kan finne de igjen.

/db/REAP/Agreements

Lagrer xml dokumenter som uttrykker rettigheter over et materiale som er gitt en bruker i form av en avtale. Systemet vil generere en nøkkel a la ”user oyvindve colId 1 itemId 2”og lagre dokumentet under denne nøkkelen

/db/REAP/Profiles

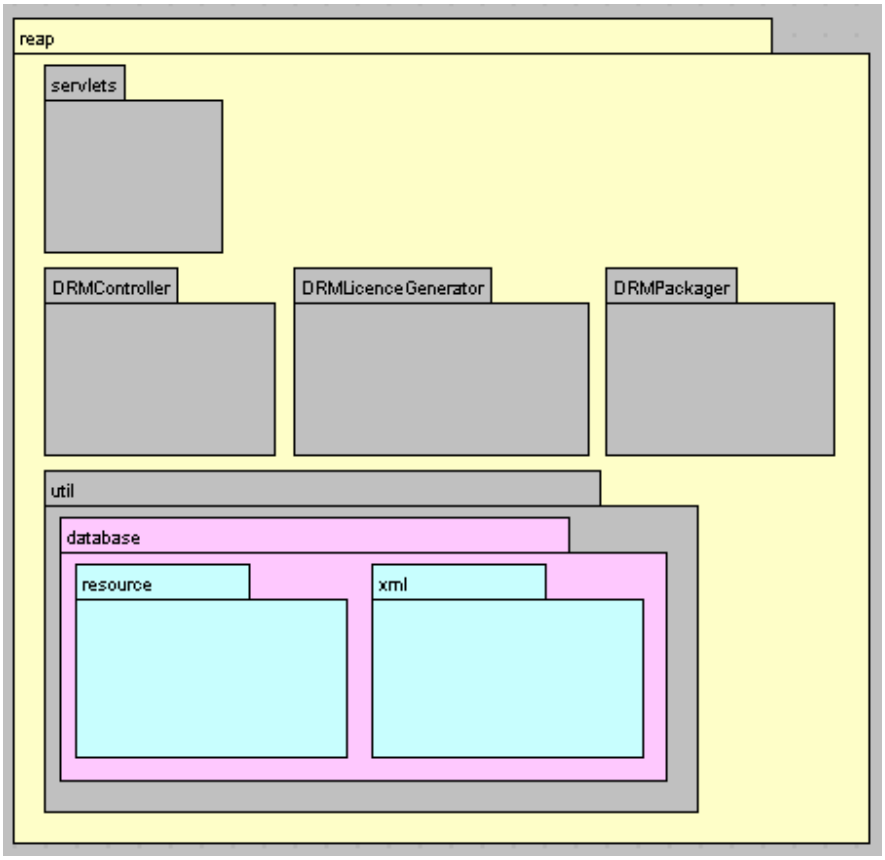
Lagrer alle opplysninger en bruker oppga om seg selv da han registrerte seg. Systemet lagrer brukerens profile under en nøkkel a la ”oyvindve@idi.ntnu.no”

Hvis databasen byttes ut må klassesn xmlDatabaseFrontEnd.java reprogrammeres da denne bruker stiene angitt over for å finne riktige dokumenter for offers, agreements og profiles under eksekvering.

”Databasen” for materiale i denne prototypen består kun av filer lagt inn i et unix filsystem. Systemet bruker en tredimensjonell tabell for å oversette fra kombinasjonen av collectionId og itemId (reapId) til et filnavn/sti. Tabellen har tre kolonner; collectionId, itemId og filepath og hver rekke representerer en ressurs. Tabellen befinner seg i en tabseparert tekstfil som systemet gjør et sekvensielt søk i hver gang det kommer inn en forespørsel til systemet. Den tabseparerte fila kan åpnes i et hvilket som helst regnearkprogram for inspeksjon og oppdatering.

10.5.4 Organisasjon av kildekoden

Kildekoden for prototypen REAP er ordnet i pakker etter type funksjonalitet. Denne pakkestrukturen er grovt sett delt inn i en trelagsarkitektur som består av et grensesnitt (servlets), et mellomlag (Alle pakker som begynner med DRM..) og et informasjonslag i bunn. Noen klasser som egentlig skulle vært del av mellomlaget er for enkelhets skyld lagt inn i pakken reap.util. Under er beskrevet hvilken funksjonalitet som ligger i hver av pakkene



Figur 10-33 Organisering av kildekoden. Pakkestruktur.

All kode for REAP ligger i pakken reap. Denne pakken har 5 subpakker:

- reap.servlets: Inneholder servlets som brukeren interaksjonerer med når han bruker systemet pluss en klasse som fungerer som et substitutt for en framvisningsapplikasjon som brukes til å vise fram materialet, la brukeren skrive ut osv. Denne applikasjonen lar ikke brukeren utføre slike handlinger uten at han har rettigheter til å gjøre det.
- reap.DRMController: Inneholder kun en klasse (DRMController.java). I dette systemet fungerer denne klassen rett og slett som en formidler mellom servlets og resten av systemet
- reap.DRMLicenceGenerator: Inneholder kun en klasse (DRMLicenceGenerator.java). Denne klassen har som hovedansvar å generere avtaler for brukeren og evaluere om brukeren kan aksessere en gitt rettighet når brukeren forespør adgang til denne rettigheten
- reap.DRMPackager: Inneholder kun en klasse (DRMPackager.java) Denne klassen er ansvarlig for å forberede ressursen for levering til klienten.
- reap.utilities: Inneholder klassen DRMPackage som er en representasjon av en DRMPackage. Inneholder klassen FileEncapsulator som brukes for å sende

innholdet av filer og litt metadata om filer som parametre i metodekall mellom klasser. Inneholder klassen UsernamePwdHolder som brukes for å sende brukernavn og passord som parametre i metodekall mellom klasser. Inneholder klassen profilehandler som er ansvarlig for å generere profiler for brukere og returnere et passord som brukeren kan logge på systemet med. Inneholder en pakke som i sin tur inneholder all databasefunksjonalitet

- reap.util.database: Inneholder 2 pakker som deler databasefunksjonaliteten mellom seg.
    - reap.util.database.resource: Denne pakken inneholder funksjonalitet for å gjenhente materialet som brukeren ønsker tilgang til.
    - reap.util.database.xml: Denne pakken innholder funksjonalitet for å lagre og gjenhente offers, agreements og brukerprofiler fra databasen Xindice.

For en detaljert beskrivelse av de enkelte klasser som befinner seg i pakkene, se Javadoc dokumentasjonen som er vedlagt i appendix D.

11 Evaluering / Testing

11.1 Hva er Evaluering og testing

Evaluering og testing er ofte den vanskeligste prosessen ved utviklingen av ny programmvare. Evaluering kan defineres som å fastsette hvor mye av den ønskede funksjonalitet som er kommet med ut fra den ønskede funksjonalitet og om denne funksjonaliteten fungerer som forventet. Man kan også ta med at evaluering dreier seg om å fastsette om den ønskede funksjonalitet virkelig er den man ville ha. Testing er prosessen med å komme fram til en slik evaluering.

Testing kan deles inn i ”White Box Testing” og ”Black Box Testing”. White Box Testing verifiserer syntaksen og strukturen i programvaren og krever adgang til full kildekode. Kildeinspeksjon, debugging og gjennomgang av de forskjellige eksekveringsruter i programvaren er eksempler på slik testing. Black Box Testing tester observerbar oppførsel til programvaren som ikke krever adgang til kildekoden, dvs hvordan programvaren reagerer på diverse input og produserer output på grunnlag av det.

Man kan teste forskjellige aspekter ved en programvare. **Enhetstesting** betyr å teste en viss del av programvaren uavhengig av resten av programvaren. Dette kan være nyttig for å isolere feil som det ellers kan være vanskelig å lokalisere. **Integrasjonstesting** betyr å teste kommunikasjonskanaler, linker og data deling mellom komponenter eller applikasjoner. **Systemtesting** betyr å teste hele systemet etter at alle delene er satt sammen. **Ytelsestesting/stresstesting** betyr å teste ytelsen av et program for å finne ut når og under hvilke forutsetninger belastningen på systemet blir så stor at systemet bryter sammen eller genererer feil output. **Regresjonstesting** betyr å teste at man ikke har innført nye feil i programvaren når man har forandret programvaren. **Quality assurance testing** er ikke så mye en test av programvaren selv, men heller en test av at

programvaren er utviklet i overenstemmelse med standarder for kodegenerering, generering av dokumentasjon og generering av spesifikasjoner.

En god testing forutsetter at alle disse aspektene testes i de faser der de er relevante i utviklingsprosessen. For noen tester er det greiest å bruke blackbox testing, mens i andre tilfeller kreves det at man bruker testing som evaluerer ”innmaten” i programvaren som er under utvikling. En god prosess for å utvikle programvare har en skriftlig bindende plan eller paradigme for testing for å kunne utvikle god programvare.

11.2 Tester av systemet

11.2.1 Hva omfattes av testene, og testsamlinger

Man kan si at det er tre aspekter som må testes når det gjelder REAP. Det ene er at agreements genereres riktig basert på et offer og input fra brukeren. Det andre er at tickets genereres på en måte som er konsekvent med avtalen som ble inngått. Hvis disse to delene av systemet fungerer som forutsatt vil REAP være konsekvent i forhold til forretningslogikken som rettighetsspråket for REAP definerer. Det tredje er at systemet skriver en riktig rapport til logfila hver gang det inngås en avtale. Det anses ikke som viktig å teste andre aspekter ved REAP som system fordi hensikten med denne prototypen nettopp er å vise at systemet kan brukes til å håndheve en forretningslogikk som fastlagt i et rettighetsspråk.

For å teste disse aspektene har jeg satt opp en samling av testdokumenter i form av rettighetstilbud (Offers). Disse er satt opp spesifikt for testing og trenger ikke nødvendigvis være spesielt semantisk logiske, men de gjengir de aller fleste viktige kombinasjoner av rettighetsuttrykk som kan gis som input til prosessen med å generere agreements. For alle disse tilbudene har jeg forsøkt å generere agreements vha systemet slik at man kan sammenholde de genererte agreements med det retttighetstilbud de bygger på og de inndata brukeren har gitt.

11.2.2 Testscenarioer

Når man tester på om overgangen fra offers til agreements skjer på riktig måte er det greit å vite, selv om selve testingen er av såkalt blackbox testing, at dannelsen av agreements skjer ved at enkelte deler av den genererte avtalen kopieres direkte fra det korresponderende offer, mens andre deler genereres på bakgrunn av det gitte offer, opplysninger som brukeren gir inn om seg selv og hvilke requirements han vil tilfredsstille. Når det dannes en agreement vil det være mest interessant å kikke på permission delen av en avtale fordi både context-, asset-, og partydelene av avtalen kun rekonstrueres fra opplysninger i ressursens offer. (Med unntak av at brukeren som inngår avtalen føyes inn i partydelen av avtalen som part i avtalen. Elementet ”user” representerer brukeren som har inngått avtalen.) Når det skal dannes en avtale

determineres det hvilke forpliktelser brukeren har tilfredsstilt, og de rettigheter brukeren skal ha tilgang til ut fra dette tas med i avtalen. Alle beskrivelser av forutsetninger lagres ikke i avtalen fordi de kun blir evaluert i det avtalen blir inngått. Alle constraints eller begrensniger som er lagt på utøvelsen av rettigheter kopieres direkte til avtalen som er under generering, fordi disse må evalueres i request-time, altså i det brukeren sender en forespørsel om å utøve en rettighet over ressursen i forhold til sin avtale. Grunnen til at disse begrensningene ikke evalueres når avtalen inngås er at forutsetninger som type internettilknytning eller dato kan være en annen når brukeren inngår en avtale enn når brukeren senere vil eksekvere en rettighet i forhold til sin avtale.

For genereringen av avtaler kan man derfor sette opp følgende tester:

1. Test på om alle opplysninger som skal kopieres fra et offer til en agreement overføres riktig og i sin helhet.
2. Test på om systemet genererer en riktig rapport til logfila når systemet genererer en avtale mellom en bruker og en rettighetsholder
3. Tester som sjekker om riktige rettigheter genereres i avtalen basert på det bakenforliggende offer og opplysninger som brukeren gir om seg selv. Dette kan deles inn i
    a. Gitt at det finnes en global requirement/forutsetning som må oppfylles for å få tilgang til rettigheter over ressursen. Nektes brukeren tilgang til rettigheter dersom forutsetningen ikke oppfylles??
    b. Gitt at det finnes en lokal requirement for en rettighet. Nektes brukeren tilgang til rettigheten dersom den lokale requirement ikke oppfylles?? Denne testen må gjennomføres for alle rettigheter i et offer.

    Når brukeren har generert en avtale kan han ved hjelp av en klient eksekvere rettighetene han er blitt gitt gjennom sin avtale. Før en slik klient kan utføre handlingen brukeren ønsker, må den få en ticket for handlingen fra systemet. Før det utstedes tickets skal constraints evalueres for å se om brukeren kan eksekvere en rettighet han er gitt tilgang til. Denne adgangen er gitt under betingelse av at constraints er tilfredsstilt i det øyeblikk brukeren forsøker å aksessere en rettighet i avtalen sin. Hvis alle constraints er tilfredsstilt kan rettigheten tildeles. Fordi det er 4 forskjellige constraints som kan defineres som del av en avtale, er det også fire tester som må gjennomføres for å sjekke at dette funger tilfredsstillende:

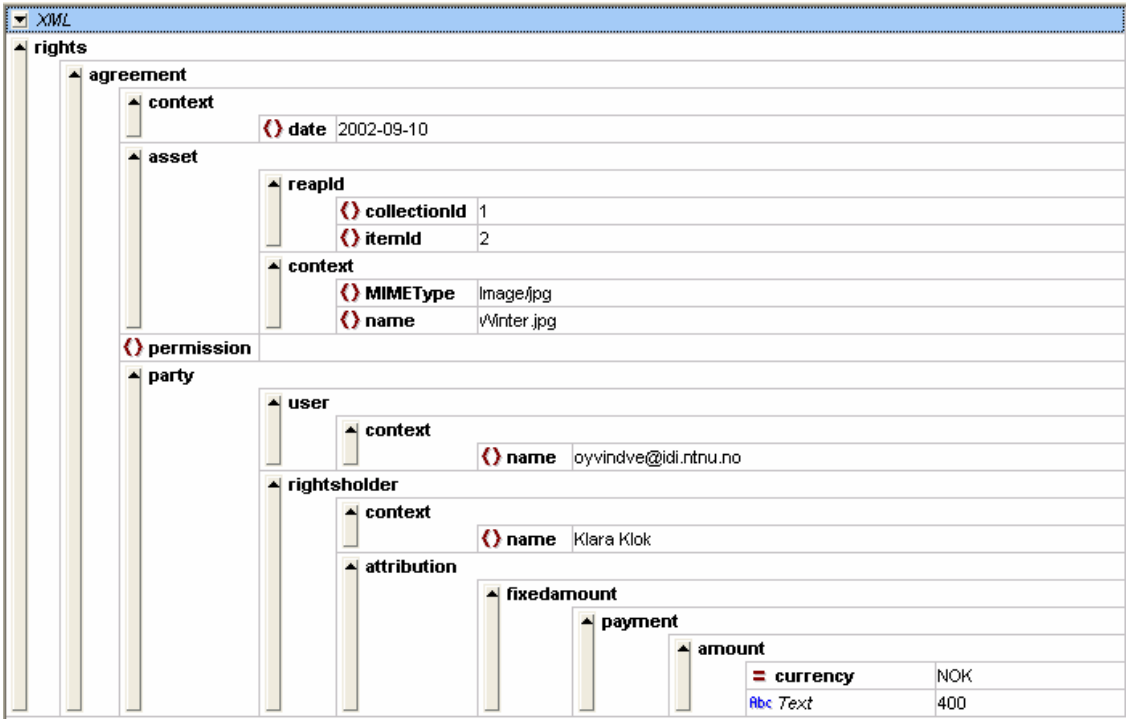
4. Test av at antall eksekveringer ikke overstiger max antall eksekveringer angitt av count. Nektes brukeren adgang til rettigheten dersom han allerede har aksessert rettigheten det antall ganger som er gitt i avtalen?
5. Test av at eksekveringen av rettigheten foregår innenfor det tidsrom som er satt som begrensning for eksekvering. Nektes brukeren adgang til en rettighet dersom systemklokka på maskina som systemet kjører på angir at tidspunktet for eksekvering av rettigheten ikke ligger innenfor det daterange som er satt som begrensening for utøvelse av rettigheten?

6. Test av at brukeren er den som rettigheten er begrenset til. Nektes brukeren tilgang til rettigheten dersom det registrert brukernavnet som brukeren er logget ikke stemmer overens med hva som er angitt av ”individual” i avtalen?
7. Test av om brukeren sitter tilkoblet det nettverk som er angitt av en nettverksmaske eller har en IP adresse som er innenfor det range av IP adresser som er angitt av startIP og endIP. Nektes brukeren adgang til rettigheten dersom hans netverkstilknytning ikke stemmer overens med den som er angitt som constraint for rettigheten?

11.2.3 Testresultater

Test 1:

Test på om alle opplysninger som skal kopieres fra et offer til en agreement overføres riktig og i sin helhet. Denne testen tar utgangspunkt i fila *basic_offer.xml* som er gjengitt i sin helhet i Appendix C. Den resulterende avtalen er gjengitt under.

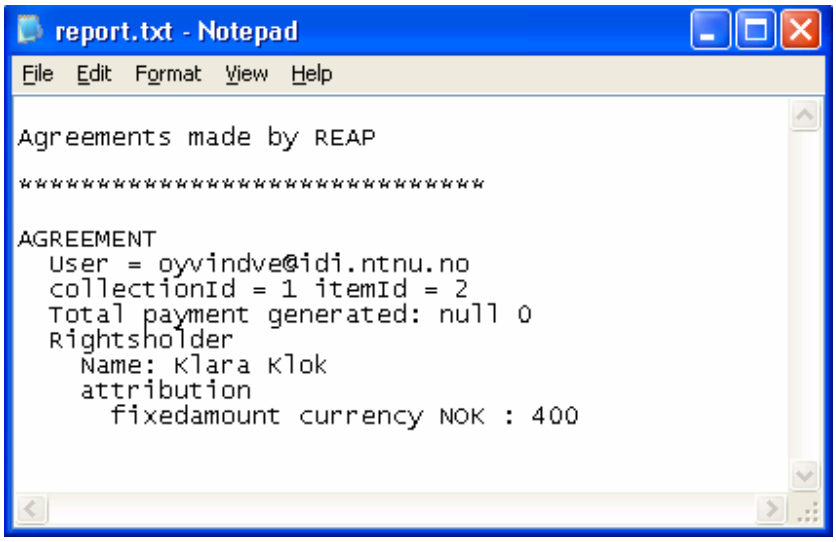


Figur 11-1 Testresultater. Test 1

Dette viser at alle elementer som kun skal gjenskapes i den form de har i det tilsvarende offer er riktig gjengitt i avtalen som genereres av systemet. Det viser også at systemet legger til et element user under elementet rightsholder som representer brukeren som har inngått avtalen som det skal.

Test 2:

Test på om systemet genererer en riktig rapport til logfila når systemet genererer en avtale mellom en bruker og en rettighetsholder
Under er gjengitt det entry som ble generert i logfila når avtalen over ble generert:



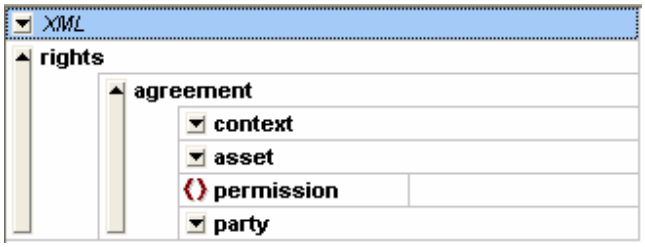
Figur 11-2 Testresultater. Test 2

Dette viser at det ble lagd en velformatert rapport i logfila da denne avtalen ble inngått. Likevel er det kanskje noe som det kan settes en finger på i dette log-entryet. Det inneholder en linje som sier
”Total payment generated: null 0”. Her skulle ”null” angitt en valutatype. Men siden det ikke ble generert noen betaling er heller ikke verdien for valutatype satt. Det hadde kanskje vært bedre om systemet ikke skrev ut teksten ”null”, men dette er tross alt ikke så viktig for logikken i systemet. Noen vil kanskje reagere på at selv om det ikke ble generert en betaling i forbindelse med transaksjonen angir loggen at rettighetshaveren skal tildeles 400 kroner. Dette har å gjøre med at rettighetsspråket for REAP (som alle språk forøvrig) kan lage uttrykk som ikke nødvendigvis gir mening. Ved at rettighetshavere tvinges til å bruke en applikasjon til å generere offers kunne man sikret at det bare ble generert uttrykk i språket som er logisk korrekte.

Test 3a:

Gitt at det finnes en global requirement/forutsetning som må oppfylles for å få tilgang til rettigheter over ressursen. Nektes brukeren tilgang til rettigheter dersom forutsetningen ikke oppfylles??
Denne testen tar utgangspunkt i fila *Test_Requirement_Offer.xml*. Som respons på at brukeren velger å ikke oppfylle den globale requirement om å betale 100 kroner for å få

tilgang til rettigheter skal systemet, hvis det virker slik det skal, generere en avtale som ikke inneholder noen rettigheter fordi oppfyllelse av globale requirements er en forutsetning for å få tilgang rettigheter i det hele tatt. Under er gjengitt avtalen som ble generert av systemet (Konstruksjonene under context asset og party-elementet er sammenfoldet):

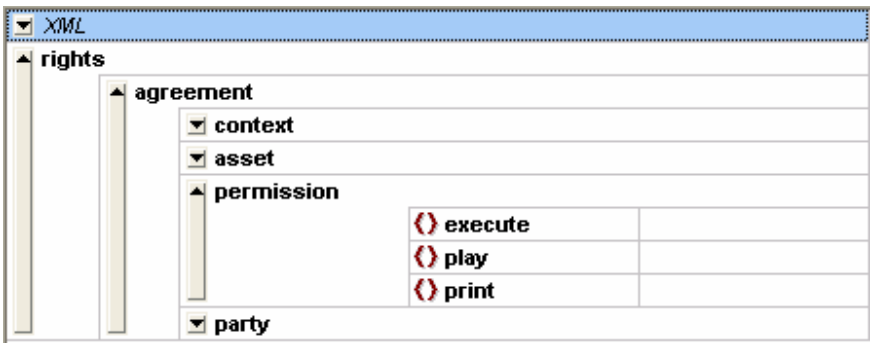


Figur 11-3 Testresultat. Test 3a

Her ser vi at permission elementet er tomt, dvs ikke har subelementer. Det betyr at som respons på at brukeren ikke har oppfylt globale forutsetninger gis han heller ikke tilgang til de enkelte rettighetene som kan gis i henhold til noen av de enkelte rettighetene.

Test 3b

Hvis brukeren faktisk oppfyller globale requirements/forutsetninger der de er gitt må han fortsatt oppfylle lokale requirements/forutsetninger for å få tilgang til de enkelte rettigheter. Gitt at det finnes en lokal requirement for rettigheten display. Nektes brukeren tilgang til rettigheten dersom den lokale requirement ikke oppfylles? Merk at denne testen er gjennomført en gang for alle fire mulige rettigheter, men fordi det er den samme kildekoden som utfører denne testen for alle rettigheter i systemet nøyer jeg meg med å gjengi resultatet for rettigheten display her. Denne testen tar utgangspunkt i fila ”*Test_Requirement_Offer.xml*”. Som respons på at brukeren velger å ikke oppfylle den lokale requirement om å betale 100 kroner for å få tilgang til display-rettigheten skal systemet, hvis det virker slik det skal, generere en avtale som ikke inneholder denne rettigheten. Under er gjengitt avtalen som ble generert av systemet når brukeren oppfylte requirements som måtte oppfylles for å få tilgang til rettigheter generelt og requirements som måtte oppfylles for å få tilgang til rettighetene execute, play og print, men *ikke* requirements for å få tilgang til rettigheten display.



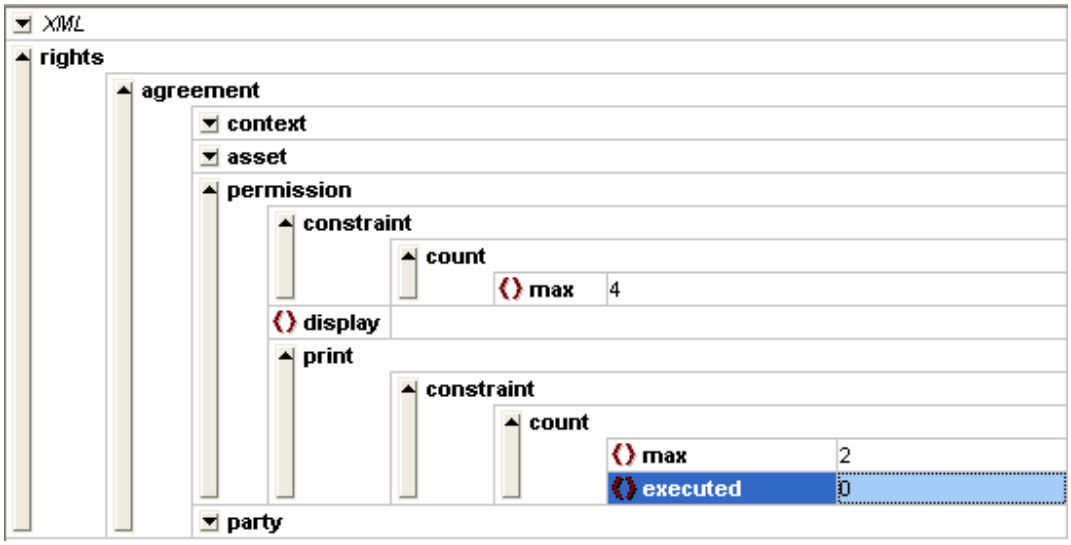
Figur 11-4 Testresultat. Test 3b.
Vi ser at som forventet inneholder avtalen rettighetene execute, play og print, men ikke rettigheten display.
Med denne testen avslutter vi testingen av genereringen av avtaler og at dette skjer i overenstemmelse med paradigmet som systemet bygger på. Vi kan nå gå over til å teste om systemet genererer Tickets som foutsatt.

Test 4

Test av at antall eksekveringer ikke overstiger max antall eksekveringer angitt av count. Nektes brukeren adgang til rettigheten dersom han allerede har aksessert rettigheten det antall ganger som er gitt i avtalen?

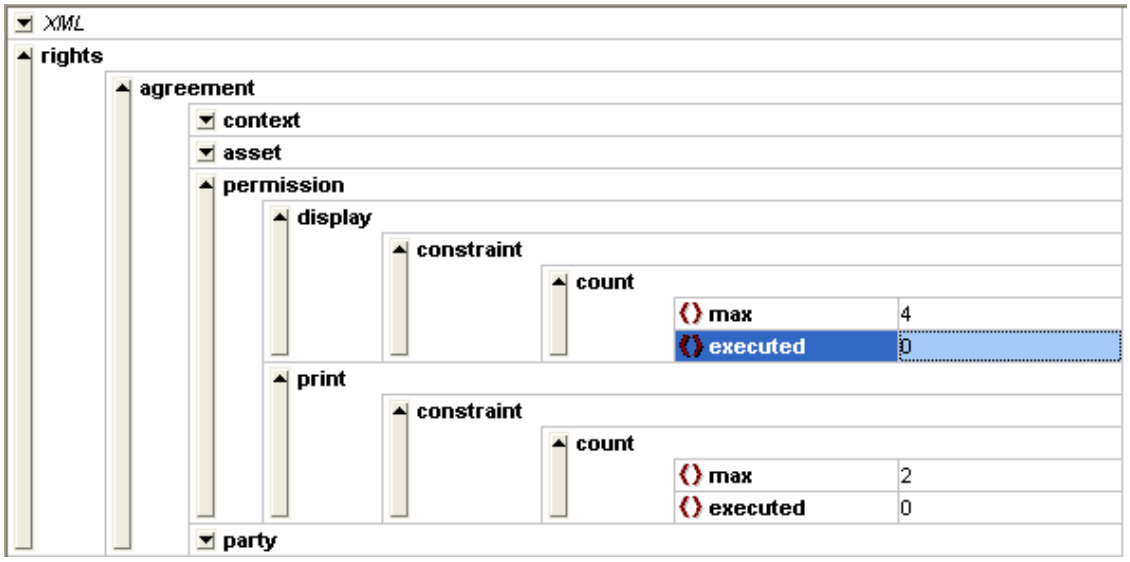
Denne testen tar utgangspunkt i fila *Test_Constraint_Count_Offer.xml.* som er gjengitt i Appendix C. På bakgrunn av denne ble fila *Test_Constraint_Count_Agreement.xml* generert. Ut fra denne genererte fila kan man så teste om test 4 er vellykket

Imidlertid ble det oppdaget en feil i genereringen av avtalen som gjør at systemet ikke fungerer slik det skal. Når det er definert en global constraint count som gjelder for alle rettigheter uten at det er generert en tilsvarende lokal constraint for count for en individuell rettighet, kan brukeren eksekvere denne rettigheten så mange ganger han vil. Adgangen til å eksekvere den individuelle rettigheten skal, når det ikke er definert en lokal constraint for count, ikke kunne eksekveres flere ganger enn det max antall ganger som er definert i den globale deklarasjonen av denne rettigheten. Hvis det derimot er definert en lokal begrensning vha av count fungerer avtalegenereringen som forutsatt, og brukeren gis ikke tilgang til å eksekvere rettigheten mer enn det max antall ganger som er definert i den lokale constraint. Resultatet av avtalegenereringen er gitt under:



Figur 11-5 Testresultat. Test 4. Figur 1.

Forventet resultat av avtalegenereringen skulle vært:



Figur 11-6 Testresultat. Test 4. Figur 2

Ved å legge inn denne ”forventede avtalen” manuelt i systemet ble det fastslått at systemet begrenset adgangen til å eksekvere rettigheten display til det angitte maximum ganger som er fire. Dette var som forventet.

Test 5

Test av at eksekveringen av rettigheten foregår innenfor det tidsrom som er satt som begrensning for eksekvering. Nektes brukeren adgang til en rettighet dersom systemklokka på maskina som systemet kjører på angir at tidspunktet for eksekvering av rettigheten ikke ligger innenfor det daterange som er satt som begrensening for utøvelse av rettigheten?
Denne testen tar utgangspunkt i fila *Test_Constraint_Daterange.xml.*
Systemet ser ut til å håndheve denne begrensningen i utøvelsen av rettigheter som forutsatt, bortsett fra at det ser ut til at rettigheten ikke gis hvis ”i dag” svarer til datoen som er satt som første eller siste dato i et dateRange. Hvis datoen i dag er *mellom* startdate og enddate oppfører systemet seg som forventet.

Test 6

Test av at brukeren er den som rettigheten er begrenset til. Nektes brukeren tilgang til rettigheten dersom det registrerte brukernavnet som brukeren er logget ikke stemmer overens med hva som er angitt av ”individual” i avtalen? Denne testen tar utgangspunkt i fila ”*Test_Constraint_Individual.xml*”. Ved å logge på som med et annet brukernavn enn det som er angitt under individual blir man nektet tilgang til retttigheten selv om man har

en avtale for å eksekvere rettigheten. Det betyr at denne funksjonaliteten virker som forutsatt.

Test 7

Test av om brukeren sitter tilkoblet det nettverk som er angitt av en nettverksmaske eller har en IP adresse som er innenfor det range av IP adresser som er angitt av startIP og endIP. Nektes brukeren adgang til rettigheten dersom hans netverkstilknytning ikke stemmer overens med den som er angitt som constraint for rettigheten?
Dessverre viser det seg at metoden for å sjekke om brukeren sitter tilknyttet det riktige nettverket som angitt av en nettverksmaske eller et range av IP-adresser ikke har blitt implementert i systemet. Dermed kan heller ikke denne funksjonaliteten testes.

11.3 Diskusjon

11.3.1 Er kravsspesifikasjonen oppfylt?

Alle funksjonelle systemkrav er stort sett oppfylt. Man kan ikke se bort fra at en grunn til at de fleste funksjonelle systemkrav er oppfylt er at de var relativt grovt spesifisert i første omgang, men i det store og det hele er funksjonelle og ikke funksjonelle systemkrav oppfylt. Hvis kravsspesifikasjonen hadde vært bedre spesifisert i første omgang ville kanskje en del av de elementene som ble oppdaget under testing kunne kvalifisert som brudd på kravsspesifikasjonen. Spesielt gjelder det test 7 i testingen av systemet som fastsetter at systemet ikke sjekker at brukeren er tilnyttet et netverk som forutsatt i avtalen for et materiale før brukeren tildeles en ticket til å eksekvere rettigheter over materialet.

Ellers er ikke punktet definert i kapittel 9.3.10 om kommunikasjon med brukeren oppfylt. Brukeren trenger som forutsatt kun en nettleser for å aksessere systemet og dets funksjonalitet, men systemet sørger ikke for at all kommunikasjon mellom en nettleser og systemet er kryptert. Grunnen til det er at dette er funksjonalitet som bør tilligge web-tjeneren som systemet kjører på, ikke systemet selv. De fleste web-tjenere har mulighet til å definere porter som fungerer som krypterte kanaler inn til systemet. Siden disse portene kan være forskjellig mellom forskjellige webtjener og instanser av samme webtjener vil det ikke være naturlig for kildekoden å sjekke om en mottatt forespørsel er mottatt over en kryptert forbindelse for så, hvis forespørsel er mottatt over en åpen kanal, å redirigere brukerens nettleser til en gitt port på webtjeneren. Det er ikke sikkert at denne porten er sikret eller i det hele tatt i bruk på den aktuelle webserverinstallasjonen. Derfor overlates det til de som konfigurerer webtjeneren å sørge for at systemet kun kan aksesseres over krypterte forbindelser.

11.3.2 Evaluering av rettighetsspråket for REAP

Siden rettighetsspråket for REAP er så sterkt influert av ODRL er det naturlig å sammenligne de to for å se hvilke aspekter ved ODRL som ikke er dekket av rettighetsspråket og dermed kunne komme fram til en slags evalering av rettighetsspråket for REAP.

Det viktigste elementet er vel kanskje at rettighetsspråket for REAP ikke har noen sikkerhetsmodell. ODRL har en sikkerhetsmodell som lar rettighetsholdere signere ODRL Offers vha av digitale signaturerer slik at deres opphav og autentitet er garantert og lar brukere signere ODRL Agreements på samme måte. Dessuten inneholder ODRL en modell for å angi kryptering for materiale som rettighetsbeskrivelsene gjelder og rettighetsbeskrivelsene selv, slik at bare de som skal ha tilgang til materialet får se det. I et virkelig DRM system er en slik sikkerhetsmodell av vital betydning, selv om det i vil være vanskelig å kreve at brukere av det digitale bibliotek.

Dessuten inneholder ikke rettighetsspråket noen støtte for å trekke tilbake rettigheter som er tilbudt. I ODRL kan dette gjøres på to måter. Enten kan det legges inn ”conditions” for en eller flere rettigheter som fungerer som triggere som reagerer på at en eller annen tilstand blir realitet. Hvis en condition trigges vil da ikke lenger den rettigheten eller rettighetsbeskrivelsen den er definert for være gyldig. Eller så kan man legge inn beskrivelser av såkalte revokes som gjør en rettighetsbeskrivelse ugyldig i det den legges inn i systemet.

I tillegg til at det er mange rettigheter, requirements og constraints som er definert i ODRL som ikke er funnet å være viktige i rettighetsspråket til REAP finnes det også i ODRL en del funksjonalitet som kan brukes for å kombinere entiteter, kalt containers. Containers er en utvidbar aggregering av boolske operatører. I denne aggregeringen inngår operatørene ”and”, ”exclusive or” og ”inclusive or”. Dette gjør at man kan for eksempel kan kombinere betingelser i en avtale. For eksempel kan man uttrykke at for å få tilgang til en gitt tillatelse må man være tilknyttet et nettverk med nettverksmaske #netverksmaske1 ELLER #nettverksmaske2. Dette finnes ikke i rettighetsspråket for REAP men har vist seg å være viktigere enn først antatt.

Ellers finnes det en del metoder for gjenbruk og arving av fragmenter av rettighetsbeskrivelser i ODRL som Expression Linking og Inheritance som spesielt i store dokumenter kan være nyttige. Disse er ikke støttet i rettighetsspråket for REAP, men er heller ikke så viktige for uttrykkskraften til språket. Dette er tatt med i ODRL mer ut fra at språket skal være smidig og enkelt å bruke, ikke for at det skal ha stor uttrykkskraft. ODRL inneholder også støtte for å linke mellom en ODRL Agreement og det ODRL Offer som avtalen er basert på som ikke er støttet i rettighetsspråket for REAP.

En viktig ting å merke seg ved rettighetsspråket for REAP er at det for elementet ”count” er definert 2 mulige subelementer. ”Max” brukes for å angi hvor mange ganger en rettighet kan eksekveres i henhold til en evt avtale. Systemet må da kunne holde orden på hvor mange ganger brukeren har eksekvert rettigheten så langt for å kunne avgjøre om brukeren kan tillates å eksekvere den en gang til. For å holde orden på dette, inkrementerer REAP verdien av subelementet ”executed” hver gang brukeren eksekverer

den gitte rettigheten. Problemene som oppstår i forbindelse med dette og en mulig løsning er behandlet i neste avsnitt.
Constraint elementet individual er heller ikke særlig nyttig. Det er ment at dette elementet skal kunne brukes til å begrense rettigheter til en gitt person. For det første er det ganske begrenset å bare kunne begrense en rettighet til en enkelt person og ikke også til en gruppe av personer. For det andre forventer systemet at verdien av individual tilsvarer et registrert brukernavn i systemet. Rettighetsholdere vil mest sannsynlig ikke vite hvilket brukernavn en bruker er registrert under, og vil da ikke kunne legge inn riktig brukernavn i sine offers. Å forvente et brukernavn her er gjort for at det skal være enkelt å evaluere innholdet av elementet, men denne konstruksjonen gjør at hele elementet er lite nyttig i praktisk bruk.

Ellers er det oppdaget en bug i rettighetsspråket for REAP. Det har vist seg at elementet ”context” er begrenset til å ha 3 subelementer. Dette er i noen tilfeller ikke nok. Det kan for eksempel være ønskelig å beskrive en ressurs med elementer for MIMEType, name, uid og urlReference. Men siden ”context” elementet som inneholder alle disse elementene bare kan ha 3 sub-elementer, må man velge bort minst et av disse sub-elementene. Her burde ikke antall subelementer vært begrenset til 3.

11.3.3 Identifiserte problemer ved min design

Selv om min prototype ikke inneholder funksjonalitet for å vise fram materialet som brukeren forespør, er det likevel støtte i prottypen for å hente materialet fra samlingene på vegne av brukeren. Et problem med prototypen er at metodene som er definert for å gjøre dette baserer seg på at dokumentet som brukeren ønsker tilgang til befinner seg i en enkelt fil. Koden henter da denne ene fila fra hvor det er angitt at denne fila befinner seg. For mange dokumenter er ikke dette nok da ett dokument kan bestå av flere filer. For eksempel vil en typisk html-side gjerne bestå av flere filer, hvor selve htmlfila utgjør en slags masterfil som lenker inn andre filer som for eksempel bildefiler. Levering over http har løst dette problemet ved å sende disse filene i sekvens der klienter først forespør selve html-dokumentet og når klienten finner en referanse til en fil som skal presenteres som del av dokumentet sender den en ny forespørsel om å levere denne. Prototypen REAP er ikke tilpasset slike doumenter som resulterer i en sekvens av forespørsler til systemet. Problemet som melder seg er om det skal være nødvendig å lage avtaler for alle filer som inngår i dokumentet eller ikke. Å kreve at brukeren lager individuelle avtaler for hver fil vil bli veldig arbeidskrevende både for de som lager tilbud om rettighetsoverføring og for brukere som ønsker tilgang til alle filer som utgjør dokumentet. Dessuten må alle html-filer skrives om slik at hver anchor-tag i html-fila refererer til URL for systemet sammen med en collectionId og itemId som svarer til reapId for fila som skal lenkes inn i html-fila. En løsing er å si at bare masterdokumentet skal ha en rettighetsbeskyttelse knyttet til seg og at denne rettighetsbeskrivelsen dekker alle filer som inngår i dokumentet. I så fall vil ikke prototypen i sin nåværende form være i stand til å levere annet enn hovedfila/html-fila da en forespørsel om dokumenter som det ikke er definert rettighetsbeskrivelser for vil føre til at systemet sender en feilmelding tilbake til klienten fordi det ikke fant noe offer/rettighetstilbud for materialet. Den tredje muligheten er

selvfølgelig å kreve at alle filer som skal leveres gjennom REAP må være atomiske dvs av typer tilsvarende ADOBEs pdf format eller lignende. Men dette vil igjen stride mot kravet om at REAP skal kunne levere alle typer digitalt materiale.

Når systemet skal evaluere om en begrensning av antall ganger en rettighet skal kunne eksekveres er brukt opp, altså om brukeren allerede har aksessert rettigheten det antall ganger som er tillatt i følge avtalen han har inngått, må systemet på en eller annen måte finne ut hvor mange ganger rettigheten har vært eksekvert tidligere. I mn løsning er dette løst ved at det er lagt inn et ekstra element executed som inneholder en verdi som angir hvor mange ganger brukeren har eksekvert en gitt rettighet. Denne verdien kan så sammenlignes opp mot verdien av elementet max som angir hvor mange ganger en rettighet kan eksekveres. Hvis verdien av executed er lik verdien av max nekter systemet brukeren å eksekvere rettigheten når brukeren ber om en såkalt ”ticket”. Problemet med denne løsningen er at dette forutsetter at systemet må skrive til en allerede inngått avtale. Dette er både juridisk tvilsomt og umuliggjør digital signering av en avtale. Digitale signaturer baserer seg blant annet på en hash-verdi av det som signeres, og hvis det som signeres forandres vil grunnlaget for signaturen også endres. Dermed vil ikke signaturen lengre være gyldig for avtalen. Det betyr at opplysninger om hvor mange ganger en rettighet i en avtale har vært eksekvert må lagres utenfor avtalen. Det naturlige valg ville da være å lagre slike opplysninger som del av brukerens profil. Dette kunne man sett i forbindelse med hvilket paradigme som er valgt for å gi tilgang til materiale under REAP. Arkitekturen bygger på at man må generere en avtale om bruk av materialert før man får tilgang til det samme materialet. Selv om dette gjenspeiler hvordan man håndterer rettighetsoverføring i den analoge verden kunne man kanskje vurdere om det ikke var like hensiktsmessig å ikke generere avtaler i det hele tatt. En del verdier måtte da lagres i brukerens profil (se for eksempel avsnittet over) og det ville være vanskelig å bruke den løsning som er valgt nå med at brukeren betaler en gang for adgang til å aksessere en rettighet i et tilbud/offer. Dette kunne kanskje vært løst ved at brukeren ble tvunget til å betale for hver enkelt gang han skulle aksessere en ressurs, men det kunne igjen føre til mye ekstra arbeid hver gang brukeren for eksempel ville skrive ut materialet. Jeg er heller ikke sikker på de juridiske aspektene ved en slik løsning tatt i betraktning av at avtaler som regel skrives under for å være juridisk forpliktende. Det er en åpen løsning hvordan man evt skulle kunne gjøre en overføring av rettigheter på denne måten juridisk forpliktende.

En annen ting som kan være verdt å merke seg er at det ikke finnes støtte for å reforhandle avtaler i systemet. Hvis for eksempel brukeren har aksessert rettigheten display det antall ganger som ble avtalt da brukeren inngikk en avtale med systemet har han ikke mulighet til å inngå en ny avtale og betale mer for for eksempel å aksessere en gitt rettighet flere ganger. Det er likevel mulig for brukere å utnytte en svakhet i systemet for å reforhandle avtaler. Det er ingenting i veien for å registrere seg på nytt med et annet brukernavn/epostadresse. Dermed vil brukere kunne lage så mange avtaler som de har fantasi til å finne på epostadresser/brukernavn. Systemet sjekker ikke at brukernavnet faktisk er en gyldig epost-adresse og hvem som eier denne epostadressen.

En annen svakhet med systemet er at når brukere registrerer seg med systemet sjekker ikke systemet om brukernavnet faktisk er registrert tidligere. Hvis noen registrerer seg med en epostadresse som er registrert fra før skriver systemet ganske enkelt over den gamle profilen registrert under det aktuelle brukernavnet.

I prototypen er det ikke tatt høyde for at rettighetsforhold i forbindelse med at opphavsretten er begrenset til et visst antall år. Jeg har heller ikke funnet andre systemer som tar høyde for at rettigheter over et materiale er begrenset i tid. For enkelte av disse systemene er nok dette mer eller mindre begrunnet i et ønske om å kunne ta betalt også for adgang til materale som ikke lengre er beskyttet av opphavsrett, men i et digitalt bibliotek burde dette kanskje vært behandlet annerledes. Man burde kanskje ha en automatisk kontroll på om opphavsretten er utgått, og dermed rettighetshaveres rett til å kunne definere bruksmønstre for sine verk. I de tilfeller opphavsretten er utgått på dato burde materialet være fritt tilgjengelig uten at REAP blandet seg inn i distribusjonen av materialet.

Kodingen av prototypen kunne vært gjort mer smart ved bruk av for eksempel Xpath i de tilfeller der det er snakk om å hente ut enkle verdier eller å fastsette strukturen i xml dokumenter. I prototypen er et gjort utstrakt bruk av manuell traversering av xml-dokumenter for å gjøre slike ting noe som fører til at prototypen blir både uoversiktelig og stor. Dessuten kunne prototypen vært gjort mer stabil ved å fokusere mer på feilhåndtering når uforutsette problemer med eksekveringen dukker opp.

12 Veien framover

Denne prototypen viser at materiale i digitale bibliotek kan beskyttes mot brudd på opphavsrettslige prinsipper, men er på ingen måte en produksjonsløsning. Denne prototypen må ses på som et første forsøk, men kan gi mange innspill til nye forsøk på å lage skikkelige adgangsprotokoller.

Den mest synlige indikasjonen på at prototypen ikke er en produksjonsløsning sett fra brukernes side er vel at det ikke finnes noen god løsning for å la brukeren utøve de rettigheter over et materiale som blir tildelt gjennom generering av en avtale i form av en fremvisningsapplikasjon for rettighetsbasert materiale. En løsning på dette kunne kanskje være å lage en applet som innkorporerer en webleser som kan vise fram materiale til å erstatte. Dette virker kanskje litt bakvendt for mange (En webleser som viser fram en applet som inneholder en webleser), men grunnen til at det evt må gjøres på denne måten er at vanlige weblesere caher og lagrer materialet den laster ned på maskina den kjører på. Dette konstiturer en uønsket eksemplarfremstilling. En Applet har ikke lov til å lagre til maskina den kjører på. Dessuten inneholder tradisjonelle weblesere metoder for for eksempel å skrive ut, lagre og kopiere materiale den laster ned. Disse funksjonene kontakter ikke et DRM system for å få ut en tillatelse for å utføre handlingen før handlingen utføres slik en applikasjon for REAP må.. Derfor kan en applet som kjører i en webleser-vindu som har disse funksjonene koblet ut inneha egne menyer for disse

funksjonene som kontakter DRM systemer for å få tillatelse for å utføre handlingene før de faktisk utføres. En naturlig webleser å inkludere i en slik applet kunne være ICEbrowser fra ICESOFT A/S. **[ICESoftAS]**

For å lage en produksjonsversjon av en slik rettighetsprotokoll som REAP var ment å være må det brukes et annet språk enn det som er brukt her. Å lage et eget språk for slik rettighetsbehandling syntes som et bra valg da det ble gjort, men i fremtiden bør slike systemer basere seg på definerte standarder. Språket som velges i fremtiden må inneholde en sikkerhetsmodell som gjør systemet sikkert i den forstand at rettighetsbeskrivelser krypteres og at brukere kan sikkert identifiseres i henhold til et eller annet skjema (for eksempel et brukernavn fra et annet system) eller vha av digitale certifikater. Dessuten må komunikasjonen mellom brukeren og systemet foregå over krypterte forbindelser. Det siste er først og fremst et spørsmål om konfigurasjon av webserveren som systemet kjører på, og ikke et spørsmål om reprogrammering av systemet som sådan.

Jeg ser ingen bakdeler med å basere en ny implementasjon på referansemodellen som ble presentert tidligere i oppgava.

Selv om REAP ikke fungerer helt som forutsatt mener jeg likevel at denne oppgava viser at et rettighetsspråk og et system som gir eller nekter tilgang til rettigeheter basert på kvalifiserte uttrykk i dette rettighetsspråket kan fungere som adgangsprotokoll for rettighetsbelagt materiale i digitale bibliotek.

13 Referanseliste

[Aalberg2000]
Trond Aalberg:
”IDI DIGLIB, Komponentbasert arkitektur for forskning og utvikling av digitale bibliotek”
”Paper” til Norges forskningsråds 4. nasjonale konferanse i bibliotek- og informasjonsvitenskap, 2000

[AalbergHegna2000]
Trond Aalberg og Knut Hegna
”Arkitektur for digitale bibliotek”
ISBN: 82-7729-026-8
BIBSYS 2000
http://www.bibsys.no/BDB/arkitektur/ArkDigBib.pdf

[Alexandria]
Alexandria Digital Library Project
University of California, Santa Barbara
Sist sjekket jan 2002
http://www.alexandria.ucsb.edu

[ArBlOv97]
Arms, William Y., Christophe Blanchi and Edward A. Overly
" An Architechture for Information in Digital Libraries"
Dlib Magazine, February 1997
http://www.dlib.org/dlib/february97/cnri/02arms1.html

[Arms95]
"Key Consepts in the Architecture of the Digital Library"
William Y. Arms
D-Lib Magazine, July 1995
http://www.d-lib.org/dlib/July95/07arms.html

[Belkin1998]
"Understanding and supporting Multiple Information Seeking Behaviors in a Single Inteface Framework"
Belkin, Nicholas J.
Proseedings of Eight DELOS Workshop: User Interfaces in Digital Libraries.
DELOS Working Group Report No.99/W001, (1998)
http://www.ercim.org/publication/ws-proceedings/DELOS8/delos8.pdf

[Bernkonvensjonen]
Berne Convention for the Protection of Literary and Artistic works
http://www.wipo.int/clea/docs/en/wo/wo001en.htm

[BernkonvensjonenNorsk]
Bernkonvensjonen om vern av litterære og kunstneriske verk
Til Norsk ved advokat Astri M. Lund
Kopinor, Oslo, 1990
http://www.kopinor.no/dokumentbank/bernknv.html

[Bing]
Rettslige aspekter ved elektronisk formidling av maateriale fra arkiv, museum, bibiliotek, universitet og visse andre institusjoner
Arbeidsnotat nr. 12
Norsk kulturråd-utredning 1996
http://www.kulturrad.no/

[Borgman2000]
Borgman, Christine L.
"From Gutenberg to the global information infrastructure: Access to information in the networked world"
MIT Press, c2000
ISBN: 0-262-02473-x

[Copyright law]

Copyright Law of the United States of America"
Title 17, United States Code
http://www.copyright.gov/title17/92Chap1.html

[DLFstrategy&business2000]
"DLF draft strategy and Business plan"
Public version 2.0
D. Greenstein
Digital Library Federation.
25. September 2000
http://www.diglib.org/dlfhomepage.htm

[DMCA]
THE DIGITAL MILLENNIUM COPYRIGHT ACT OF 1998
U.S. Copyright Office Summary
December 1998
http://www.loc.gov/copyright/legislation/dmca.pdf

[DMCA_Cons]
"EFF Whitepaper: Unintended Consequences. Three Years under the DMCA"
Senior Intellectual Property Attorny Fred von Lohman
Electronic Frontier Foundation
May 2002
http://www.eff.org/IP/DMCA/20020503_dmca_consequences.html

[DOI]
Digital Object Identifier
Sist sjekket jan 2002
http://www.doi.org

[DRMWatch]
DRMWatch
GiantSteps Media Technology Strategies
Sist sjekket aug 2002
http://www.giantstepsmts.com/drmwatch.htm

[E.Gerck98]
Overview of certification Systems: X.509, CA, PGP and SKIP
Version 1.8
1 aug. 1998
http://www.mcg.org.br/cert.htm

[ElektroniskSignaturLov]
LOV 2001-06-15 nr 81: Lov om elektronisk signatur
http://www.lovdata/all/hl-20010615-081.html

[Erickson2001]
"Information Objects and Rights Management
A Mediation-based Approach to DRM Interoperability"
John S. Erickson
Hewlett-Packard Laboratories
1. april 2001
http://www.dlib.org/dlib/april01/erickson/04erickson.html

[FairUse&DRM]
"Fair Use and Digital Rights Management: Preliminary thoughts on the (Irreconcilable?) Tension between Them."
Senior Intellectual Property Attorny Fred von Lohman
Electronic Frontier Foundation
16. april 2002
http://www.eff.org/IP/DRM/fair_use_and_drm.html

[Gillian-Swetland]
"Defining Metadata"
Anne J. Gilliand-Swetland
M.Baca, ed. Introduction to Metadata: Pathways to Digital Information. Los Angeles, Getty Information Institute, 1998
http://faui80.informatik.uni-erlangen.de/IMMD8/Lectures/DIGLIB/IntroMetadata/toc.htm

[Ianella2001]
"Digital Rights Management (DRM) architechtures"
Renato Ianella
Chief Scientist, IPR System
1. juni 2001
http://www.dlib.org/dlib/june01/ianella/06iannella.html

[ICE]
The Information and Content Exchange
ICE Spesification version 1.1
Sist sjekket Februar 2002
http://www.icestandard.org

[ICESoftAS]
ICESoft A/S
IceSoft Home Page
Sist sjekket 11. Sept 2002
http://www.icesoft.no

[IETF]
Internet Engineering Task Force Homepage
Sist sjekket mars 2002

http://www.ietf.org/

[INDECS]
<in*d*ecs>
interoperability of data in e-commerce systems
Sist sjekket Januar 2002
http://www.indecs.org

[ISO 8601]
ISO 8601:2000
Data elements and interchange formats—Information interchange—Representation of dates and times
International Organization for Standardization, 2000
http://www.iso.ch/markete/8601.pdf

[JakartaTomcat]
The jakarta project
http://jakarta.apache.org/tomcat

[Kopinor]
Om Kopinor
http://www.kopinor.no/norsk/om_kopinor/idex.html

[Lagoze]
Defining Collections in Distributed Digital Libraries.
Lagoze, C., Fielding D.
DLIB Magazine (November 1998)
http://www.dlib.org/dlib/november98/lagoze/11Lagoze.html

[Lagoze2000]
The Cornell Digital Library Research Group: Architectures and Policies for Distributed Digital Libraries
Invited Paper for DLW17, Tsukuba, Japan February 2000
Carl Lagoze
Februar 2000
http://www.cs.cornell.edu/lagoze/papers/DLW17/cdlrg.htm

[LevyMarshal95]
”Going Digital: A Look at Assumptions Underlying Digital Libraries”
David M. Levy and Catherine C. Marshal
Communications of the ACM April 1995/vol38, No. 4

[Nielsen1999]
“User interface Directions for the web.”
Jacob Nielsen
Communications of the ACM Vol. 42, No 1.

Januar 1999

[OASIS]
OASIS
Organization for the Advancement of Structured Information Standards
Sist sjekket Mai 2002
http://www.oasis-open.org

[ODRL]
The Open Digital Rights Language Initiative
Sist sjekket April 2002
http://odrl.net

[OSI]
Open Systems Interconnect Reference Model
ISO/IEC 7498
International Standards Organization
http://www.iso.org

[PayetteLagoze2000]
"Policy-Carrying, Policy-Enforcing Digital Objects"
Sandra Payette and Carl Lagoze
Department of Computer Science, Cornell University
Research and advanced Technology for Digital Libraries
4th European Conference on ECDL
Lisbon, Portugal Sept 2000
Proceedings

[Purl]
Persistent Uniform Resource Locator
Sist sjekket jan 2002
http://www.purl.org

[RSA]
A method of obtaining digital signatures and public key cryptosystems
Rivest , R.L., Shamir, A., and Adelman, L., 1978
Comms. ACM, vol 21, No 2, pp 120-6

[RosenblattTrippeMooney]
Digital Rights Management Business and Technology
Bill Rosenblatt, Bill Trippe, and Stephen Mooney
Hungry Minds, 2002
ISBN: 0-7645-4889-1

[SDLIP]
The Simple Digital Library Interoperability Protocol (SDLIP-Core)

Sist sjekket juni 2002
http://www-diglib.stanford.edu/~testbed/doc2/SDLIP/

[SSL]
Secure Socket Layer
SSL 3.0 Specification
Sist sjekket sept 2002
http://home.netscape.com/eng/ssl3/index.html

[Stefik]
"Letting Loose the Light: Igniting Commerce in Electronic Publication"
In: Internet Dreams: Archetypes, Myths and Metaphors
Authors Mark J Stefik, Vinton g.Cerf
Isbn: 0262692023 (may 9. 1997)

[Torvund]
Opphavsrett – en introduksjon
Professor dr. juris Olav Torvund
Sist sjekket juli 2002
http://www.torvund.net/artikler/art-opphav.asp

[TSL]
Transport Layer Security Protocol version 1.0
Internet Engineering TaskForce
Januar 1999
http://www.ietf.org/rfc/rfc2246.txt?number=2246

[URN]
Uniform Resource Names (urn)
31. Juli 2001
http://www.ietf.org/html.charters/urn-charter.html

[w3c]
World Wide Web Consortium.
Sist sjekket mars 2002
http://www.w3.org

[WIPO]
World Intellectual Property Organization
Sist sjekket August 2002
http://www.wipo.org/

[Xindice]
Apache Xindice
Native xml database.
Sist sjekket Mars 2002

http://xml.apache.org/xindice

[XMCL]
eXtensible Media Commerce Language
draft spesification
sist sjekket januar 2002
http://www.xmcl.org/spesification.html

[XML_Rec]
W3C
Extensible Markup Language (XML) 1.0 (Second Edition)
W3C Recommendation 6. October 2000
http://www.w3c.org/TR/REC-xml

[XMLSchema]
W3C
XML Schema
Sist sjekket juni 2002
http://www.w3c.org/XML/Schema

[XMLSchemaTutorial]
”Using the W3C XML Schema.”
Eric van der Vlist
29 november 2000
http://www.xml.com/lpt/a/2000/11/29/schemas/part1.html

[XrML]
eXtensible rights Markup Language
http://www.xrml.org

[Z39.50]
Z39.50 Maintainance Agency
Sist sjekket jan 2002
http://www.loc.gov/z3950/agency/

[Åndsverksloven]
Lov 1961-05-12 nr 2: Lov om opphavsrett til åndsverk m.v.(Åndsverksloven)
12 Mai 1961, sist endret Lov-2000-06-23-52 fra 2001-07-01
ISBN: 82-504-1143-9
http://www.lovdata.no/all/hl/-19610512-002.html

Appendix A

Rettighetsspråk for REAP

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Øyvind Vestavik (NTNU) -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">

<!--This is the root element of all rights descriptions-->
    <xsd:element name="rights" type="rightsType"/>

    <!--A rights description has to be either an offer or an agreement-->
    <xsd:complexType name="rightsType">
        <xsd:choice>
            <xsd:element ref="offer"/>
            <xsd:element ref="agreement"/>
        </xsd:choice>
    </xsd:complexType>

    <!--Both offer and agreements consists of the four elements; context, asset, permisson, and party. Context
    is only used for labeling the offer/agreement with a date. Dates are given in the form
    2001-05-01 meaning may 1st 2001-->
    <xsd:element name="offer" type="offerAgreeType"/>
    <xsd:element name="agreement" type="offerAgreeType"/>
    <xsd:complexType name="offerAgreeType">
        <xsd:sequence>
            <xsd:element ref="context"/>
            <xsd:element ref="asset"/>
            <xsd:element ref="permission"/>
            <xsd:element ref="party"/>
        </xsd:sequence>
    </xsd:complexType>

    <!--Description of assets under REAP.-->
    <xsd:complexType name="assetType">
        <xsd:sequence>
            <xsd:element ref="reapId"/>
            <xsd:element ref="context"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="asset" type="assetType"/>

    <!--unique identification of material under REAP-->
    <xsd:complexType name="reapIdType">
        <xsd:sequence>
            <xsd:element name="collectionId" type="xsd:string"/>
            <xsd:element name="itemId" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="reapId" type="reapIdType"/>

    <!--defines the combinations of permissions, constraints and requrements that can be defined directly under
    the permission element. Constraints and requirements defined directly under the permission element must be
    honoured by all permissions. For each permissionelement, further constraints and requirements can be
    defined-->
    <xsd:complexType name="permissionType">
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="permissionElement"/>
            <xsd:element ref="constraint"/>
            <xsd:element ref="requirement"/>
        </xsd:choice>
```

```
</xsd:complexType>
<xsd:element name="permission" type="permissionType"/>
<!--These are the permissions that can be granted over an material under REAP
For each permission one can define further requrements and constraints -->
<xsd:element name="display" type="constraintRequirementType" substitutionGroup="permissionElement"/>
<xsd:element name="print" type="constraintRequirementType" substitutionGroup="permissionElement"/>
<xsd:element name="play" type="constraintRequirementType" substitutionGroup="permissionElement"/>
<xsd:element name="execute" type="constraintRequirementType" substitutionGroup="permissionElement"/>
<xsd:element name="permissionElement" type="constraintRequirementType" abstract="true"/>

<!--for each of the above permissions constraints and requirements can be set-->
<xsd:complexType name="constraintRequirementType">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element ref="constraint" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="requirement" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:choice>
</xsd:complexType>

<!--These are the constraints that can be defined for permissions-->
<xsd:complexType name="constraintType">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="individual" type="xsd:string"/>
        <xsd:element name="network" type="networkType"/>
        <xsd:element name="count" type="countType"/>
        <xsd:element name="dateRange" type="dateRangeType"/>
    </xsd:choice>
</xsd:complexType>
<xsd:element name="constraint" type="constraintType"/>

<!--construction of a date range Dates must conform to ISO standards: yyyy.mm.dd-->
<xsd:complexType name="dateRangeType">
    <xsd:sequence>
        <xsd:element name="start" type="xsd:date"/>
        <xsd:element name="end" type="xsd:date"/>
    </xsd:sequence>
</xsd:complexType>

<!--defines the construction of a network constraint -->
<xsd:complexType name="networkType">
    <xsd:choice>
        <xsd:element name="networkMask"/>
        <xsd:sequence>
            <xsd:element name="startIPnr"/>
            <xsd:element name="endIPnr"/>
        </xsd:sequence>
    </xsd:choice>
</xsd:complexType>

<!-- Define the constructions of a count constraint -->
<xsd:complexType name="countType">
    <xsd:sequence>
        <xsd:element name="max">
            <xsd:simpleType>
                <xsd:restriction base="xsd:int">
                    <xsd:minInclusive value="0"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:element>
        <xsd:element name="executed" minOccurs="0">
            <xsd:simpleType>
                <xsd:restriction base="xsd:int">
                    <xsd:minInclusive value="0"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:element>
    </xsd:sequence>
```

```
</xsd:complexType>

<!--Requirementmodel-->
<xsd:element name="requirement" type="requirementType"/>
<xsd:complexType name="requirementType">
    <xsd:sequence>
        <xsd:element name="prepay" type="feeType"/>
    </xsd:sequence>
</xsd:complexType>

<!--defines a payment -->
<xsd:element name="payment">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="amount">
                <xsd:complexType>
                    <xsd:simpleContent>
                        <xsd:extension base="xsd:decimal">
                            <xsd:attribute name="currency" type="xsd:NMTOKEN" use="required"/>
                        </xsd:extension>
                    </xsd:simpleContent>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<!-- By extending this construct, other feetypes than payments can be supported-->
<xsd:complexType name="feeType">
    <xsd:sequence>
        <xsd:element ref="payment"/>
    </xsd:sequence>
</xsd:complexType>

<!--context model-->
<xsd:complexType name="contextType">
    <xsd:sequence>
        <xsd:element ref="contextElement" minOccurs="0" maxOccurs="3"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:element name="context" type="contextType"/>

<!—These are the context elements possible-->
<xsd:element name="MIMEType" type="xsd:string" substitutionGroup="contextElement"/>
<xsd:element name="date" type="xsd:date" substitutionGroup="contextElement"/>
<xsd:element name="name" type="xsd:string" substitutionGroup="contextElement"/>
<xsd:element name="URLreference" type="xsd:anyURI" substitutionGroup="contextElement"/>
<xsd:element name="uid" substitutionGroup="contextElement">
    <xsd:complexType>
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="idscheme" use="required">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:NMTOKEN">
                            <xsd:enumeration value="URI"/>
                            <xsd:enumeration value="DOI"/>
                            <xsd:enumeration value="x500"/>
                            <xsd:enumeration value="ISBN"/>
                            <xsd:enumeration value="ISTC"/>
                            <xsd:enumeration value="ISO3166"/>
                            <xsd:enumeration value="FPI"/>
                            <xsd:enumeration value="GUID"/>
                            <xsd:enumeration value="mpeg_ID"/>
                            <xsd:enumeration value="ISAN"/>
                            <xsd:enumeration value="ISRC"/>
                            <xsd:enumeration value="ISSN"/>
```

```
                                    <xsd:enumeration value="ISWC"/>
                                    <xsd:enumeration value="IDDN"/>
                                    <xsd:enumeration value="UUID"/>
                                </xsd:restriction>
                            </xsd:simpleType>
                        </xsd:attribute>
                    </xsd:extension>
                </xsd:simpleContent>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="contextElement" abstract="true"/>

        <!-- partymodel.-->
        <xsd:complexType name="partyType">
            <xsd:sequence>
                <xsd:element ref="user" minOccurs="0"/>
                <xsd:element ref="rightsholder" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="party" type="partyType"/>

        <!-- Element user don't apply in offers, but are inserted into agreements by the system as they are
        created -->
        <xsd:element name="user">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element ref="context"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>

        <!--defining the constructs to describe a rightsholder. The rightsholder element appears directly under the
        party element and describes the ones that are entitled to attribution if an agreement is made-->
        <xsd:complexType name="rightsHolderType">
            <xsd:sequence>
                <xsd:element ref="context"/>
                <xsd:element ref="attribution"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="rightsholder" type="rightsHolderType"/>

        <xsd:complexType name="attributionType">
            <xsd:choice>
                <xsd:element ref="fixedamount"/>
                <xsd:element ref="percentage"/>
            </xsd:choice>
        </xsd:complexType>

        <!-- defining the attribution to be given to rightsholders-->
        <xsd:element name="attribution" type="attributionType"/>
        <xsd:element name="fixedamount" type="feeType"/>
        <xsd:element name="percentage">
            <xsd:simpleType>
                <xsd:restriction base="xsd:decimal">
                    <xsd:minInclusive value="0"/>
                    <xsd:maxInclusive value="100"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:element>

</xsd:schema>
```

Appendix B

Ekstern programvare og programvare biblioteker

Tekstredigering

Denne oppgaven er skrevet i MSWord fra Microsoft

For å lage grafiske fremstillinger i har det blitt brukt forskjellig ekstern programvare. Her er en kort liste av de applikasjoner som er brukt:

Poseidon UML Community Edition fra Gentleware AG
Kan lastes ned gratis fra http://www.gentleware.com
Microsoft Visio.
XML Spy version 4.4
Kan lastes ned fra *http://www.xmlspy.com*

XML manipulering og editering

For all behandling av XML-dokumenter har jeg brukt XMLSpy v.4.4. XMLSpy er shareware og kan lastes ned fra http://www.xmlspy.com

For at koden skal være i stand til å traverser xml filer og dokumenter har jeg benyttet programvarebiblioteket Xerces fra Apache.

Lagring av og gjenhenting av informasjon i prototypen

All informasjon i xmlformat som lagres i forbindelse med REAP lagres i en ”native xml data base” kalt Apache Xindice som kan lastes ned gratis fra http://www.apache.org/xindice Jeg har brukt versjon 1.0 beta 4. Dette er en database som utvikler seg raskt og som Apache Software Foundation har adoptert fra dbxml.org. Det er definert et api for tilgang til xml-databaser kalt XML:DB. Utviklingsgruppen for dette api har laget en referanseside som kan ses på http://www.XML:DB.org, og Xinidice støtter dette api. Mine servelets og klasser bruker dette api for å få tilgang til data som er lagret i databasen og for å lagre/oppdatere data i databasen. Informasjon om maskinoppsett for databasen og setting av Path og classpath er beskrevet i filer som distribueres sammen med databasen. Distribusjonen inneholder også jar-filer som trengs både for kompilering av min kode og for kjøring av min kode.

Koding

Javaklasser og html-dokumenter er skrevet delvis i programmeringseditoren **EditPlus**, men for det meste har utviklingen av prototypen vært utviklet vha Borland Jbuilder 6.0 Enterprise Edition. EditPlus kan lastes ned som shareware

fra www.editplus.com. Borland Jbuilder 6.0 Enterprise Edition er en relativt dyr programvarepakke, men Borland har også en versjon av programvaren som kan lastes ned gratis fra *www.borland.com*

Kompilering

Kompilering av kode har vært gjort fra kommandolinjen vha jdk 1.3.1 (Java2) og internt i Borland Jbuilder fra Borland. For kompilering av servlets kreves at en del klasser som utvider Java2 Standarden befinner seg i classpath på maskina som servletene kompileres på. Jeg har brukt en zip-fil kalt **servlet-2_3-fcs-classfiles.zip** som kan lastes ned fra http://java.sun.com. I tillegg trengs diverse pakker av JAVA klasser som utgjør api mot databasen.(se avsnitt om Lagring og gjenhenting av informasjon over) Disse jar-filene må også befinne seg i classpath på maskina man kompilerer på. (Merk at Jbuilder lager sin egen classpath som ikke er avhengig av classpath på maskina)

Eksekvering av servlets

Som grensesnittet mot sluttbrukere brukes servlets som genererer html-kode dynamisk som svar på brukeres kall til disse servlet'ene. Tolking av forespørsler og eksekvering av kode i servlets gjøres av **Jakarta Tomcat** som er Apaches referanseimplementasjon av en servletcontainer, og støtter hele servlet-spesifikasjonen. Tomcat kan lastes ned fra http://jakarta.apache.org/tomcat**]**

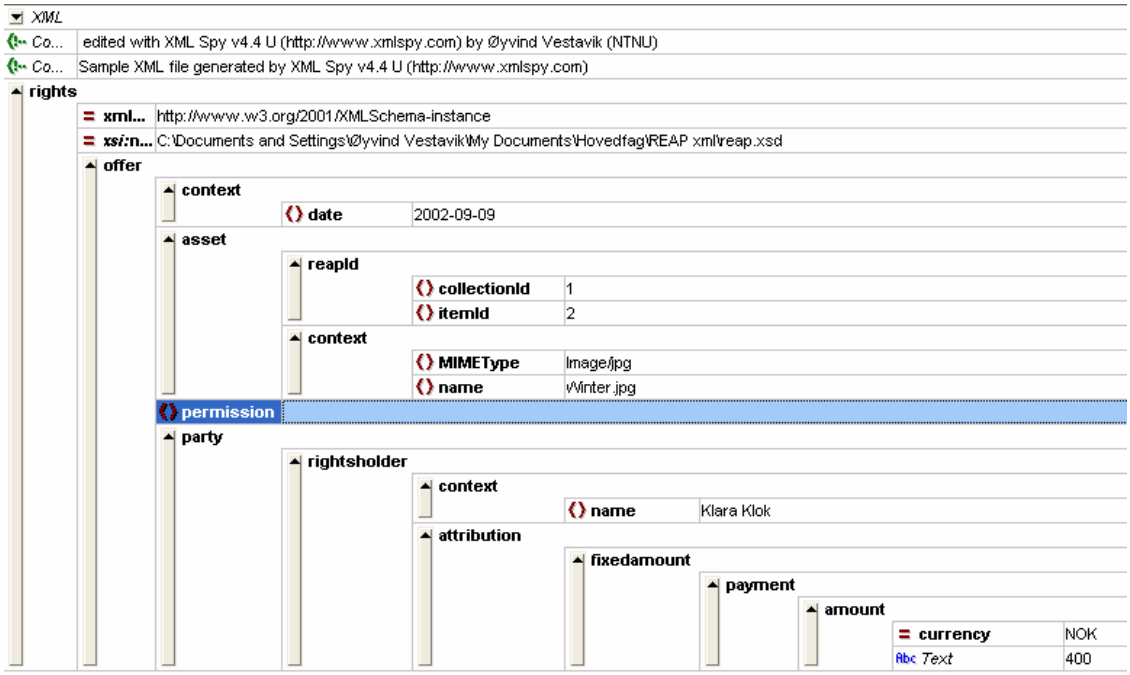
Under eksekvering av koden min trenger Tomcat tilgang til jar filene som var angitt under avsnittet om kompilering. Hele prototypen er laget som en webapp, og alle eksterne kodebiblioteker som brukes av koden min distribueres sammen med web-applikasjonens war-fil. Kodebibliotekene plasseres i katalogen *web-inf/lib* i webapplikasjonen.

Appendix B Ekstern programvare og programvarebiblioteker

Testdokumenter.

Dette er en samling xml dokumenter som angir eksempler på offers eller tilbud om rettighetsoverføringer slik de kan angis når materiale skal publiseres gjennom REAP. Dette er dokumentene som er brukt i testene av REAP

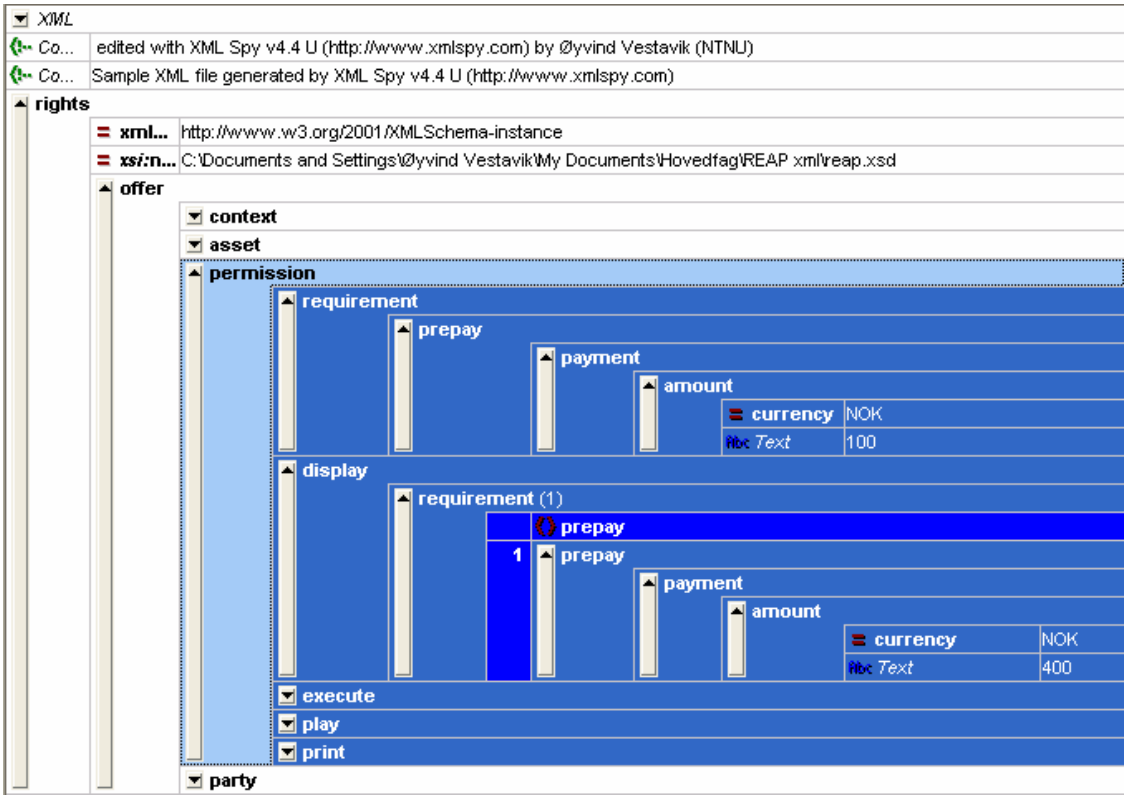
BasicOffer.xml




```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Øyvind Vestavik (NTNU) -->
<!--Sample XML file generated by XML Spy v4.4 U (http://www.xmlspy.com)-->
<rights xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\Documents and Settings\Øyvind Vestavik\My Documents\Hovedfag\REAP
xml\reap.xsd">
    <offer>
        <context>
            <date>2002-09-09</date>
        </context>
        <asset>
            <reapId>
                <collectionId>1</collectionId>
                <itemId>2</itemId>
            </reapId>
            <context>
                <MIMEType>Image/jpg</MIMEType>
                <name>Winter.jpg</name>
            </context>
        </asset>
        <permission/>
        <party>
            <rightsholder>
                <context>
```

```
                <name>Klara Klok</name>
            </context>
            <attribution>
                <fixedamount>
                    <payment>
                        <amount currency="NOK">400</amount>
                    </payment>
                </fixedamount>
            </attribution>
        </rightsholder>
    </party>
</offer>
</rights>
```

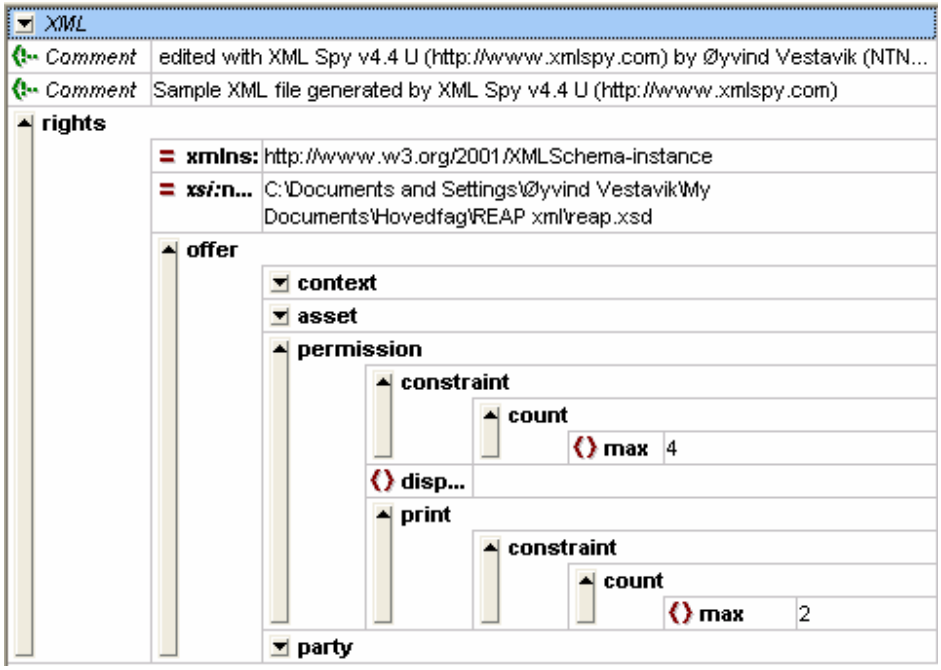
Test_Requirement_Offer.xml




```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Øyvind Vestavik (NTNU) -->
<!--Sample XML file generated by XML Spy v4.4 U (http://www.xmlspy.com)-->
<rights xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\Documents and Settings\Øyvind Vestavik\My Documents\Hovedfag\REAP
xml\reap.xsd">
    <offer>
        <context>
            <date>2002-09-09</date>
        </context>
        <asset>
            <reapId>
                <collectionId>1</collectionId>
                <itemId>2</itemId>
            </reapId>
```

```
            <context>
                <MIMEType>Image/jpg</MIMEType>
                <name>Winter.jpg</name>
            </context>
        </asset>
        <permission>
            <requirement>
                <prepay>
                    <payment>
                        <amount currency="NOK">100</amount>
                    </payment>
                </prepay>
            </requirement>
            <display>
                <requirement>
                    <prepay>
                        <payment>
                            <amount currency="NOK">400</amount>
                        </payment>
                    </prepay>
                </requirement>
            </display>
            <execute>
                <requirement>
                    <prepay>
                        <payment>
                            <amount currency="NOK">400</amount>
                        </payment>
                    </prepay>
                </requirement>
            </execute>
            <play>
                <requirement>
                    <prepay>
                        <payment>
                            <amount currency="NOK">400</amount>
                        </payment>
                    </prepay>
                </requirement>
            </play>
            <print>
                <requirement>
                    <prepay>
                        <payment>
                            <amount currency="NOK">100</amount>
                        </payment>
                    </prepay>
                </requirement>
            </print>
        </permission>
        <party>
            <rightsholder>
                <context>
                    <name>Klara Klok</name>
                </context>
                <attribution>
                    <fixedamount>
                        <payment>
                            <amount currency="NOK">100</amount>
                        </payment>
                    </fixedamount>
                </attribution>
            </rightsholder>
        </party>
    </offer>
</rights>
```

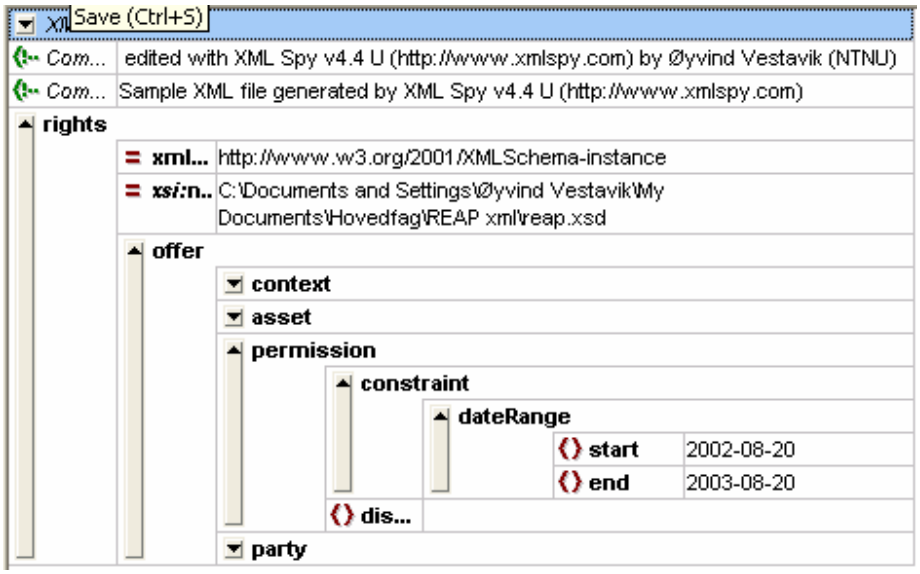
Test_Constraint_Count_Offer.xml




```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Øyvind Vestavik (NTNU) -->
<!--Sample XML file generated by XML Spy v4.4 U (http://www.xmlspy.com)-->
<rights xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\Documents and Settings\Øyvind Vestavik\My Documents\Hovedfag\REAP
xml\reap.xsd">
    <offer>
        <context>
            <date>2002-09-09</date>
        </context>
        <asset>
            <reapId>
                <collectionId>1</collectionId>
                <itemId>2</itemId>
            </reapId>
            <context>
                <MIMEType>Image/jpg</MIMEType>
                <name>Winter.jpg</name>
            </context>
        </asset>
        <permission>
            <constraint>
                <count>
                    <max>4</max>
                </count>
            </constraint>
            <display/>
            <print>
                <constraint>
                    <count>
                        <max>2</max>
                    </count>
                </constraint>
            </print>
```

```
        </permission>
        <party>
            <rightsholder>
                <context>
                    <name>Klara Klok</name>
                </context>
                <attribution>
                    <fixedamount>
                        <payment>
                            <amount currency="NOK">100</amount>
                        </payment>
                    </fixedamount>
                </attribution>
            </rightsholder>
        </party>
    </offer>
</rights>
```

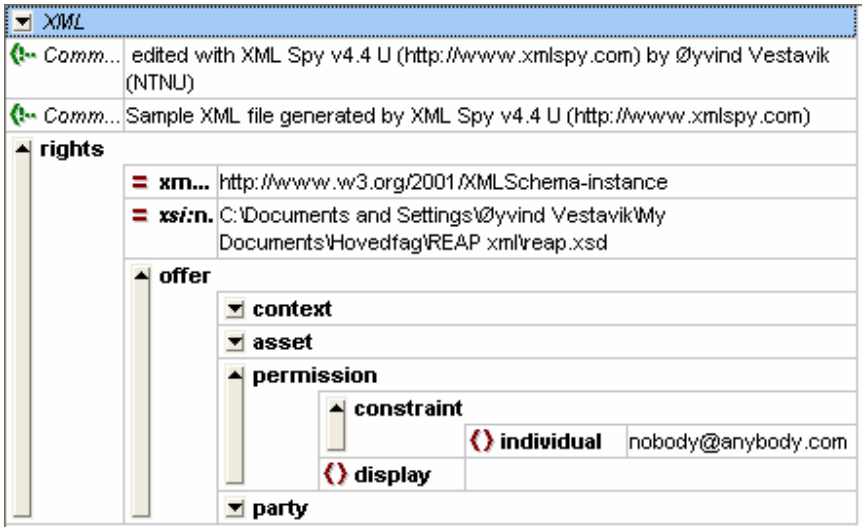
Test_Constraint_dateRange_Offer.xml




```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Øyvind Vestavik (NTNU) -->
<!--Sample XML file generated by XML Spy v4.4 U (http://www.xmlspy.com)-->
<rights xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\Documents and Settings\Øyvind Vestavik\My Documents\Hovedfag\REAP
xml\reap.xsd">
    <offer>
        <context>
            <date>2002-09-09</date>
        </context>
        <asset>
            <reapId>
                <collectionId>1</collectionId>
                <itemId>2</itemId>
            </reapId>
            <context>
                <MIMEType>Image/jpg</MIMEType>
                <name>Winter.jpg</name>
            </context>
        </asset>
        <permission>
```

```
            <constraint>
                <dateRange>
                    <start>2002-08-20</start>
                    <end>2003-08-20</end>
                </dateRange>
            </constraint>
            <display/>
        </permission>
        <party>
            <rightsholder>
                <context>
                    <name>Klara Klok</name>
                </context>
                <attribution>
                    <fixedamount>
                        <payment>
                            <amount currency="NOK">100</amount>
                        </payment>
                    </fixedamount>
                </attribution>
            </rightsholder>
        </party>
    </offer>
</rights>
```

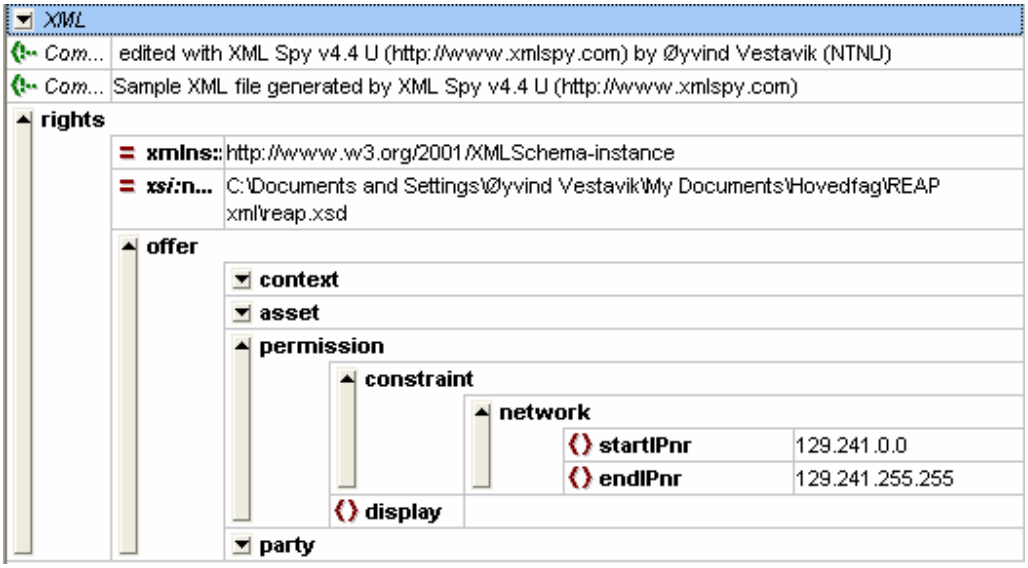
Test_Constraint_Individual_Offer.xml




```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Øyvind Vestavik (NTNU) -->
<!--Sample XML file generated by XML Spy v4.4 U (http://www.xmlspy.com)-->
<rights xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\Documents and Settings\Øyvind Vestavik\My Documents\Hovedfag\REAP
xml\reap.xsd">
    <offer>
        <context>
            <date>2002-09-09</date>
        </context>
        <asset>
            <reapId>
                <collectionId>1</collectionId>
                <itemId>2</itemId>
            </reapId>
```

```
            <context>
                <MIMEType>Image/jpg</MIMEType>
                <name>Winter.jpg</name>
            </context>
        </asset>
        <permission>
            <constraint>
                <individual>nobody@anybody.com</individual>
            </constraint>
            <display/>
        </permission>
        <party>
            <rightsholder>
                <context>
                    <name>Klara Klok</name>
                </context>
                <attribution>
                    <fixedamount>
                        <payment>
                            <amount currency="NOK">100</amount>
                        </payment>
                    </fixedamount>
                </attribution>
            </rightsholder>
        </party>
    </offer>
</rights>
```

Test_Constraint_Network_Offer.xml




```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Øyvind Vestavik (NTNU) -->
<!--Sample XML file generated by XML Spy v4.4 U (http://www.xmlspy.com)-->
<rights xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\Documents and Settings\Øyvind Vestavik\My Documents\Hovedfag\REAP
xml\reap.xsd">
    <offer>
        <context>
            <date>2002-09-09</date>
```

```
        </context>
        <asset>
            <reapId>
                <collectionId>1</collectionId>
                <itemId>2</itemId>
            </reapId>
            <context>
                <MIMEType>Image/jpg</MIMEType>
                <name>Winter.jpg</name>
            </context>
        </asset>
        <permission>
            <constraint>
                <network>
                    <startIPnr>129.241.0.0</startIPnr>
                    <endIPnr>129.241.255.255</endIPnr>
                </network>
            </constraint>
            <display/>
        </permission>
        <party>
            <rightsholder>
                <context>
                    <name>Klara Klok</name>
                </context>
                <attribution>
                    <fixedamount>
                        <payment>
                            <amount currency="NOK">100</amount>
                        </payment>
                    </fixedamount>
                </attribution>
            </rightsholder>
        </party>
    </offer>
</rights>
```

Appendix D.

JavaDoc dokumentasjon for prototypen REAP

Dette er dokumentasjonen for kildekoden for REAP basert på JavaDoc kommentarer i kildekoden. De resulterende websidene som gjengis i dette appendix er generert vha JBuilders innebygde JavaDoc tool.

Package Summary

Package reap.servlets

This package contains all java servlets that makes up the user interface

reap.servlets
Class get_Document

public class **get_Document**
extends HttpServlet

Description: This servlet is the starting point for users wanting access to material protected by REAP and should be called with http get parameters in the link to the servlet to determine which material the user is seeking access to. An example of such a url would be *http://www.myserver.com[:portnumber]/reap/get_document?CollectionId=4&ItemId=8*

Copyright: Copyright (c) 2002 Øyvind Vestavik

Version:
1.0
Author:
Øyvind Vestavik

Field Detail

CONTENT_TYPE
private static final String **CONTENT_TYPE**

Description: Contains the default mime return type of the servlet.

Method Detail

doGet
public void **doGet**(HttpServletRequest request,
HttpServletResponse response)

Title: method doGet

Description: This method receives the CollectionId and ItemId for the document the user want access to in the "request object". (See the overall description of the class). If CollectionId or ItemId is missing, an errorpage is written. Users must be logged in to get access to the services of this method and are redirected to the login servlet if not logged in. Makes session to store details of request to be retrieved later. If the user is logged in, checks to see if the user has an agreement for the material he is requesting. If not, the user is redirected to the makeAgreement servlet in order to have such an agreement. If the agreement generation succeeds the user is thrown back to this servlet, which starts up the rendering application

Parameters:
request - An object Containing all details of the request
response - An object used for sending information and directives back to the users browser
Throws:
ServletException -
IOException -

---

reap.servlets
Class login

public class **login**
extends HttpServlet

Description: This servlet is used for logging in users

Copyright: Copyright (c) 2002 Øyvind Vestavik

Version:
1.0
Author:
Øyvind Vestavik

Field Detail

CONTENT_TYPE
private static final String **CONTENT_TYPE**

Description: Contains the default mime return type of the servlet.

Method Detail

doGet
public void **doGet**(HttpServletRequest request,
HttpServletResponse response)

Description: Presents the user with a login page. The page also contains a link to the registration servlet

Parameters:
request - an object containing details of the request
response - an object representing this servlets response
Throws:
ServletException -
IOException -

---

doPost
public void **doPost**(HttpServletRequest request,
HttpServletResponse response)

Description: Receives username and password from the form generated by doGet() and tries to log in the user. If the login fails, the the user is redirected back to the loginpage (doGet method of this servlet)

Parameters:
request -

response -

Throws:

ServletException -
IOException -

---

reap.servlets
Class makeAgreement

public class **makeAgreement**
extends HttpServlet

Description: This servlet is used for interaction with the user in the process of making an agreement

Copyright: Copyright (c) 2002 Øyvind Vestavik

Version:
1.0
Author:
Øyvind Vestavik

Field Detail

CONTENT_TYPE
private static final String **CONTENT_TYPE**

Description: Contains the default mime return type of the servlet.

Method Detail

presentContext
private String **presentContext**(Node contextNode)

Description: makes a string containing an html presentation of all elements in an offers context

Parameters:
contextNode - The part of the agreement to be presented
Returns:
a html formatted string to be written to the webbrowser by the calling method

---

presentAsset
private String **presentAsset**(Node assetNode)

Description: makes a string containing a html presentation of all elements in an offers asset description

Parameters:
assetNode - The asset part of the offer to be presented
Returns:
a html formatted string to be written to the webbrowser by the calling method

---

presentRequirements

private String **presentRequirements**(String parentPermission,
boolean local,
Node requirementNode)

Description: makes a string containing a html presentation of a group of requirements in an offer

Parameters:
parentPermission - The parent permission of the requirement
local - a boolean value telling if the requirement to be presented is a global or local requirement
requirementNode - The requirement to be presented
Returns:
a html formatted string to be written to the webbrowser by the calling method

---

presentConstraints

private String **presentConstraints**(boolean local,
Node constraintNode)

Description: makes a string containing a html presentation of a group of constraints in an offer

Parameters:
local - a boolean value telling if this is a local or global constraint
constraintNode - The constraint to be presented
Returns:
a html formatted string to be written to the webbrowser by the calling method

---

presentPermissions

private String **presentPermissions**(Node permissionNode)

Description: presents the permission part of an offer

Parameters:
permissionNode - the permission part of an offer
Returns:
a html formatted string to present the permission part of an offer

---

doPost

public void **doPost**(HttpServletRequest request,
HttpServletResponse response)

Description: Prosesses all http post requests.
Receives the obligations the user is willing to pay, (see doGet()) and illustrates a payment transaction. If the user chooses to "pay" the method is called again and an agreement is created

Parameters:
request -
response -
Throws:
ServletException -
IOException -

---

doGet

public void **doGet**(HttpServletRequest request,
HttpServletResponse response)

Description: Process the HTTP Get request
presents the offer for the material allowing the user to select wich obligations he wants to fulfill

Parameters:
request -
response -
Throws:
ServletException -
IOException -

---

reap.servlets
Class register

public class **register**
extends HttpServlet

Description: This servlet is used for registering new users

Copyright: Copyright (c) 2002 Øyvind Vestavik

Version:
1.0
Author:
Øyvind Vestavik

Field Detail

CONTENT_TYPE
private static final String **CONTENT_TYPE**

Description: The default mimetype of this servlet

Method Detail

printHead
private void **printHead**(PrintWriter out)

Description: Just a utility to write some html that can be called from different places in this servlet

Parameters:
out - A referense to the printwriter

---

printForm
private void **printForm**(PrintWriter out,
String email,
String fname,
String lname)

Description: This method prints a registration form to the Printwriter given as parameter to the method. The method can be called from doGet() which always provide empty Strings as parameters to let the user fill in all fields themselves, and from doPost() in response to a user not filling in all required fields. In order not to force the user to fill in all fields again, the values already filled in is passed to this method and the form is printed to the user containing these values

Parameters:
out - A referense to the printwriter
email - The email adress the user already filled out
fname - The first name the user already filled out

lname - The last name the user already filled out

---

doGet

public void **doGet**(HttpServletRequest request,
HttpServletResponse response)

Description: This method responds to http get requests and just prints a registration page with a registration form

Parameters:
request -
response -
Throws:
ServletException -
IOException -

---

doPost

public void **doPost**(HttpServletRequest request,
HttpServletResponse response)

Description: This method responds to http post requests generated as the user clicks submit on a form generated by the doGet-method. It first forces the user to fill in all mandatory fields and then creates and stores a profile for the user. The user is then presented with a username and password to log in with.

There is a flaw in the functionality here. If a user registers with a username (email adress) that is already registered the old profile stored under this username will simply be written over. It is not a priority task to fix this in this prototype

Parameters:
request - object containing the users request
response - object used for writing back to the user
Throws:
ServletException -
IOException -

---

reap.servlets
Class renderingApplication

public class **renderingApplication**
extends HttpServlet

Description: This servlet functions as a substitute for a renderingApplication and lets the user see if they can be granted a a ticket for the permission they want to exercise according to the agreement they have for access to this material

Copyright: Copyright (c) 2002 Øyvind Vestavik

Version:
1.0
Author:
Øyvind Vestavik

Method Detail

presentTicket

private void **presentTicket**(PrintWriter out,
Vector ticket)

Decription : Utility method that prints the result of the request for a ticket. Prints whether the request was granted or not with an indication of why if the ticket was not granted

Parameters:
out - referance to the PrintWriter
ticket - a Vector representing the ticket. The ticket contains information on wether the ticket has been granted or not, and if not, the reason why it could not be granted.

---

doGet

public void **doGet**(HttpServletRequest request,
HttpServletResponse response)

Description: Prosesses all http get requests from the browser
Showing an message first the first time it is called, then presenting the substitute renderingApplication on subsequent calls

Parameters:
request -
response -
Throws:
ServletException -
IOException -

Package Summary

Package reap.DRMPackager

This package contains only one class. The class is put into a separat package to allow for grouping with future classes that extends the DRMPackager or utility methods of the current class.

DRMPackager

reap.DRMPackager
Class DRMPackager

public class **DRMPackager**

Title: DRMPackager

Description: This is the class acting as the DRMPackager which delivers a Content Package containing the material to the DRM Controller (And in future versions could insert material metadata and rights into the collections

Copyright: Copyright (c) 2002

Version:
1.0
Author:
Øyvind Vestavik

Constructor Detail

DRMPackager
public **DRMPackager**()

Description: Default constructor

Method Detail

getPackage
public DRMPackage **getPackage**(String collectionId,
String itemId,
String userId)

Description: Making a DRMPackage an returning it to the caller

Parameters:
collectionId -
itemId -
userId -
Returns:
DRMPackage

Package Summary

Package reap.DRMController

DRMController

reap.DRMController
Class DRMController

public class **DRMController**

Title: REAP

Description: a Rights Enforcing Access Protocol
All parts of the userinterface interacts with the rest of the system through this class

Copyright: Copyright (c) 2002

Version:
1.0
Author:
Øyvind Vestavik

Constructor Detail

DRMController
public **DRMController**()

Description: Just the constructor

Method Detail

getOffer
public Document **getOffer**(String collectionId,
String itemId)

Description: Just forwards the request from the frontend to get the materials offer to the DRMLicencegenerator

Parameters:
collectionId - a positive integer identifing the collection the item is in
itemId - a positive integer identifing a resource witin a collection
Returns:
a DOM Document containing the initial offer made by the rightsholder(s) describing which permissions are offered for this resource
Throws:
Exception - Any exceptions thrown in this method or in methods called by this method is thrown back to the calling class to be handled there.

---

getTicket
public Vector **getTicket**(String collectionId,
String itemId,
String permission,
String username,

String IPAdress)

Description: A ticket is the permission to exercise a permission once. Users with an agreement can request a ticket for a permission on a resource calling this method. The method will try to obtain a ticket on behalf of the user from the DRMLicenceGenerator

Parameters:
collectionId - a positive integer identifing the collection the item is in
itemId - a positive integer identifing a resource witin a collection
permission - the permission the user wants to access
username - the registered username of the user
IPAdress - the IPAdress of the users current connection

Returns:
A vector containing strings on the form of key-value pairs telling if the ticket could be granted or not and why not.

Throws:
Exception - Any exceptions thrown in this method or in methods called by this method is thrown back to the calling class to be handled there.

---

materialExists

public boolean **materialExists**(String collectionId,
String itemId,
String username)

Description: based on collectionId and itemId checks to see if the material can be retrieved/exists. Username shouldn't bee needed here but thats how it works here, because this method makes a DRMPackage and checks to see if it contains material

Parameters:
collectionId - a positive integer identifing the collection the item is in
itemId - a positive integer identifing a resource witin a collection
username - the registered username of the user

Returns:
boolean value (true||false) to indicate if the material exists

---

checkForAgreement

public boolean **checkForAgreement**(String collectionId,
String itemId,
String username)

Description: based on collectionId, itemId and username checks to see if the user has an agreement for this material. Request is just forwarded to DRMLicenceGenerator

Parameters:
collectionId - a positive integer identifing the collection the item is in
itemId - a positive integer identifing a resource witin a collection
username - the registered username of the user

Returns:
boolean value (true||false) to indicate if the user has an agreement for this material.

Throws:
Exception - Any exceptions thrown in this method or in methods called by this method is thrown back to the calling class to be handled there.

---

makeAgreement

public void **makeAgreement**(String username,
String collectionId,
String itemId,
Vector fullfilledReq)

Description: Receives a request to make an agreement over an material. Forwards this request to the DRMLicenceGenerator. If the agreement could not be created, an exeption is thrown

Parameters:

username - the registered username of the user
collectionId - a positive integer identifing the collection the item is in
itemId - a positive integer identifing a resource witin a collection
fullfilledReq - Vector of formatted strings indicating what requirements the user has fullfilled.

Throws:

Exception - Any exceptions thrown in this method or in methods called by this method is thrown back to the calling class to be handled there.

Package Summary

Package reap.DRMLicenceGenerator

DRMLicenceGenerator

reap.DRMLicenceGenerator
Class DRMLicenceGenerator

public class **DRMLicenceGenerator**

Title: REAP

Description: a Rights Enforcing Access Protocol

Copyright: Copyright (c) 2002

Company:

Version:
1.0
Author:
Øyvind Vestavik

Constructor Detail

DRMLicenceGenerator
public **DRMLicenceGenerator**()

Description: Default constructor

Method Detail

getOffer
public Document **getOffer**(String collectionId,
String itemId)

Description: This method contacts the database frontend for the database containing the offers, agreements and userprofiles and retrieves the offer for the material

Parameters:
collectionId - positive integer telling which collection the resource is in
itemId - positive integer identifying the resource within the collection
Returns:
a DOM Document containing the offer from the rightsholder(s)
Throws:
Exception - Any exceptions thrown by this method og methods called by it are thrown back to the caller to be handled there

---

checkForAgreement
public boolean **checkForAgreement**(String collectionId,
String itemId,
String username)

Description: This method contacts the database frontend for the database containing the offers, agreements and userprofiles and checks if the user has an agreement for this material

Parameters:

collectionId - positive integer telling which collection the resource is in
itemId - positive integer identifying the resource within the collection
username - The registered username for the user

Returns:

boolean value indicating if the user has an agreement for the resource

Throws:

Exception - Any exceptions thrown by this method og methods called by it are thrown back to the caller to be handled there

---

generateAgreementContext

private void **generateAgreementContext**(Element agreement,
Document Agreement)

Description: just a helper method for the generateAgreement method.
Receives the agreement Element and the DOM Document and appends a context to it containing the generation date of the agreement, no return value needed

Parameters:

agreement - the xml Element to be appended with context
Agreement - the reference to the DOM Document the agreement Element is in

---

generateAgreementAsset

private void **generateAgreementAsset**(Element agreement,
Document Agreement,
String collectionId,
String itemId,
Node offerAsset)

Description: just a helper method for the generateAgreement method.
Adds the asset description of the offer to the agreement in making

Parameters:

agreement - the xml Element to be appended with asset description
Agreement - the reference to the DOM Document the agreement Element is in
collectionId - the collectionId of the material the agreement is about
itemId - the itemId of the material the agreement is about
offerAsset - the asset description of the offer

---

checkRequirement

private boolean **checkRequirement**(Vector reqFullfilled,
Node requirement,
String parentPermission)

Description: Just a helper method for the generateAgreement method.
Receives a requirement from the offer and a list of requirements that the user has fullfilled and checks to see if the requirement is in the list of requirements the user has fullfilled. If the

requirement has been fullfilled the method returns true, else false. The parent permission parameter is required to make the comparison.

Parameters:
reqFullfilled - A vektor of formatted strings telling which requirements the user has fullfilled
requirement - the requirement from the offer to be evaluated
parentPermission - the parent permission of the requirement to be evaluated
Returns:
boolean value indicating if the user has fullfilled this requirement

---

copyConstraint

private void **copyConstraint**(Document Agreement,
Node offerConstraintNode,
Element agreementConstraintElement,
String parentPermission)

Description: Just a helper method for the generateAgreement method.
Since constraints are not evaluated in the process of making an agreement any constraints in the offer are just copied to the agreement so that they can be evaluated when the user requests a ticket based on his agreement

Parameters:
Agreement - reference to the agreement in making
offerConstraintNode - the constraint from the offer to be added to the agreement
agreementConstraintElement - The Element of the agreement that is to be extended with a constraint
parentPermission - The Element representing the permission to be given a constraint in the agreement

---

generateAgreementPermissions

private void **generateAgreementPermissions**(Element agreement,
Document Agreement,
Node offerPermission,
Vector fullfilledReq)

Description: Just a helper method for the generateAgreement method.
Generates the permission part of the agreement based on the permission part of the offer. Evaluates the global and local requirements and limits the permissions to those for which the requirements (global and local) has been satisfied. Global and local constraints are just copied into the agreement here. They are evaluated when the user tries to access an permission in this agreement

Parameters:
agreement - The agreement Element of the agreement
Agreement - Reference to the agreement in making
offerPermission - The Element representing the permission part of the offer
fullfilledReq - a vector containing the requirements the user has fullfilled

---

calculatePayment

private Vector **calculatePayment**(Vector fullfilledReq)

Description :Just a helper method for the generateAgreementParty method.
Summarizes the payments the user has opted to pay

Parameters:
fullfilledReq - The requirements the user has fullfilled
Returns:
a vector with a totalPayment and a currency

---

generateAgreementParty

private void **generateAgreementParty**(Element agreement,
String userName,
String collectionId,
String itemId,
Vector fullfilledReq,
Node offerParty,
Document Agreement)

Description: Just a helper method for the generateAgreement method.
Generates the party part of the agreement, adding the user element and copying the rightsholder elements from the offer to the agreement. Prints to a file the details of the agreement created needed for later distribution of attribution to the rightsholders by the system

Parameters:
agreement - the agreement element of the agreement under wich the party part of the agreement is to be added
userName - The registered username of the user making the agreement
collectionId - The collection the resource is in
itemId - The identifier for the item in the collection
fullfilledReq - The requirements the user has fullfilled
offerParty - The party part of the offer
Agreement - a reference to the agreement in making

---

generateAgreement

public void **generateAgreement**(String username,
String collectionId,
String itemId,
Vector fullfilledReq)

Description: generates an agreement and stores it in a database

Parameters:
username - the registered username of the user making the agreement
collectionId - The collection the resource is n
itemId - The identifier for the item in the collection
fullfilledReq - The requirements the user has fullfilled
Throws:
Exception - Any exceptions is thrown back to the user to be handled there

---

evaluateConstraints

private boolean **evaluateConstraints**(Node constraintNode,
String username,

String IPAdress,
Vector ticket,
String parentPermission)

Description: Helpermethod for the generateTicket method
Evaluates if a constraint can be met by the user taking into consideration the users profile and his current connection tp the Internet

Parameters:

constraintNode - The constraint from the agreement to be evaluated
username - The registered username of the user
IPAdress - The IPadress of the users current Internet connection
ticket - The vector to be given an error message if the constraint can't be met.
parentPermission - The parent permission of the constraint to be evaluated

Returns:

boolean value indication if the constraint has been met

---

generateTicket

public Vector **generateTicket**(String collectionId,
String itemId,
String permission,
String username,
String IPAdress)

Description: generates a ticket which gives the user access to a permission if all constraints of the agreement are met.

Parameters:

collectionId - The collectionId of the resource
itemId - The itemId of the resource
permission - The permission the user wants to exercise
username - The registered username of the user
IPAdress - The IPAdress of the users current Internet connection

Returns:

a vector implementing the ticket

Throws:

Exception - Any exeption is thrown back to the calling class to be handled there

---

Package Summary

Package reap.util

DRMPackage
FileEncapsulator
ProfileHandler
UsernamePwdHolder

reap.util
Class DRMPackage

public class **DRMPackage**

Description: a DRMPackage contains the material and the rights of a material. In this prototype this is just an encapsulator for the material and some information about itself and its material

Copyright: Copyright (c) 2002

Company:

Version:
1.0
Author:
Øyvind Vestavik

Field Detail

containsMaterial
public boolean **containsMaterial**

Description: indicates if this package contains any material

---

rights
public Document **rights**

Description: The rightsmodel for the material. In this prototype this field is not in use

---

material
public FileEncapsulator **material**

Description: This field contains the material itself as a sequence of bits

---

fileSize
int **fileSize**

Description: This field indicates the size of the file in bits

---

fileName
String **fileName**

Description: This field contains the filename of the resource

reap.util
Class FileEncapsulator

public class **FileEncapsulator**

Description: This class encapsulates a resource/file and some information about the file

Copyright: Copyright (c) 2002

Company:

Version:
1.0
Author:
Øyvind Vestavik

Field Detail

fileName
public String **fileName**

Description: This field contains the filename of the resource

---

fileSize
public int **fileSize**

Description: This field contains the filesize of the resource

---

content
public int[] **content**

Description: This array contains the contents of the file as an array of bytes

---

reap.util
Class ProfileHandler

public class **ProfileHandler**

Title:

Description: This class receives user info from registration servlet and makes a profile in the form of an xml DOM tree for the user. It generates a password to insert it along with the username in the profile and sends the profile to the xmlDataBaseFrontEnd class to be stored there. Then the generated password is returned to the registraiton servlet

Copyright: Copyright (c) 2002 Øyvind Vestavik

Version:
1.0
Author:
Øyvind Vestavik

Method Detail

generate_pwd
private String **generate_pwd**()

Description: generates an eight character long password

Returns:
String pwd. The generated password

---

make_profile
public UsernamePwdHolder **make_profile**(String email,
String fname,
String lname)

Description: Receives details about a user collected by servlet reap.servlets.register and creates a profile in the database. Returns an object containing the status of the delivered service and the generated password

Parameters:
email - The users email adress. Functions as username
fname - First name of the user
lname - Last name of the user
Returns:
UsernamePwdHolder
Throws:
Exception -

---

reap.util
Class UsernamePwdHolder

public class **UsernamePwdHolder**

Title:

Description: Just a keeper for a username and a password with an optional message field that can be used for passing messages between classes

Copyright: Copyright (c) 2002

Company:

Version:
1.0
Author:
Øyvind Vestavik

Field Detail

username
private String **username**

Description: contains the users username

---

password
private String **password**

Description: contains the users password

---

message
private String **message**

Description: contains an optional message

Constructor Detail

UsernamePwdHolder
public **UsernamePwdHolder**()

Description: Default constructor

UsernamePwdHolder

public **UsernamePwdHolder**(String username,
String password,
String message)

Description: This contstruktor receives the users username, password and optionally a message to be stored in the object

Parameters:
username -
password -
message -

Method Detail

getUsername

public String **getUsername**()

Description: getMethod for username

Returns:
username

---

getPassword

public String **getPassword**()

Description: getMethod for password

Returns:
password

---

getMessage

public String **getMessage**()

Description: getMethod for message

Returns:
message

---

setUsername

public void **setUsername**(String username)

Description: setMethod for username

Parameters:
username -

---

setPassword

public void **setPassword**(String password)

Description: setMethod for password

Parameters:

password -

---

setMessage

public void **setMessage**(String message)

Description: setMethod for message

Parameters:

message -

Package Summary

Package reap.util.database.resource

IdentifierResolver
resourceDatabaseFrontend

reap.util.database.resource
Class IdentifierResolver

public class **IdentifierResolver**

Description: This class contains functionality for converting from a REAP identification scheme containing a CollectionId and ItemId to a physical location.

Copyright: Copyright (c) 2002

Version:
1.0
Author:
Øyvind Vestavik

Method Detail

getPhysicalLocation

```
public String getPhysicalLocation(String collectionId,
                                  String itemId)
```

Description: Takes a collectionId and an itemId and returns the physical location from where the resource can be retrieved

Parameters:
`collectionId` -
`itemId` -
Returns:
A String containing a filepath to the material

reap.util.database.resource
Class resourceDatabaseFrontend

public class **resourceDatabaseFrontend**

Description: This class is responible for collecting the material the user is requesting from a physical storage. It gets the location of the storage from the IdentifierResolver class

Copyright: Copyright (c) 2002

Version:
1.0
Author:
Øyvind Vestavik

Method Detail

getMaterial

```
public FileEncapsulator getMaterial(String collectionId,
                                    String itemId)
```

Retrieves the resource from its physical location and packs it into a reap.util.FileEncapsulator object before returning this object to the caller

Parameters:
`collectionId` -
`itemId` -
Returns:
reap.util.FileEncapsulator

Package Summary

Package reap.util.database.xml

xmlDatabaseFrontend

reap.util.database.xml
Class xmlDatabaseFrontend

public class **xmlDatabaseFrontend**

Description:

This class is responsible for storing and retrieving xml documents like profiles, Offers and Agreements from an Xindice database instance.

flaw: There is no check on existence of profiles. If a user registers using an already registered mail adress, the old profile stored under this username will simply be written over..

Copyright: Copyright (c) 2002. Øyvind Vestavik

Version:
1.0
Author:
Øyvind Vestavik

Method Detail

openDB

```
private Collection openDB(String collectionName)
```

Description: Opens a collection in the xml Database based on the parameter collectionName. This method is a local method not required by the xmlDatabaseFrontEndInterface

Parameters:
`collectionName` - The name of the collction to be opened
Returns:
col a reference to the collection opened
Throws:
`Exception` - If collection can't be opened, an Exception is thrown back to the caller for handeling there

---

get_Profile

```
public Document get_Profile(String username)
```

Description: retrieves the users profile from a database and returns it as an Dom Document.

Parameters:
`username` - : Identifies which profile to retrieve
Returns:
A Document (xml DOM tree) with all information stored about a user.
Throws:
`Exception` -

---

store_Profile

```
public void store_Profile(String username,
                          Document profile)
```

Description: Stores a profile for a user

Parameters:

`username` - : Identifies the profile
`profile` - : Dom Document with all info about a user

Throws:

`Exception` -

---

get_Agreement

```
public Document get_Agreement(String username,
                              String collectionId,
                              String itemId)
```

Description: Returns an Agreement already negotiated governing the allowed consumption of a given material by a given user

Parameters:

`username` - : Indicates the user that made the agreement
`collectionId` - : Indicates the collection the Item is in.
`itemId` - : Indicates the Item the agreement is about

Returns:

A Dom Document containing the rights the user has obtained over this material including the constraints and obligations these rights are given and can be executed under.

Throws:

`Exception` -

---

agreement_exists

```
public boolean agreement_exists(String username,
                                String collectionId,
                                String itemId)
```

Description : Receives an identification of the user (username) and an identification of the material (collectionId and itemId) and returns true if the user has an agreement or else returns false
problem: Can throw local exeption if db not started. Not visible outside method...

Parameters:

`username` -
`collectionId` -
`itemId` -

Returns:

boolean value telling if the resource exists in the database

Throws:

`Exception` - Throws back exceptions from the database to the calling class to be handled there

---

store_Agreement

```
public void store_Agreement(String username,
```

```
                                String collectionId,
                                String itemId,
                                Document Agreement)
```

Description: Stores an agreement governing how a user may consume a material based on a received xml DOM Document.

Parameters:

`username` - the user that has made the agreement
`collectionId` - The collection of the Item
`itemId` - The number of the item within the collection
`Agreement` - A dom Document containing the actual agreement

Throws:

`Exception` -

---

update_Agreement

```
public void update_Agreement(String username,
                             String collectionId,
                             String itemId,
                             Document new_Agreement)
```

Description: Simply removes the old Agreement from the db and stores the new version under the same key as the old one

Parameters:

`username` - email/username for the user who has generated the agreement
`collectionId` - Together with itemId the identifier for the material in question
`itemId` - Together with collectionId the identifier for the material in question
`new_Agreement` - The updated version of the agreement

Throws:

`Exception` -

---

get_Offer

```
public Document get_Offer(String collectionId,
                          String itemId)
```

Description: Returns an Offer for an item identified by a combination of collectionId and itemId

Parameters:

`collectionId` - : Indicates the collection the Item is in.
`itemId` - Indicates the Item in the collection that the offer is about

Returns:

Offer A DOM Document with information about rights over the material that can be given to users and the conditions and constraints for doing so

Throws:

`Exception` -

---

store_Offer

```
public void store_Offer(String collectionId,
                        String itemId,
```

```
                        Document Offer)
```

Does : stores an ODRL Offer for a given Item/Material This method will not be implemented in this prototype. The declaration is included for consistency and to imply how this prototype can be extended to support the making of ODRL offers and publishing of material in possible future versions.

Parameters:

`collectionId` - : Indicating which collection the Item is in

`itemId` - : Indicating the item in question

`Offer` - : The Offer to be stored (Dom Document)

Throws:

`Exception` -

---

update_Offer

```
public void update_Offer(String collectionId,
                         String itemId,
                         Document new_Offer)
```

Description: This method is just taken in for clarity. It will not be used in version one of this prototype. Simply removes the old Offer from the db and stores the new version under the same key as the old one

Parameters:

`collectionId` -

`itemId` -

`new_Offer` -

Throws:

`Exception` -

---

checkLogin

```
public int checkLogin(String username,
                      String pwd)
```

Does : Receives a username and a password and checks if there is a matching profile stored in the database. If such a profile is found the method returns 1, else it returns 0. Returned value 2 means something is wrong, (two or more users with same username and password) but this should never occure.

Parameters:

`username` - Username entered by user

`pwd` - password entered by user

Returns:

A Boolean being true if user exist and has this pwd, else returns false

Throws:

`Exception` -

Appendix D: Javadoc Dokumentasjon for prototypen REAP

Appendix E

Kildekode for REAP

Dette er kildekoden for alle klasser som utgjør prototypen REAP.
Før alle klasser er det angitt en overskrift som gir pakketilhørighet og filnavn for fila som er gjengitt.

Reap.servlets.get_Document.java

```
package reap.servlets;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.net.*;
import org.w3c.dom.*;
import reap.DRMController.DRMController;

/**
 * <p>Description: This servlet is the starting point for users wanting access
 * to material protected by REAP and should be called with http get
 * parameters in the link to the servlet to determine which material the user is
 * seeking access to. An example of such a url would be<br>
 * <i>http://www.myserver.com[:portnumber]/reap/get_document?CollectionId=4&ItemId=8</i></p>
 *
 * <p>Copyright: Copyright (c) 2002 Øyvind Vestavik</p>
 *
 * @author Øyvind Vestavik
 * @version 1.0
 */

public class get_Document extends HttpServlet {

  /**
   *<p>Description: Contains the default mime return type of the servlet.
   */
  private static final String CONTENT_TYPE = "text/html";

/*****************************************************************************/

  /**
   * <p>Title: method doGet</p>
   * <p>Description: This method receives the CollectionId and ItemId for the
   * document the user want access to in the "request object". (See the
   * overall description of the class). If CollectionId or ItemId is missing, an
   * errorpage is written. Users must be logged in to get access to
   * the services of this method and are redirected to the login servlet
   * if not logged in. Makes session to store details of request to be retrieved
   * later. If the user is logged in, checks to see if the user has an agreement
   * for the material he is requesting. If not, the user is redirected to the
   * makeAgreement servlet in order to have such an agreement. If the agreement
   * generation succeeds the user is thrown back to this servlet, which starts
   * up the rendering application
   *
   * @param request An object Containing all details of the request
   * @param response An object used for sending information and directives back
   *                 to the users browser
   * @throws ServletException
   * @throws IOException
   */
  public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    //making a printer to write to internet browser
    PrintWriter out = response.getWriter();
    //retrieves (or creates if doesn't exist) the users session
    HttpSession session = request.getSession(true);

    String collectionId = request.getParameter("CollectionId");
    String itemId = request.getParameter("ItemId");

    //checks for valid arguments
    if ((collectionId==null)||(itemId==null)){
```

```
    response.setContentType("text/plain");
    out.println("Error: Malformed URL");
    out.println("          URL must contain a CollectionId and an ItemId");
    out.println("Example of wellformed URL:");
    out.println("          https://"+request.getServerName()+":8443/REAP/get_Document?CollectionId=4&ItemId=3");
  }
  else{
    //unbinds evt previous values and binds parameters to session varables
    session.removeAttribute("CollectionId");
    session.removeAttribute("ItemId");
    session.setAttribute("CollectionId", collectionId);
    session.setAttribute("ItemId", itemId);
    //If the session has no username, the user is not logged in and is
    //redirected to the login page
    if (session.getAttribute("username")== null){ //not logged in
      //creating a url for the login_servlet. This url should be independent
      //of servername, portnumber and contextPath, but is not...
      String filePath = request.getContextPath() + "/login";
      URL url = new URL("http", request.getServerName(),8080, filePath); //portnumber is fixed...
      //sending the user to the created url
      response.sendRedirect(url.toString());
    }
    else { //user is logged in. Let the fun begin

      String username = (String) session.getAttribute("username");

      response.setContentType("text/html");
      out.println("<p align=ŀight'>You are logged in as: ");
      out.println(username+"</p><br>");
      out.println(" <center>");
      out.println("<h1>Welcome to REAP</h1>");
      out.println("You have requested a document identified by: <br>");
      out.println("CollectionId: "+collectionId+" <br>");
      out.println("ItemId: "+itemId+"<br><br>");
      out.println("</center>");

      //finds out if the resource exists
      out.println("The system now checks to see if the resource exists..<br>");
      DRMController controller = new DRMController();
      boolean exists = controller.materialExists(collectionId,itemId,username);
      if (exists){
        out.println("<blockquote>");
        out.println(" Resource exists<br>");
        out.println("</blockquote>");

        //checking for agreement
        out.println("The system now checks to see if you already have an");
        out.println("agreement for access to this material<br>");
        boolean gotAgreement = false;
        try{
          gotAgreement =
              controller.checkForAgreement(collectionId, itemId, username);
        }
        catch(Exception e){
          out.println("<blockquote>");
          out.println("Sorry. Because of an internal error we are not able");
          out.println("to determine if you already have an agreement for ");
          out.println("access to this material. We regret that we cannot give");
          out.println("access to the material at this time");
          out.println("</blockquote>");
        }

        if (!gotAgreement){
          out.println("<blockquote>");
          out.println(" You don't have an agreement, and will be redirected");
          out.println(" to a negotiation page");
          out.println("</blockquote>");

          //redirect to negotiate agreement Servlet..
          //creating a url for the get_Document servlet.
```

```
          //This url should be independent
          //of servername, portnumber and contextPath, but is not...
          String filePath = request.getContextPath() + "/makeagreement";
          URL url = new URL("http", request.getServerName(),8080, filePath);
          //sending the user to the created url
          response.sendRedirect(url.toString());
        }
        else{
          out.println("<blockquote>");
          out.println(" You have an agreement");
          out.println("</blockquote>");

        //launch the viewer application
        out.println("The system will now launch the viewer application for you<br>");
        String filePath = request.getContextPath() + "/renderingapplication";
        URL url = new URL("http", request.getServerName(),8080, filePath);
        //sending the user to the created url
        response.sendRedirect(url.toString());
        }
      }
      else{
        out.println("<blockquote>");
        out.println("Sorry. The resource you requested was not found on ");
        out.println("this Server");
        out.println("</blockquote>");
      }
     }
    }
    out.close();
  }
}
```

reap.servlets.login.java

```
package reap.servlets;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.net.*;
import reap.util.database.xml.xmlDatabaseFrontend;

/**
 * <p>Description: This servlet is used for logging in users</p>
 *
 * <p>Copyright: Copyright (c) 2002 Øyvind Vestavik</p>
 *
 * @author Øyvind Vestavik
 * @version 1.0
 */


public class login extends HttpServlet {

  /**
   *<p>Description: Contains the default mime return type of the servlet.
   */
  private static final String CONTENT_TYPE = "text/html";

/******************************************************************************/

  /**
   * <p>Description: Presents the user with a login page. The page also contains
   * a link to the registration servlet</p>
   *
```

```
   * @param request an object containing details of the request
   * @param response an object representing this servlets response
   * @throws ServletException
   * @throws IOException
   */
  public void doGet(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {

    //collects session created in get_Document_servlet
    HttpSession session = request.getSession(true);
    String collectionId = (String) session.getAttribute("CollectionId");
    String itemId = (String)session.getAttribute("ItemId");

    //prints out a login page
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<HEAD>");
    out.println("<TITLE> REAP Login </TITLE>");
    out.println("<META NAME='Generator' CONTENT='EditPlus'>");
    out.println("<META NAME='Author' CONTENT='Øyvind Vestavik'>");
    out.println("<META NAME='Description' CONTENT='This is the page used by user to log in to the system'/>");
    out.println("</HEAD>");
    out.println("<BODY>");
    out.println(" <center>");
    out.println("   <h1>Welcome to REAP</h1>");
    out.println("   <B>REAP is a Rights Enforcing Access Protocol used to protect material from unauthorized access by end
users.<br> To get access to the documents protected by this protocol you have to log in to the system.</B>");
    out.println("   <br><br>");
    out.println("   You have requested a document identified by: <br>");
    out.println("   CollectionId: "+collectionId+" <br>");
    out.println("   ItemId: "+itemId+"<br><br>");
    out.println(" </center>");
    out.println(" <!-- Login form for registered users -->");
    out.println(" <center>");
    out.println("   <B>If you already have a username and password, then fill in the first form and press submit</B> ");
    out.println("   <form method='post' action='login'>");
    out.println("     Username <br> <input name='username' type='text'><br>");
    out.println("     Password <br> <input name='pwd' type='password'><br><br>");
    out.println("     <input type='reset' value='Reset'>  <input type='submit' value='Submit'>");
    out.println("   </form>");
    out.println("   <B>If you don't have a username and password, then click the link below to register</B><BR> ");
    //making link to registration page
    String filePath = request.getContextPath() + "/register";
    URL url = new URL("http", request.getServerName(),8080, filePath); //portnumber is fixed...
    out.println("   <A href="+url.toString()+">Register here</A>");
    out.println(" </center>");
    out.println("</BODY>");
    out.println("</HTML>");
    out.close();
  }

/*****************************************************************************************/

  /**
   * <p>Description: Receives username and password from the form generated by
   * doGet() and tries to log in the user. If the login fails, the the user is
   * redirected back to the loginpage (doGet method of this servlet) </p>
   *
   * @param request
   * @param response
   * @throws ServletException
   * @throws IOException
   */
  public void doPost(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException{
    //collects the entered username and password
    String username = request.getParameter("username");
    String pwd = request.getParameter("pwd");

    //collects session created in get_Document_servlet
```

Appendix E: Kildekode for REAP

```
HttpSession session = request.getSession(true);
String collectionId = (String) session.getAttribute("CollectionId");
String itemId = (String)session.getAttribute("ItemId");

//Checking for valid user
try{
  //making instance of class with database functionality and checks for valid user
  xmlDatabaseFrontend xmldb = new xmlDatabaseFrontend();
  int result = xmldb.checkLogin(username, pwd);

  //evaluating response
  switch (result){
    case 0: { //no such user/password found
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      out.println("<HEAD>");
      out.println("<TITLE> REAP Login </TITLE>");
      out.println("<META NAME='Generator' CONTENT='EditPlus'>");
      out.println("<META NAME='Author' CONTENT='Øyvind Vestavik'>");
      out.println("<META NAME='Description' CONTENT='This is the page shown to users who ");
      out.println("entered an invalid username and/or password'/>");
      out.println("</HEAD>");
      out.println("<BODY>");
      out.println("<center>");
      out.println("<h1>Welcome to REAP</h1>");
      out.println("<B>REAP is a Rights Enforcing Access Protocol used to protect ");
      out.println("material from unauthorized access by end users.<br> To get access ");
      out.println("to the documents protected by this protocol you have to log in");
      out.println("to the system.</B>");
      out.println("<br>");
      out.println("<br>");
      out.println("You have requested a document identified by: <br>");
      out.println("CollectionId: "+collectionId+" <br>");
      out.println("ItemId: "+itemId+"<br><br>");
      out.println("</center>");
      out.println("<!-- Login form for registered users -->");
      out.println("<center>");
      out.println("<font color='red'><B>You entered an invalid username and/or password. Please try again</B></font> ");
      out.println("<form method='post' action='login'>");
      out.println("Username <br> <input name='username' type='text'><br>");
      out.println("Password <br> <input name='pwd' type='password'><br><br>");
      out.println("<input type='reset' value='Reset'>  <input type='submit' value='Submit'>");
      out.println("</form>");
      out.println("<B>If you don't have a username and password, then click the link below to register</B><BR> ");
      //making link to registration page
      String filePath = request.getContextPath() + "/register";
      URL url = new URL("http", request.getServerName(),8080, filePath); //portnumber is fixed...
      out.println("<A href="+url.toString()+">Register here</A>");
      out.println("</center>");
      out.println("</BODY>");
      out.println("</HTML>");
      out.close();
      break;
    }
    case 1: { //user found
      //sending user back to starting page (get_Document_Servlet) with original parameters
      session.setAttribute("username", username);
      String filePath = request.getContextPath() + "/get_document?CollectionId="+collectionId+"&ItemId="+itemId;
      URL url = new URL("http", request.getServerName(),8080, filePath);
      //response.sendRedirect("get_Document");
      response.sendRedirect(url.toString());
      break;
    }
    case 2: { //Database error. Most likely more than one profile returned. Should never occure
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      out.println("<HTML>");
      out.println("<HEAD>");
      out.println("<TITLE>temp</TITLE>");
      out.println("</HEAD>");
```

```
        out.println("<BODY>");
        out.println("<center>");
        out.println("<FONT COLOR='#FF0000'><FONT SIZE=+2>Error!!!!!</FONT></FONT>");
        out.println("<BR><FONT COLOR='#FF0000'><FONT SIZE=+2>    Your profile");
        out.println("has somehow been corrupted.</FONT></FONT><FONT COLOR='#FF0000'><FONT SIZE=-
1></FONT></FONT>");
        out.println("<P><FONT COLOR='#FF0000'><FONT SIZE=+2>    Contakt System");
        out.println("Administrator for help</FONT></FONT><FONT COLOR='#FF0000'><FONT SIZE=+2></FONT></FONT>");
        out.println("<P><FONT COLOR='#FF0000'><FONT
SIZE=+2>            ");
        out.println("</FONT><FONT SIZE=-1>Error: The system found more than one profile for");
        out.println("your username</FONT></FONT>");
        out.println("</center>");
        out.println("</BODY>");
        out.println("</HTML>");
        out.close();
       }
     }//switch

     //explicitly closing reference to databaseclass.
     xmldb = null;
    }
    catch (Exception e){
     //XMLDBException xmldbe = (XMLDBException)e;
     response.setContentType("text/plain");
     PrintWriter out = response.getWriter();
     out.println("We are sorry, but due til an iternal error, we are not ");
     out.println("able to log you on right now");
     out.close();
    }
  }
}
```

reap.servlets.makeAgreement

```
package reap.servlets;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.net.URL;
import reap.DRMController.DRMController;
import org.w3c.dom.*;

/**
 * <p>Description: This servlet is used for interaction with the user in
 * the process of making an agreement</p>
 *
 * <p>Copyright: Copyright (c) 2002 Øyvind Vestavik</p>
 *
 * @author Øyvind Vestavik
 * @version 1.0
 */


public class makeAgreement extends HttpServlet {

  /**
   *<p>Description: Contains the default mime return type of the servlet.
   */
  private static final String CONTENT_TYPE = "text/html";

/******************************************************************************/

  /**
```

```
  * <p>Description: makes a string containing an html presentation of all
  * elements in an offers context</p>
  *
  * @param contextNode The part of the agreement to be presented
  * @return a html formatted string to be written to the webbrowser by the calling method
  */
 private String presentContext (Node contextNode){

  String info = "<B>Information about the offer:</B><br>";
  String attributeString="";
  NodeList nodeList = null;
  int i,j;
  Node textValue = null;
  String attValue = "";
  String nodeName;
  String nodeValue;

  //traverses the context element directly under the offer tag printing
  //the name value pairs inkl attributes and their values
  nodeList = contextNode.getChildNodes();
  for (i=0;i<nodeList.getLength();i++){
   nodeName = nodeList.item(i).getNodeName();
   if (!nodeName.equals("#text")){
    //gets the nodes name
    nodeName = nodeList.item(i).getNodeName();
    //gets the attribute value pair(s)
    NamedNodeMap attributes = nodeList.item(i).getAttributes();
    for (j=0; j<attributes.getLength(); j++){
     attributeString = " (" + attributes.item(j).getNodeName()+"=";
     attValue = attributes.item(j).getFirstChild().getNodeValue();
     attributeString = attributeString + attValue + ")";
    }
    //gets the nodes textvalue
    textValue = nodeList.item(i).getFirstChild();
    nodeValue = textValue.getNodeValue();
    //prints to the output string
    info = info + nodeName + attributeString + ":";
    info = info + nodeValue + "<br>";
   }
  }
  return info;
 }

/*************************************************************************/

 /**
  * <p>Description: makes a string containing a html presentation of all
  * elements in an offers asset description</p>
  *
  * @param assetNode The asset part of the offer to be presented
  * @return a html formatted string to be written to the webbrowser by the calling method
  */
 private String presentAsset(Node assetNode){

  String info ="<B>Information about the resource</B><br>";
  String attributeString="";
  NodeList nodeList = null;
  int i,j, k;
  Node textValue = null;
  Node contextNode=null;
  String attValue = "";
  String nodeName;
  String nodeValue;

  //traverses the asset element directly under the offer tag printing
  //the name value pairs inkl attributes and their values
  nodeList = assetNode.getChildNodes();
  for (i=0;i<nodeList.getLength();i++){
   nodeName = nodeList.item(i).getNodeName();
   //skips the reapId and prints just the contextpart of the asset description
```

```
    if (nodeName.equals("context")){
      contextNode = nodeList.item(i);
      nodeList = contextNode.getChildNodes();
      for (k=0; k<nodeList.getLength(); k++){
        nodeName = nodeList.item(k).getNodeName();
        if (!nodeName.equals("#text")){
          //gets the nodes name
          nodeName = nodeList.item(k).getNodeName();
          //gets the attribute value pair(s)
          NamedNodeMap attributes = nodeList.item(k).getAttributes();
          for (j=0; j<attributes.getLength(); j++){
            attributeString = " (" + attributes.item(j).getNodeName()+"=";
            attValue = attributes.item(j).getFirstChild().getNodeValue();
            attributeString = attributeString + attValue + ")";
          }
          //gets the nodes textvalue
          textValue = nodeList.item(k).getFirstChild();
          nodeValue = textValue.getNodeValue();
          //prints to the output string
          info = info + nodeName + attributeString + ":";
          info = info + nodeValue + "<br>";
        }
      }
    }
   }
   return info;
  }
/*****************************************************************************/

  /**
   * <p>Description: makes a string containing a html presentation of a group
   * of requirements in an offer</p>
   *
   * @param parentPermission The parent permission of the requirement
   * @param local a boolean value telling if the requirement to be presented is
   * a global or local requirement
   * @param requirementNode The requirement to be presented
   * @return a html formatted string to be written to the webbrowser by the calling method
   */
  private String presentRequirements (String parentPermission,
                                      boolean local,
                                      Node requirementNode){

    String info = "";
    String nodeName;
    //evaluating the requirementNode.
    NodeList requirementNodeList = requirementNode.getChildNodes();
    for (int i=0; i<requirementNodeList.getLength(); i++){
      nodeName = requirementNodeList.item(i).getNodeName();
      if (nodeName.equals("prepay")){
        Node prepayNode = requirementNodeList.item(i);
        NodeList prepayNodeList = prepayNode.getChildNodes();
        for (int j=0; j<prepayNodeList.getLength(); j++){
          nodeName = prepayNodeList.item(j).getNodeName();
          if (nodeName.equals("payment")){
            Node paymentNode = prepayNodeList.item(j);
            NodeList paymentNodeList = paymentNode.getChildNodes();
            for (int k=0; k<paymentNodeList.getLength(); k++){
              nodeName = paymentNodeList.item(k).getNodeName();
              if (nodeName.equals("amount")){
                Node amountNode = paymentNodeList.item(k);
                //getting the value of amount
                String amount =amountNode.getFirstChild().getNodeValue();
                //getting the currency attribute and its value
                NamedNodeMap attributes = amountNode.getAttributes();
                Node currencyAttribute = attributes.getNamedItem("currency");
                String currencyType = currencyAttribute.getFirstChild().getNodeValue();

                //printing to the output String
                if (parentPermission.equals("global")){
```

```
                info = info + "To get access to any permissions you have to ";
                info = info + "pay "+currencyType+" "+amount+".<br>";
                info = info + "    ";
                //the name of the checkbox which context the checkbox is about
                info = info + "<input type='checkbox' value='checkbox'";
                String name = "name='req=prepay,currency="+currencyType;
                name = name + ",amount="+amount+",parent="+parentPermission+"'";
                info = info + name+">";
                info = info + "Yes, I want access to permissions.";
              }
              else{
                info = info + "To get access to this permission you have to ";
                info = info + "pay "+currencyType+" "+amount+".<br>";
                info = info + "    ";
                //the name of the checkbox which context the checkbox is about
                info = info + "<input type='checkbox' value='checkbox'";
                String name = "name='req=prepay,currency="+currencyType;
                name = name + ",amount="+amount+",parent="+parentPermission+"'";
                info = info + name+">";
                info = info + "Yes, I want access to this permission.";
              }
            }
          }
        }
      }
    }

    if (info.equals("")){
      info = null;
    }
    return info;
  }

/*****************************************************************************/

  /**
   * <p>Description: makes a string containing a html presentation of a group
   * of constraints in an offer</p>
   *
   * @param local a boolean value telling if this is a local or global constraint
   * @param constraintNode The constraint to be presented
   * @return a html formatted string to be written to the webbrowser by the calling method
   */
  private String presentConstraints (boolean local,
                                     Node constraintNode){
    int i,j,k;
    String info = "";
    NodeList nodeList = null;
    String nodeName = null;

    //evaluating constraints
    //info = info + "constraints:<br>";
    //Node constraintNode = nodeList.item(i);
    nodeList = constraintNode.getChildNodes();

    for (j=0; j<nodeList.getLength(); j++){
      Node constraintElementNode = nodeList.item(j);
      nodeName = constraintElementNode.getNodeName();

      if (nodeName.equals("count")){
        //evaluating the count constraint
        NodeList countNodeList = constraintElementNode.getChildNodes();
        for (k=0; k<countNodeList.getLength(); k++){
          String countElementName = countNodeList.item(k).getNodeName();
          if (countElementName.equals("max")){
            //getting the max value
            Node maxNode = countNodeList.item(k);
            String max = maxNode.getFirstChild().getNodeValue();
            if (local){
```

```
          info = info + "This permission ";
          info = info + "can be excercised maximum "+max+" times<br> ";
        }
        else{
          info = info + "Any individual permission ";
          info = info + "can be excercised maximum "+max+" times if not ";
          info = info + "further restricted <br>    ";
          info = info + "under an individual permission<br>";
        }
      }
    }
  }
  if (nodeName.equals("dateRange")){
    //evaluating the dateRange constraint
    NodeList dateRangeNodeList = constraintElementNode.getChildNodes();
    String start = "unbounded";
    String end = "unbounded";
    for (k=0; k<dateRangeNodeList.getLength(); k++){
      String dateRangeElementName = dateRangeNodeList.item(k).getNodeName();
      if (dateRangeElementName.equals("start")){
        //getting the start date
        Node startNode = dateRangeNodeList.item(k);
        start = startNode.getFirstChild().getNodeValue();
      }
      if (dateRangeElementName.equals("end")){
        //getting the end date
        Node endNode = dateRangeNodeList.item(k);
        end = endNode.getFirstChild().getNodeValue();
      }
    }
    if (local){
      info = info + "This permission ";
      info = info + "can be excercised from "+start+" untill ";
      info = info + end + ", inclusive<br>";
    }
    else{
      info = info + "Any individual permission ";
      info = info + "can be excercised from "+start+" untill ";
      info = info + end + ", inclusive, if not further<br>";
      info = info + "    ";
      info = info + "restricted under an individual permission<br> ";
    }
  }
  if (nodeName.equals("individual")){

    Node individualNode = nodeList.item(j);

    if (individualNode.hasChildNodes()){
      String individualValue = individualNode.getFirstChild().getNodeValue();
      if (local){
        info = info + "This permission ";
        info = info + "can be excercised by, and only by, the user ";
        info = info + individualValue+"<br>";
      }
      else{
        info = info + "Any individual permission ";
        info = info + "can be excercised by, and only by, the user ";
        info = info + individualValue+"<br>";
      }
    }
    else{
      if (local){
        info=info+"<font color=red>The system ";
        info=info+"has found that this permission has ";
        info=info+"been restricted to a single <br>    ";
        info=info+"individual. But since no information about ";
        info=info+"this individual is given, no users <br> ";
        info=info+"    will be granted access to this permission.";
        info=info+"</font><br>";
      }
```

```
      else{
        info=info+"<font color=red>The system ";
        info=info+"has found that access to this resource has ";
        info=info+"been restricted to a single <br>    ";
        info=info+"individual. But since no information about ";
        info=info+"this individual is given, no users <br> ";
        info=info+"    will be granted access to any ";
        info=info+"permission.";
        info=info+"</font><br>";
      }
    }
  }
  if (nodeName.equals("network")){
    //evaluating the network constraint
    NodeList networkNodeList = constraintElementNode.getChildNodes();
    for (k=0; k<networkNodeList.getLength(); k++){
      String networkElementName = networkNodeList.item(k).getNodeName();
      //info = info + networkElementName +"<br>";

      if (networkElementName.equals("startIPnr")){
        //getting the start and end Node
        Node startNode = networkNodeList.item(k);
        Node endNode = startNode.getNextSibling().getNextSibling();
        //checking that both startIPnr and endIPnr has values
        boolean startHasValue = startNode.hasChildNodes();
        boolean endHasValue = endNode.hasChildNodes();
        String startIPnrValue;
        String endIPnrValue;
        if (startHasValue && endHasValue){
          //getting the values of start and end Nodes
          startIPnrValue = startNode.getFirstChild().getNodeValue();
          endIPnrValue = endNode.getFirstChild().getNodeValue();
          //printing to the outputString
          if (local){
            info = info + "This permission ";
            info = info + "can be excercised from any computer with an <br> ";
            info = info + "     IP adress within the ";
            info = info + "range limited by "+ startIPnrValue + " and ";
            info = info +  endIPnrValue + ", <br>";
          }
          else{
            info = info + "Any individual permission ";
            info = info + "can be excercised from any computer with an <br> ";
            info = info + "     IP adress within the ";
            info = info + "range limited by "+ startIPnrValue + " and ";
            info = info +  endIPnrValue + ", <br>";
            info = info + "    if not further restricted ";
            info = info + "under an individual permission<br> ";
          }
        }
        else{ //start or end value for the range missing
          if (local){
            info=info+"<font color=red>The system ";
            info=info+"has found that this permission has ";
            info=info+"been restricted to <br>    ";
            info=info+"users connecting from computers with an IP adress ";
            info=info+"within a given range. <br>    ";
            info=info+"But since this range is not ";
            info=info+"fully defined, no users will be given access <br>";
            info=info+"    ";
            info=info+"to this permission<br>";
          }
          else{
            info=info+"<font color=red>The system ";
            info=info+"has found that the access to this resource has ";
            info=info+"been restricted to <br>    ";
            info=info+"users connecting from computers with an IP adress ";
            info=info+"within a given range. <br>    ";
            info=info+"But since this range is not ";
            info=info+"fully defined, no users will be given access <br>";
```

```
            info=info+"    ";
            info=info+"to any permission for this resource<br>";
           }
          }
         }

         if (networkElementName.equals("networkMask")){
          //getting the node
          Node ipMaskNode = networkNodeList.item(k);
          if (ipMaskNode.hasChildNodes()){
           String ipMaskValue=
            ipMaskNode.getFirstChild().getNodeValue();
           //printing to the outputString
           if (local){
            info = info + "This permission can be excercised from any ";
            info = info + "computer with an IP <br>    ";
            info = info + "adress which matches";
            info = info + "the netMask "+ipMaskValue;
           }
           else{
            info = info + "Any individual permission can be excercised ";
            info = info + "from any computer with an IP adress which ";
            info = info + "matches<br>     ";
            info = info + "the netMask "+ipMaskValue+" if not further ";
            info = info + "restricted under an individual permission<br>";
           }
          }
          else{
           if (local){
            info=info+"<font color=red>The system ";
            info=info+"has found that this permission has ";
            info=info+"been restricted to <br>    ";
            info=info+"users connecting from ";
            info=info+"computers with an IP adress which matches a <br> ";
            info=info+"    netMask. But since no ";
            info=info+"information about this netMask is given, <br>    ";
            info=info+"no users will be granted this permission.";
            info=info+"</font><br>";
           }
           else{
            info=info+"<font color=red>The system ";
            info=info+"has found that the access to this resource has ";
            info=info+"been restricted to <br>    ";
            info=info+"users connecting from ";
            info=info+"computers with an IP adress which matches a <br> ";
            info=info+"    netMask. But since no ";
            info=info+"information about this netMask is given, <br>    ";
            info=info+"no users will be granted access to any rights.";
            info=info+"</font><br>";
           }
          }
         }
        }
       }
      }
     }
     if (info==""){
      info=null;
     }
     return info;
    }

/*****************************************************************************/

    /**
     * <p>Description: presents the permission part of an offer
     *
     * @param permissionNode the permission part of an offer
     * @return a html formatted string to present the permission part of an offer
     */
    private String presentPermissions(Node permissionNode){
```

```
String info = "";
NodeList nodeList = null;
int i,j;
String nodeName ="";


//start printing form to the output sting here
info = info + "<form method='post'>";

//getting Strings that presents global constraints and requirement

String globalConstraintString = null;
String globalRequirementString = null;

nodeList = permissionNode.getChildNodes();
for (i=0;i<nodeList.getLength();i++){
  nodeName = nodeList.item(i).getNodeName();

  if (nodeName.equals("constraint")){
    Node globalConstraintNode = nodeList.item(i);
    globalConstraintString = this.presentConstraints(false, globalConstraintNode);
  }

  if (nodeName.equals("requirement")){
    Node globalRequirementNode = nodeList.item(i);
    globalRequirementString =
      this.presentRequirements("global",false, globalRequirementNode);
  }
}

//Printing global constraints
info = info + "<h2>Global constraints</h2>";
info = info + "These constraints applies to all permissions defined below";

if (globalConstraintString==null){
  info = info + "<blockquote>";
  info = info + "No information about global constraints available";
  info = info + "</blockquote>";
}
else{
  info = info + "<blockquote>";
  info = info + globalConstraintString;
  info = info + "</blockquote>";
}

//Printing global requirements
info = info + "<h2>Global requirements</h2>";
info = info + "To get access to any of the permissions defined below these ";
info = info + "requirements all have to be fullfilled";

if (globalRequirementString==null){
  info = info + "<blockquote>";
  info = info + "No information about global requirements available";
  info = info + "</blockquote>";
}
else{
  info = info + "<blockquote>";
  info = info + globalRequirementString;
  info = info + "</blockquote>";
}


//presents permissions with the constraints and requirements that applies
//only to the individual permissions

info = info + "<h2>Permissions offered:</h2>";
info = info + "<blockquote>";

nodeList = permissionNode.getChildNodes();
```

```
for (i=0;i<nodeList.getLength();i++){
 nodeName = nodeList.item(i).getNodeName();

 if (nodeName.equals("display")){

  info = info + "<h3>Display</h3>";
  Node displayNode = nodeList.item(i);
  if (displayNode.hasChildNodes()){
   //traversing the displayNode
   String displayConstraintString=null;
   String displayRequirementString=null;
   NodeList displayNodeList = displayNode.getChildNodes();
   for (j=0; j<displayNodeList.getLength(); j++){
    nodeName = displayNodeList.item(j).getNodeName();
    if (nodeName.equals("constraint")){
     //getting the constraints for this permission
     Node displayConstraintNode = displayNodeList.item(j);
     displayConstraintString =
      this.presentConstraints(true, displayConstraintNode);
    }
    if (nodeName.equals("requirement")){
     //getting the constraints for this permission
     Node displayRequirementNode = displayNodeList.item(j);
     displayRequirementString =
      this.presentRequirements("display", true, displayRequirementNode);
    }
   }
   //printing the constraints and requirements to the info string
   info = info + "<blockquote>";
   info = info + "<h4>Constraints for the Display permission</h4>";
   info = info + "<blockquote>";
   if (displayConstraintString!=null)
    info = info + displayConstraintString;
   else
    info = info + "No constraints defined";
   info = info + "</blockquote>";
   info = info + "</blockquote>";
   //printing requirements for this permission
   info = info + "<blockquote>";
   info = info + "<h4>Requirements for the Display permission</h4>";
   info = info + "<blockquote>";
   if (displayRequirementString != null)
    info = info + displayRequirementString;
   else
    info = info + "No requirements defined";
   info = info + "</blockquote>";
   info = info + "</blockquote>";
  }
 }

 if (nodeName.equals("print")){

  info = info + "<h3>Print</h3>";
  Node printNode = nodeList.item(i);
  if (printNode.hasChildNodes()){
   //traversing the printNode
   String printConstraintString=null;
   String printRequirementString=null;
   NodeList printNodeList = printNode.getChildNodes();
   for (j=0; j<printNodeList.getLength(); j++){
    nodeName = printNodeList.item(j).getNodeName();
    if (nodeName.equals("constraint")){
     //getting the constraints for this permission
     Node printConstraintNode = printNodeList.item(j);
     printConstraintString =
      this.presentConstraints(true, printConstraintNode);
    }
    if (nodeName.equals("requirement")){
     //getting the constraints for this permission
     Node printRequirementNode = printNodeList.item(j);
```

```
        printRequirementString =
          this.presentRequirements("print", true, printRequirementNode);
       }
      }
      //printing the constraints and requirements to the info string
      info = info + "<blockquote>";
      info = info + "<h4>Constraints for the Print permission</h4>";
      info = info + "<blockquote>";
      if (printConstraintString!=null)
        info = info + printConstraintString;
      else
        info = info + "No constraints defined";
      info = info + "</blockquote>";
      info = info + "</blockquote>";
      //printing requirements for this permission
      info = info + "<blockquote>";
      info = info + "<h4>Requirements for the Print permission</h4>";
      info = info + "<blockquote>";
      if (printRequirementString != null)
        info = info + printRequirementString;
      else
        info = info + "No requirements defined";
      info = info + "</blockquote>";
      info = info + "</blockquote>";
     }

    }
    if (nodeName.equals("execute")){

     info = info + "<h3>Execute</h3>";
     Node executeNode = nodeList.item(i);
     if (executeNode.hasChildNodes()){
      //traversing the printNode
      String executeConstraintString=null;
      String executeRequirementString=null;
      NodeList executeNodeList = executeNode.getChildNodes();
      for (j=0; j<executeNodeList.getLength(); j++){
        nodeName = executeNodeList.item(j).getNodeName();
        if (nodeName.equals("constraint")){
          //getting the constraints for this permission
          Node executeConstraintNode = executeNodeList.item(j);
          executeConstraintString =
            this.presentConstraints(true, executeConstraintNode);
        }
        if (nodeName.equals("requirement")){
          //getting the constraints for this permission
          Node executeRequirementNode = executeNodeList.item(j);
          executeRequirementString =
            this.presentRequirements("execute",true, executeRequirementNode);
        }
      }
      //printing the constraints and requirements to the info string
      info = info + "<blockquote>";
      info = info + "<h4>Constraints for the Execute permission</h4>";
      info = info + "<blockquote>";
      if (executeConstraintString!=null)
        info = info + executeConstraintString;
      else
        info = info + "No constraints defined";
      info = info + "</blockquote>";
      info = info + "</blockquote>";
      //printing requirements for this permission
      info = info + "<blockquote>";
      info = info + "<h4>Requirements for the Execute permission</h4>";
      info = info + "<blockquote>";
      if (executeRequirementString != null)
        info = info + executeRequirementString;
      else
        info = info + "No requirements defined";
      info = info + "</blockquote>";
```

```
        info = info + "</blockquote>";
      }
    }

    if (nodeName.equals("play")){

      info = info + "<h3>Play</h3>";
      Node playNode = nodeList.item(i);
      if (playNode.hasChildNodes()){
        //traversing the printNode
        String playConstraintString=null;
        String playRequirementString=null;
        NodeList playNodeList = playNode.getChildNodes();
        for (j=0; j<playNodeList.getLength(); j++){
          nodeName = playNodeList.item(j).getNodeName();
          if (nodeName.equals("constraint")){
            //getting the constraints for this permission
            Node playConstraintNode = playNodeList.item(j);
            playConstraintString =
              this.presentConstraints(true, playConstraintNode);
          }
          if (nodeName.equals("requirement")){
            //getting the constraints for this permission
            Node playRequirementNode = playNodeList.item(j);
            playRequirementString =
              this.presentRequirements("play", true, playRequirementNode);
          }
        }
        //printing the constraints and requirements to the info string
        info = info + "<blockquote>";
        info = info + "<h4>Constraints for the Play permission</h4>";
        info = info + "<blockquote>";
        if (playConstraintString!=null)
          info = info + playConstraintString;
        else
          info = info + "No constraints defined";
        info = info + "</blockquote>";
        info = info + "</blockquote>";
        //printing requirements for this permission
        info = info + "<blockquote>";
        info = info + "<h4>Requirements for the Play permission</h4>";
        info = info + "<blockquote>";
        if (playRequirementString != null)
          info = info + playRequirementString;
        else
          info = info + "No requirements defined";
        info = info + "</blockquote>";
        info = info + "</blockquote>";
      }
    }
  }
  info = info + "</blockquote>";

  info = info + "<input type='hidden' name='status' value='hasShownOffer'>";
  //adding buttons to the form
  info = info + "<input type='reset' value='Reset'>";
  info = info + "<input type='submit' value='Make agreement'>";
  //ending form here
  info = info + "</form>";
  //returning the string containing the presentation of the offer as an form
  return info;
 }

/*************************************************************************/

 /**
  * <p>Description: Prosesses all http post requests. <br>
  * Receives the obligations the user is willing to pay, (see doGet()) and
  * illustrates a payment transaction. If the user chooses to "pay" the method
  * is called again and an agreement is created</p>
```

```
 *
 * @param request
 * @param response
 * @throws ServletException
 * @throws IOException
 */
public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

  //retrieves (or creates if doesn't exist) the users session
  HttpSession session = request.getSession(true);

  //prepairing to write to browser
  response.setContentType(CONTENT_TYPE);
  PrintWriter out = response.getWriter();
  out.println("<html>");
  out.println("<head><title>makeAgreement</title></head>");
  out.println("<body>");

  String username = (String) session.getAttribute("username");
  String collectionId = (String)session.getAttribute("CollectionId");
  String itemId = (String)session.getAttribute("ItemId");

  out.println("<p align='right'>You are logged in as: ");
  out.println(username+"</p><br>");
  out.println(" <center>");
  out.println("<h1>REAP Agreement Negotiator</h1>");

  //Everything above is the same regardless of how this method was called

  boolean makePayment = (request.getParameter("payment_ok")==null);
  if (makePayment){ //payment Transaction to be made
    int totalPayment = 0;

    out.println("<form method='POST'>");
    //getting the parameterNames of the request. Only the names are important
    String attName;
    Enumeration parameterNames = request.getParameterNames();
    while (parameterNames.hasMoreElements()){
      //for all attributes exept the status evaluating
      attName = (String)parameterNames.nextElement();
      if (!(attName.equals("status"))){
        //preserves the attribute over to next call to the servlets doPost
        out.println("<input type='hidden' name='"+attName+"' value='"+attName+"'>");
        //makes presentation of this parameter to browser
        String currType ="";
        String amount ="";
        String permission ="";
        //splitting the attribute up into key=value pairs, split at ","
        String keyValueToken;
        StringTokenizer paramTokenizer = new StringTokenizer(attName,",");
        while (paramTokenizer.hasMoreTokens()){
          //evaluating one key=value pair
          keyValueToken = paramTokenizer.nextToken();
          //splitting into keys and values, split at "="
          StringTokenizer keyValueTokenizer=
                  new StringTokenizer(keyValueToken, "=");
          int tokennr = 1;
          String key ="";
          String value ="";
          while (keyValueTokenizer.hasMoreTokens()){
            //should always have two tokens
            if (tokennr==1){
              key = keyValueTokenizer.nextToken();
            }
            if (tokennr==2){
              value = keyValueTokenizer.nextToken();
            }
            tokennr++;
          }//inner while
```

```
        if (key.equals("currency")){
            currType = value;
        }
        if (key.equals("amount")){
          amount = value;
          Integer integer = new Integer(value);
          totalPayment = totalPayment + integer.intValue();
        }
        if (key.equals("parent")){
          permission = value;
        }
      }
      //printing the parameter to browser
      if (permission.equals("global")){
        out.println("You have opted to pay "+currType+" "+amount);
        out.println(" for access to any permissions<br>");
      }
      else{
        out.println("You have opted to pay "+currType+" "+amount+" for ");
        out.println("access to the "+permission+" permission<br>");
      }
    }
  }//traversing of all parameters done

  if (totalPayment>0){
    //prints out the payments the user has opted to pay.
    out.println("Total amount to pay: "+totalPayment+"<br><br>");
    out.println("<br>");
    out.println("<b>Note that if you fail to pay for a defined global requirement<br>");
    out.println("you will not be granted any permission even if you pay for <br>");
    out.println("access to an individual permission. Hit back in your browser<br> ");
    out.println("if you want to check this.</b><br><br>");

    out.println("<i>The form is just an illustration of a payment. <br>");
    out.println("No actual transaction will be made</i><br><br>");

    out.println("Enter your visa card number in the textboks below and press <br>");
    out.println("Pay to generate an Agreement<br><br>");

    //printing the rest of the form
    out.println("<input type='hidden' name='payment_ok' value='yes'>");
    out.println("<input type='text' name='dummy'><br>");
    out.println("<input type='submit' value='Pay and Generate Agreement'>");
    out.println("<form>");
  }
  else{
    out.println("You have not chosen to pay anything.<br><br>");

    out.println("<b>Note that if you fail to pay for a defined global requirement<br>");
    out.println("you will not be granted any permission even if you pay for <br>");
    out.println("access to an individual permission. Hit back in your browser<br> ");
    out.println("if you want to check this.</b><br><br>");

    //printing the rest of the form
    out.println("<input type='hidden' name='payment_ok' value='yes'>");
    out.println("<input type='submit' value='Generate Agreement'>");
    out.println("<form>");
  }
 }
 else{
  //getting the parameterNames of the request. Only the names are important
  Enumeration parameterNames = request.getParameterNames();

  //filtering out the parameters that are not interresting and collection the
  //rest in a Vector. This will be parameters in the agrement generation process
  String parameterName;
  Vector filteredParameterNames = new Vector();
  while (parameterNames.hasMoreElements()){
    parameterName = (String)parameterNames.nextElement();
```

```
      //ignoring dummy and payment_ok
      if (!(parameterName.equals("payment_ok")||parameterName.equals("dummy"))){
        filteredParameterNames.add(parameterName);
      }
    }

    //sending request to make agreement to the controller
    DRMController controller = new DRMController();
    boolean agreementMade = false;
    try{
      controller.makeAgreement(username,
                               collectionId,
                               itemId,
                               filteredParameterNames);
      agreementMade = true;
    }
    catch (Exception e){
      out.println("Sorry. Something went wrong. No agreement generated... <br><br>");
      e.printStackTrace(out);
    }

    //Agreement is now made and the user can continue on the get_Document servlet
    if(agreementMade){
      String filePath = request.getContextPath();
      filePath = filePath + "/get_document?CollectionId="+collectionId;
      filePath = filePath +"&ItemId="+itemId;
      URL url = new URL("http", request.getServerName(),8080, filePath);
      //sending the user to the created url
      //out.println(url.toString());
      response.sendRedirect(url.toString());
    }
  }
  out.println("</body>");
  out.println("</html>");
  out.close();
 }

/**************************************************************************/

 /**
  * <p>Description: Process the HTTP Get request<br>
  * presents the offer for the material allowing the user to select wich
  * obligations he wants to fulfill</p>
  *
  * @param request
  * @param response
  * @throws ServletException
  * @throws IOException
  */
 public void doGet(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {

   //prepairing to write to browser
   response.setContentType(CONTENT_TYPE);
   PrintWriter out = response.getWriter();

   //retrieves (or creates if doesn't exist) the users session
   HttpSession session = request.getSession(true);

   out.println("<html>");
   out.println("<head><title>makeAgreement</title></head>");
   out.println("<body>");

   String username = (String) session.getAttribute("username");
   String collectionId = (String)session.getAttribute("CollectionId");
   String itemId = (String)session.getAttribute("ItemId");

   out.println("<p align='right'>You are logged in as: ");
   out.println(username+"</p><br>");
   out.println(" <center>");
```

```
out.println("<h1>Welcome to REAP</h1>");
out.println("You have requested a document identified by: <br>");
out.println("CollectionId: "+collectionId+" <br>");
out.println("ItemId: "+itemId+"<br><br>");

//This method will be called twice
//Everything above will be identical to both calls.

boolean showOffer = (request.getParameter("statusGet")!=null);

//if this is a request made from this servlet this is executed
if (showOffer){
 //presents the offer as a form
 //This is the first time the doPostMethod is called (from doGet())
 out.println("<br><br>");
 out.println("This is the offer from the publisher for how you can");
 out.println("access the material identified by CollectionId ");
 out.println(collectionId+"and ItemId "+itemId+"<br><br>");

 out.println("If you want to make use of the offer you have to make ");
 out.println("an agreement by chosing which requirements you are willing ");
 out.println("to fullfill. Together with the information in your profile ");
 out.println("and the nature of your Internet connection, this will ");
 out.println("govern how you can use this material.<br><br>");

 out.println("</center>");

 DRMController controller = new DRMController();
 Document offer = null;

 try{

  //getting the offer
  offer = controller.getOffer(collectionId, itemId);

  //variables needed for traversal
  NodeList nodeList = null;
  Node offerNode = null;
  String nodeName ="";
  String temp ="";
  int i,j;

  //getting the root element as Node
  Element documentRoot = offer.getDocumentElement();

  //traversing the various parts of the offer
  nodeList = documentRoot.getChildNodes();
  for (i=0; i<nodeList.getLength();i++){
   nodeName = nodeList.item(i).getNodeName();

   if (nodeName.equals("offer")){
    //offerNode found. Traversing the elements of the offernode
    offerNode = nodeList.item(i);
    nodeList = offerNode.getChildNodes();
    for(j=0;j<nodeList.getLength();j++){
     nodeName = nodeList.item(j).getNodeName();
     if (nodeName.equals("context")){
      //contextNode found. Makes presentation
      temp = this.presentContext(nodeList.item(j))+"<br>";
      out.println(temp);
     }
     if (nodeName.equals("asset")){
      //asset node found. Makes presentation
      temp = this.presentAsset(nodeList.item(j))+"<br>";
      out.println(temp);
     }
     if (nodeName.equals("permission")){
      //permissions Node found. Makes presentation
      temp = this.presentPermissions(nodeList.item(j))+"<br>";
      out.println(temp);
```

```
            }
          }
          break; //breaks out of first loop
        }//if (nodeName.equals("offer")){
      }//for (i=0; i<nodeList.getLength();i++){
    }
    catch (Exception e){
      out.println("unexpected error encountered.Sorry..<br> ");
      e.printStackTrace(out);
    }
  }
  //this is executed if the request comes from the getDocument servlet.
  else{
    out.println("You don't have an agreement to access this material.<br>");
    out.println("Without such an agreement you can't access the material.<br><br>");

    out.println("An agreement governs how you can use the material you are");
    out.println("requesting, eg. if you for instance can view or print the ");
    out.println("material, and the conditions for getting these rights.<br>");
    out.println("Such an agreement is based on the initial rightsoffer given ");
    out.println("by the owner of the material and the information collected ");
    out.println("about you by REAP<br><br>");

    out.println("Would you like to negotiate with REAP about an agreement");
    out.println("that will give you access to the material?<br><br>");

    String filePath = request.getContextPath() + "/makeagreement";
    URL url = new URL("http", request.getServerName(),8080, filePath);
    out.println("<form action=\""+url.toString()+"\">");
    //this hidden parameter will cause the else clause to be executed when
    //the servlet is called again by pushing OK in the form
    out.println("<input type='hidden' name='statusGet' value='presentOffer'>");
    out.println("<input type=\"submit\" value=\"Yes\"");
    out.println("</form>");

    out.println("</center>");
    out.println("</body></html>");
  }
 }

/*****************************************************************************/

}
```

reap.servlets.register.java

```
package reap.servlets;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import reap.util.*;

/**
 * <p>Description: This servlet is used for registering new users</p>
 *
 * <p>Copyright: Copyright (c) 2002 Øyvind Vestavik</p>
 *
 * @author Øyvind Vestavik
 * @version 1.0
 */

public class register extends HttpServlet {

  /**
```

```
  * <p>Description: The default mimetype of this servlet</p>
  */
 private static final String CONTENT_TYPE = "text/html";

/*****************************************************************************/

 /**
  * <p>Description: Just a utility to write some html that can be called from
  * different places in this servlet</p>
  *
  * @param out A referense to the printwriter
  */
 private void printHead(PrintWriter out){
  out.println("<HEAD>");
  out.println("<TITLE> REAP Registration </TITLE>");
  out.println("<META NAME='Generator' CONTENT='register_servlet'>");
  out.println("<META NAME='Author' CONTENT='Øyvind Vestavik'>");
  out.println("<META NAME='Description'");
  out.println("CONTENT='This is the page used when registring new users'>");
  out.println("</HEAD>");
 }

/*****************************************************************************/


 /**
  * <p>Description: This method prints a registration form to the Printwriter
  * given as parameter to the method. The method can be called from doGet()
  * which always provide empty Strings as parameters to let the user fill in
  * all fields themselves, and from doPost() in response to a user not filling
  * in all required fields. In order not to force the user to fill in all fields
  * again, the values already filled in is passed to this method and the form is
  * printed to the user containing these values<p>
  *
  * @param out A referense to the printwriter
  * @param email The email adress the user already filled out
  * @param fname The first name the user already filled out
  * @param lname The last name the user already filled out
  */
 private void printForm(PrintWriter out,
                        String email,
                        String fname,
                        String lname)
 {
  out.println(" <FORM method=\"Post\" action=\"register\">");
  //One entry in the form is placed in one tablerow for easy alignment
  out.println("<center>");
  out.println("<table>");

  //email registration
  out.println(" <tr>");
  out.println("   <td align=\"right\">");
  out.println("    E-mail<font color=\"red\">*</font>: ");
  out.println("   </td>");
  out.println("   <td align=\"left\">");
  out.println("    <input type='text' name='email' value='"+ email+"'>");
  out.println("   </td>");
  out.println(" <tr>");

  //first name registraton
  out.println(" <tr>");
  out.println("   <td align=\"right\">");
  out.println("    First Name<font color=\"red\">*</font>: ");
  out.println("   </td>");
  out.println("   <td align=\"left\">");
  out.println("    <input type='text' name='fname' value='"+ fname+"'>");
  out.println("   </td>");
  out.println(" <tr>");
  out.println("</table>");
```

```
   //last name registraton
   out.println(" <tr>");
   out.println("   <td align=\"right\">");
   out.println("     Last Name<font color=\"red\">*</font>: ");
   out.println("   </td>");
   out.println("   <td align=\"left\">");
   out.println("     <input type='text' name='lname' value='"+ lname+"'>");
   out.println("   </td>");
   out.println(" <tr>");
   out.println("</table>");

   //Form Buttons
   out.println("<br><br>");
   out.println(" <input type=\"reset\" value=\"reset\">");
   out.println("<input type=\"submit\" value=\"register\">");
   out.println(" </FORM>");
   out.println("</center>");
  }

/*****************************************************************************/

  /**
   * <p>Description: This method responds to http get requests and just prints
   * a registration page with a registration form</P>
   *
   * @param request
   * @param response
   * @throws ServletException
   * @throws IOException
   */
  public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
   // preparing to write back to browser
   response.setContentType(CONTENT_TYPE);
   PrintWriter out = response.getWriter();

   //writing the registration form
   out.println("<HTML>");
   printHead(out); //prints the head portion of the document
   out.println("<body>");
   out.println(" <center>");
   out.println(" <h1>REAP registration of new users</h1>");

   out.println("By filling out the form below you register by REAP. <br>");
   out.println("This is required to get access to any documents protected");
   out.println("by REAP.<br><BR>");
   out.println(" <font color='red'>");
   out.println("Note: The more information you enter about yourself,");
   out.println("the more <BR>likely it is that you will get access to the");
   out.println("material you are requesting.</font><br><br>");
   out.println("All fields marked with <font color=\"red\">*</font>");
   out.println("are required<br>");
   out.println(" </center>");
   printForm(out, "","","");//calling method to print registration form
   out.println("</body>");
   out.println("</HTML>");

   out.close();
  }

/*****************************************************************************/

  /**
   * <p>Description: This method responds to http post requests  generated as
   * the user clicks submit on a form generated by the doGet-method. It first
   * forces the user to fill in all mandatory fields and then creates and stores
   * a profile for the user. The user is then presented with a username and
   * password to log in with. <br><br>There is a flaw in the functionality here. If a
   * user registers with a username (email adress) that is already registered
   * the old profile stored under this username will simply be written over.
   * It is not a priority task to fix this in this prototype </P>
```

```
 *
 * @param request object containing the users request
 * @param response object used for writing back to the user
 * @throws ServletException
 * @throws IOException
 */
public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
  //Setting up html writer
  response.setContentType(CONTENT_TYPE);
  PrintWriter out = response.getWriter();

  //indicates whether all mandatory fields are filled in
  boolean formComplete = true;

  //getting values entered by user in form
  String email= request.getParameter("email");
  String fname= request.getParameter("fname");
  String lname= request.getParameter("lname");

  //Checking for not filled out mandatory elements in form
  if (email.equals("")){formComplete=false;}
  if (fname.equals("")){formComplete=false;}
  if (lname.equals("")){formComplete=false;}

  //If mandatory elements not filled out print new page with form
  //values already entered by the user will be preserved
  if (!formComplete){
    out.println("<HTML>");
    printHead(out); //prints the head portion of the document
    out.println("<body>");
    out.println("<center>");
    out.println("<h1>REAP registration of new users</h1>");
    out.println("<font color=\"red\">");
    out.println("You have not filled out all mandatory fields<br>");
    out.println("Fill out all mandatory fields and submit again<br><br>");
    out.println("All fields marked with a red * are mandatory");
    out.println("</font>");
    //calling method to print form. Values already entered will be kept
    printForm(out, email, fname, lname);
    //finnishing the html print
    out.println("</body>");
    out.println("</HTML>");
  }
  else{ //form is completed
    //send values entered by user to java class for making of profile.
    //If everything executed ok, an object (up) is returned with useful info.
    ProfileHandler profilehandler = new ProfileHandler();
    UsernamePwdHolder up = null;
    up = profilehandler.make_profile(email, fname, lname);
    //collecting the response from the profilehandler
    String message = up.getMessage(); //retrieving status from testclass
    String password = up.getPassword();
    //printing back to user
    if (message.equals("profile_stored")){
      out.println("<HTML>");
      printHead(out); //prints the head portion of the document
      out.println("<body>");
      out.println(" <center>");
      out.println(" <h1>REAP registration of new users</h1>");
      out.println("You have sucessfully registered with REAP<br>");
      out.println("Below is your login information<br><br>");
      out.println("username : " + up.getUsername() +"<br>");
      out.println("password : " + up.getPassword() +"<br><r>");
      out.println("Hit back in your browser to return to the login page to <br>");
      out.println("enter your previously received password");
      out.println("</center>");
      out.println("</body>");
      out.println("</HTML>");
    }
```

```
    else{
      out.println("<HTML>");
      out.println("<HEAD>");
      out.println("<TITLE> REAP Registration </TITLE>");
      out.println("<META NAME=\"Generator\" CONTENT=\"register_servlet\">");
      out.println("<META NAME=\"Author\" CONTENT=\"Øyvind Vestavik\">");
      out.println("<META NAME=\"Description\" CONTENT=\"This is the page used when registring new users \">");
      out.println("</HEAD>");
      out.println("<body>");
      out.println(" <center>");
      out.println(" <h1>REAP registration of new users</h1>");
      out.println("<font color=\"red\">");
      out.println("Because of en error on the server, we are not able to");
      out.println("serve your request at this time<br>.");
      out.println("Most likely the system couldn't access the database<br>");
      out.println("Please try registering again later");
      out.println("</font>");
      out.println(" <center>");
      out.println("</body>");
      out.println("</HTML>");
    }
   }
  }

/*****************************************************************************/
}
```

reap.servlets.renderingApplication.java:

```
package reap.servlets;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import reap.DRMController.DRMController;

/**
 * <p>Description: This servlet functions as a substitute for a
 * renderingApplication and lets the user see if they can be granted a a ticket
 * for the permission they want to exercise according to the agreement they
 * have for access to this material</p>
 *
 * <p>Copyright: Copyright (c) 2002 Øyvind Vestavik</p>
 *
 * @author Øyvind Vestavik
 * @version 1.0
 */


public class renderingApplication extends HttpServlet {
  private static final String CONTENT_TYPE = "text/html";

/*****************************************************************************/

  /**
   * <p>Decription : Utility method that prints the result of the request for
   * a ticket. Prints whether the request was granted or not with an
   * indication of why if the ticket was not granted </p>
   *
   * @param out referance to the PrintWriter
   * @param ticket a Vector representing the ticket. The ticket contains
   * information on wether the ticket has been granted or not, and if not, the
   * reason why it could not be granted.
   *
   */
  private void presentTicket(PrintWriter out,
```

```
                        Vector ticket){

   boolean ticketGranted = false;
   String vectorElement;
   int i;
   String key = "";
   String value = "";

   //evaluating the vector ticket to ses if the ticket has been granted
   Iterator StatusIterator = ticket.iterator();
   while (StatusIterator.hasNext()){
     //one of the elements should be a key value string as follows
     //"ticketGranted=yes|no"
     vectorElement = (String)StatusIterator.next();
     StringTokenizer statusTokenizer =
         new StringTokenizer(vectorElement, "=");
     i = 1;
     while (statusTokenizer.hasMoreTokens()) {
       if (i==1){
         key = statusTokenizer.nextToken();
       }
       else{
         value = statusTokenizer.nextToken() ;
       }
       i++;
     }
     if (key.equals("ticketGranted")){
       if (value.equals("yes")){
         ticketGranted = true;
       }
       break;
     }
   }//evaluation of granted

   if (ticketGranted){
     out.println("You have been granted one execution of this permission <br>");
     out.println("and your agreement has been updated accordingly <br>");
   }
   else{
     out.println("This permission could not be granted <br>");
     out.println("here's why: <br>");
     //traverses the vector and collects the strings giving reasons
     Iterator errorIterator = ticket.iterator();
     while (errorIterator.hasNext()){
       vectorElement = (String)errorIterator.next();
       StringTokenizer statusTokenizer =
         new StringTokenizer(vectorElement, "=");
       i = 1;
       while (statusTokenizer.hasMoreTokens()) {
         if (i==1){
           key = statusTokenizer.nextToken();
         }
         else{
           value = statusTokenizer.nextToken() ;
         }
         i++;
       }//while
       if (key.equals("error")){
         out.println(value+"<br>");
       }
     }//while
   }//if else
 }

/*****************************************************************************/

  /**
   * <p>Description: Prosesses all http get requests from the browser<br>
   * Showing an message first the first time it is called, then presenting the
   * substitute renderingApplication on subsequent calls</p>
```

```
 *
 * @param request
 * @param response
 * @throws ServletException
 * @throws IOException
 */
public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
  response.setContentType(CONTENT_TYPE);
  PrintWriter out = response.getWriter();

  //getting the session.
  HttpSession session = request.getSession(true);
  String username = (String) session.getAttribute("username");
  String collectionId = (String)session.getAttribute("CollectionId");
  String itemId = (String)session.getAttribute("ItemId");

  out.println("<html>");
  out.println("<head><title>renderingApplication</title></head>");
  out.println("<body>");

  out.println("<p align='right'>You are logged in as: ");
  out.println(username+"</p><br>");
  out.println(" <center>");

  //If this is the first request to this servlets doGet a text is displayed
  //explaining what this servlet does
  //Subsequent calls deals with the actual use of the servlet
  boolean showStartMessage = (request.getParameter("startMessageShown")==null);
  if (showStartMessage){
     //shows opening warning
     out.println("<h1>Welcome to REAPs Rendering Application</h1>");
     out.println("This servlet is a substitute for a real html rendering application<br>");
     out.println("It is not ment as an actual presentation tool, but as a <br>");
     out.println("mere illustration of how REAP evaluates the rights <br>");
     out.println("over an asset that can be granted to a user given a <br>");
     out.println("certain connection to the Internet and a given agreement <br><br>");
     out.println("It also implements a communication protocol with the rest of <br>");
     out.println("system to illustrate how a real rendering application could <br>");
     out.println("determine which permissions a user could be granted on a per <br>");
     out.println("request basis.<br><br>");
     out.println("To start using this demonstration, press the button below<br><br>");
     //printing a button
     out.println("<form>");
     out.println("<input type='hidden' name='startMessageShown' value='yes'>");
     out.println("<input type='submit' value='Start'>");
     out.println("</form>");
  }
  else{
    //getting the material from the DRMController
        //Material is never presented in this prototype

    out.println("<h1>REAPs Rendering Application</h1>");
    out.println("By pressing one of the buttons below, you request an <br>");
    out.println("execution of the corresponding permission. This <br>");
    out.println("request will be evaluated against your agreement, taking <br>");
    out.println("into account the information about you in your profile and <br>");
    out.println("the caracteristics of your current connection to the Internet <br>");
    out.println("<br>");
    out.println("If the requested permission can be granted, a ticket for <br>");
    out.println("the permission is given, and your agreement is updated correspondingly. <br>");
    out.println("Else an error message is given, indicating why the permission <br>");
    out.println("can't be granted you <br><br>");


    //prints out four submit buttons representing the four permissions that
    //is supported by REAP
    out.println("<form>");
    out.println("<input type='hidden' name='startMessageShown' value='yes'>");
    out.println("<input type='submit' name='display' value='display'>");
    out.println("<input type='submit' name='print' value='print'>");
```

```
out.println("<input type='submit' name='play' value='play'>");
out.println("<input type='submit' name='execute' value='execute'>");
out.println("</form>");
out.println("<br>");

//boolean values indicating which button was used to generating the call
//if any. Maximum one of the attributes will have value true.
boolean displayRequested = (request.getParameter("display")!=null);
boolean printRequested = (request.getParameter("print")!=null);
boolean playRequested = (request.getParameter("play")!=null);
boolean executeRequested = (request.getParameter("execute")!=null);

//getting caracteristics of request
String IPAdress = request.getRemoteAddr();

//setting some attributes
Vector ticket = new Vector();
DRMController controller = new DRMController();

//generating corresponing action
if (displayRequested){
  out.println("display requested <br>");
  String permission = "display";
  try{
    ticket = controller.getTicket(collectionId,
                                  itemId,
                                  permission,
                                  username,
                                  IPAdress);
    this.presentTicket(out, ticket);
    out.println("<br>");
  }
  catch(Exception e){
    out.println(e.getMessage());
    out.println("<br>");
    e.printStackTrace(out);
    out.println("Because of an internal error the system can't evaluate <br>");
    out.println("your request at this time. No ticket is given and you <br>");
    out.println("are not given permission to execute this permission now. <br>");
  }
}

if (printRequested){
  out.println("print requested <br>");
  String permission = "print";
  try{
    ticket = controller.getTicket(collectionId,
                                  itemId,
                                  permission,
                                  username,
                                  IPAdress);
    this.presentTicket(out, ticket);
    out.println("<br>");
  }
  catch(Exception e){
    out.println("Because of an internal error the system can't evaluate <br>");
    out.println("your request at this time. No ticket is given and you <br>");
    out.println("are not given permission to execute this permission now. <br>");
  }
}
if (playRequested){
  out.println("play requested <br>");
  String permission = "play";
  try{
    ticket = controller.getTicket(collectionId,
                                  itemId,
                                  permission,
                                  username,
                                  IPAdress);
    this.presentTicket(out, ticket);
```

```
        out.println("<br>");
       }
       catch(Exception e){
        out.println("Because of an internal error the system can't evaluate <br>");
        out.println("your request at this time. No ticket is given and you <br>");
        out.println("are not given permission to execute this permission now. <br>");
       }
      }
      if (executeRequested){
       out.println("execute requested <br>");
       String permission = "execute";
       try{
        ticket = controller.getTicket(collectionId,
                                      itemId,
                                      permission,
                                      username,
                                      IPAdress);
        this.presentTicket(out, ticket);
        out.println("<br>");
       }
       catch(Exception e){
        out.println("Because of an internal error the system can't evaluate <br>");
        out.println("your request at this time. No ticket is given and you <br>");
        out.println("are not given permission to execute this permission now. <br>");
       }
      }
    }
    out.println("</center>");
    out.println("</body></html>");
  }
}
```

reap.DRMController.DRMController.java

```
package reap.DRMController;

import reap.DRMPackager.DRMPackager;
import reap.util.DRMPackage;
import reap.DRMLicenceGenerator.DRMLicenceGenerator;
import org.w3c.dom.*;
import java.util.*;

//nessecery because the servlet makeAgreement sends the whole request as a parameter to
//the class to be evaluated. A lot of overhead, but saves me from much programming
import javax.servlet.http.HttpServletRequest;

/**
 * <p>Title: REAP</p>
 * <p>Description: a Rights Enforcing Access Protocol<br>
 * All parts of the userinterface interacts with the rest of the system through
 * this class</p>
 * <p>Copyright: Copyright (c) 2002</p>
 * @author Øyvind Vestavik
 * @version 1.0
 */

public class DRMController{



/******************************************************************************/
  /**
   * <p>Description: Just the constructor</p>
   */
  public DRMController(){
  }

/******************************************************************************/
```

```
 /**
  * <p>Description: Just forwards the request from the frontend
  * to get the materials offer to the DRMLicencegenerator</p>
  *
  * @param collectionId a positive integer identifing the collection the item is in
  * @param itemId a positive integer identifing a resource witin a collection
  * @return a DOM Document containing the initial offer made by the rightsholder(s)
  * describing which permissions are offered for this resource
  * @throws Exception Any exceptions thrown in this method or in methods called by
  * this method is thrown back to the calling class to be handled there.
  */
 public Document getOffer(String collectionId, String itemId)throws Exception{
   Document offer = null;
   //get the offer from database via DRM LicenceGenerator if it exists
   //if it doesn't exist an Exception is thrown
   DRMLicenceGenerator generator = new DRMLicenceGenerator();
   try{
     offer = generator.getOffer(collectionId, itemId);
   }
   catch(Exception e){
     throw e;
   }
   return offer;
 }

/*****************************************************************************/

 /**
  * <p>Description: A ticket is the permission to exercise a permission once. Users with an
  * agreement can request a ticket for a permission on a resource calling this
  * method. The method will try to obtain a ticket on behalf of the user from
  * the DRMLicenceGenerator</p>
  *
  * @param collectionId a positive integer identifing the collection the item is in
  * @param itemId a positive integer identifing a resource witin a collection
  * @param permission the permission the user wants to access
  * @param username the registered username of the user
  * @param IPAdress the IPAdress of the users current connection
  * @return A vector containing strings on the form of key-value pairs telling if the
  * ticket could be granted or not and why not.
  * @throws Exception Any exceptions thrown in this method or in methods called by
  * this method is thrown back to the calling class to be handled there.
  */
 public Vector getTicket(String collectionId,
                         String itemId,
                         String permission,
                         String username,
                         String IPAdress)
                         throws Exception{

   //the Vector to be returned
   Vector ticket = new Vector();
   try{
     //forwarding request to DRMLicenceGenerator
     DRMLicenceGenerator licenceGenerator = new DRMLicenceGenerator();
     ticket = licenceGenerator.generateTicket(collectionId,
                                              itemId,
                                              permission,
                                              username,
                                              IPAdress);
   }
   catch(Exception e){
     throw e;
   }
   return ticket;
 }

/*****************************************************************************/

 /**
```

```
  *<p>Description: based on collectionId and itemId checks to see if the material
  *can be retrieved/exists. Username shouldn't bee needed here but thats how it
  *works here, because this method makes a DRMPackage and checks to see if it
  *contains material</p>
  *
  * @param collectionId a positive integer identifing the collection the item is in
  * @param itemId a positive integer identifing a resource witin a collection
  * @param username the registered username of the user
  * @return boolean value (true||false) to indicate if the material exists
  */
 public boolean materialExists (String collectionId,
                          String itemId,
                          String username){
  boolean exists = false;
  //Getting the resource
  DRMPackager drmPackager = new DRMPackager();
  DRMPackage myPackage = drmPackager.getPackage(collectionId,
                             itemId,
                             username);
  if (myPackage.containsMaterial)
    exists = true;
  //freeing resource
  myPackage = null;
  return exists;
 }

/*****************************************************************************/

  /**
  *<p>Description: based on collectionId, itemId and username checks to see if
  *the user has an agreement for this material. Request is just forwarded to
  *DRMLicenceGenerator</p>
  *
  * @param collectionId a positive integer identifing the collection the item is in
  * @param itemId a positive integer identifing a resource witin a collection
  * @param username the registered username of the user
  * @return boolean value (true||false) to indicate if the user has an agreement
  * for this material.
  * @throws Exception Any exceptions thrown in this method or in methods called by
  * this method is thrown back to the calling class to be handled there.
  */
 public boolean checkForAgreement(String collectionId,
                           String itemId,
                           String username)
                           throws Exception{
  //Just forvarding the call to the LicenceGenerator
  boolean agreementExists = false;
  DRMLicenceGenerator licenceGenerator = new DRMLicenceGenerator();
  try{
    agreementExists =
        licenceGenerator.checkForAgreement(collectionId, itemId, username);
  }
  catch(Exception e){
    throw e;
  }
  return agreementExists;
 }

/*****************************************************************************/

  /**
  *<p>Description: Receives a request to make an agreement over an material.
  *Forwards this request to the DRMLicenceGenerator. If the agreement could not
  *be created, an exeption is thrown</p>
  *
  * @param username the registered username of the user
  * @param collectionId a positive integer identifing the collection the item is in
  * @param itemId a positive integer identifing a resource witin a collection
  * @param fullfilledReq Vector of formatted strings indicating what requirements
  * the user has fullfilled.
```

```
   * @throws Exception Any exceptions thrown in this method or in methods called by
   * this method is thrown back to the calling class to be handled there.
   */
  public void makeAgreement(String username,
                            String collectionId,
                            String itemId,
                            Vector fullfilledReq)
                            throws Exception{
    //forwarding to DRMLicenceGenerator
    DRMLicenceGenerator generator = new DRMLicenceGenerator();
    try{
      generator.generateAgreement(username, collectionId, itemId, fullfilledReq);
    }
    catch (Exception e){
      throw e;
    }
  }

/*****************************************************************************/

/*
  public DRMPackage getPackage(String collectionId, String itemId){
    //kontakt DRMPackager for å få tak i materialet
    //tester
    System.out.println("");
    System.out.println("DRMController har mottatt DRMPackage");
    //System.out.println("  Inneholder Offer: "+drmPackage.containsOffer);
    System.out.println("  Inneholder materiale: "+drmPackage.containsMaterial);
    System.out.println("  filnavn: "+drmPackage.material.fileName);
    System.out.println("  filstørrelse i bytes: " + drmPackage.material.fileSize);
    //leser fila uten å gjøre noe med den
    int sink;
    int nrBytes = 0;
    for (int i=0 ; i<drmPackage.material.content.length ; i++){
      //System.out.print(drmPackage.material.content[i]+" ");
      sink = drmPackage.material.content[i];
      nrBytes++;
    }
    System.out.println("Antall bytes lest: " + nrBytes);
    System.out.println("Hele fila er mottatt");
  }
*/

/*****************************************************************************/

}
```

reap.DRMLicenceGenerator.DRMLicencegenerator.java

```
package reap.DRMLicenceGenerator;

import reap.util.database.xml.xmlDatabaseFrontend;
import org.w3c.dom.*;
import org.apache.xerces.dom.DOMImplementationImpl;
import java.util.*;
import java.io.*;

//receives the whole servlet request with parameters so that it can be
//evaluated
import javax.servlet.http.HttpServletRequest;

/**
 * <p>Title: REAP</p>
 * <p>Description: a Rights Enforcing Access Protocol</p>
 * <p>Copyright: Copyright (c) 2002</p>
 * <p>Company: </p>
 * @author Øyvind Vestavik
 * @version 1.0
```

```
 */

public class DRMLicenceGenerator {

  //Attribute used for temporary storage between calls by
  //private boolean evaluateConstraintsto save a global
  //constraint count value to be evaluated in a later call
  private int globalConstraintCountValue;

/*****************************************************************************/

  /**
   * <p>Description: Default constructor</p>
   */
  public DRMLicenceGenerator() {
  }

/*****************************************************************************/

  /**
   * <p>Description: This method contacts the database frontend for the database
   * containing the offers, agreements and userprofiles and retrieves the offer
   * for the material</p>
   *
   * @param collectionId positive integer telling which collection the resource is in
   * @param itemId positive integer identifying the resource within the collection
   * @return a DOM Document containing the offer from the rightsholder(s)
   * @throws Exception Any exceptions thrown by this method og methods called by
   * it are thrown back to the caller to be handled there
   */
  public Document getOffer(String collectionId, String itemId)throws Exception{
    //getting the offer from the database. If the offer doesn't exist an
    //exception is thrown
    xmlDatabaseFrontend xmldb = new xmlDatabaseFrontend();
    Document offer = null;
    try{
      offer = xmldb.get_Offer(collectionId, itemId);
    }
    catch(Exception e){
      throw e;
    }
    return offer;
  }

/*****************************************************************************/

  /**
   * <p>Description: This method contacts the database frontend for the database
   * containing the offers, agreements and userprofiles and checks if the user
   * has an agreement for this material</p>
   *
   * @param collectionId positive integer telling which collection the resource is in
   * @param itemId positive integer identifying the resource within the collection
   * @param username The registered username for the user
   * @return boolean value indicating if the user has an agreement for the resource
   * @throws Exception Any exceptions thrown by this method og methods called by
   * it are thrown back to the caller to be handled there
   */
  public boolean checkForAgreement(String collectionId,
                                   String itemId,
                                   String username)
                                   throws Exception{

    //checks to see if agrement exists
    boolean agreementExists = false;
    //xmlDatabaseFrontend xmldb = new xmlDatabaseFrontend();
    xmlDatabaseFrontend xmldb = new xmlDatabaseFrontend();
    try{
      agreementExists = xmldb.agreement_exists(username, collectionId, itemId);
    }
```

```
   catch (Exception e){
     throw e;
   }
   return agreementExists;
  }

/*****************************************************************************/

  /**
   * <p>Description: just a helper method for the generateAgreement method.<br>
   * Receives the agreement Element and the DOM Document and appends a context to
   * it containing the generation date of the agreement, no return value needed</p>
   *
   * @param agreement the xml Element to be appended with context
   * @param Agreement the reference to the DOM Document the agreement Element is in
   */
  private void generateAgreementContext(Element agreement, Document Agreement){
    Element context = Agreement.createElement("context");
       Element contextDate = Agreement.createElement("date");
       //creating a date of the agreement(local time) to be inserted
       Calendar rightNow = Calendar.getInstance();
       int yearField = rightNow.YEAR;
       int year = rightNow.get(yearField);
       //getting the month. Something funny here
       int monthField = rightNow.MONTH;
       Integer integerMonth = new Integer(rightNow.get(monthField)+1);
       String month = integerMonth.toString();
       if (month.length()==1)
         month= "0"+month;
       //getting day of month (date)
       int dayOfMonthField = rightNow.DAY_OF_MONTH;
       Integer integerDay = new Integer(rightNow.get(dayOfMonthField));
       String dayOfMonth = integerDay.toString();
       if (dayOfMonth.length()==1)
         dayOfMonth= "0"+dayOfMonth;
       //generating an ISO dateString
       String date = year+"-"+month+"-"+dayOfMonth;
       //creating a textnode to hold the generated string date
       Text contextDateText = Agreement.createTextNode(date);
       //binding the elements together
       contextDate.appendChild(contextDateText);
       context.appendChild(contextDate);
       agreement.appendChild(context);
  }

/*****************************************************************************/

  /**
   * <p>Description: just a helper method for the generateAgreement method.<br>
   * Adds the asset description of the offer to the agreement in making</p>
   *
   * @param agreement the xml Element to be appended with asset description
   * @param Agreement the reference to the DOM Document the agreement Element is in
   * @param collectionId the collectionId of the material the agreement is about
   * @param itemId the itemId of the material the agreement is about
   * @param offerAsset the asset description of the offer
   */
  private void generateAgreementAsset(Element agreement,
                                      Document Agreement,
                                      String collectionId,
                                      String itemId,
                                      Node offerAsset){
    int i;
    //Making asset part
       Element asset = Agreement.createElement("asset");
       //making the reapId part of the agreement
       Element reapId = Agreement.createElement("reapId");
       Element collectionIdElement = Agreement.createElement("collectionId");
       Text collectionIdText = Agreement.createTextNode(collectionId);
       collectionIdElement.appendChild(collectionIdText);
```

```
      Element itemIdElement = Agreement.createElement("itemId");
      Text itemIdText = Agreement.createTextNode(itemId);
      itemIdElement.appendChild(itemIdText);
      reapId.appendChild(collectionIdElement);
      reapId.appendChild(itemIdElement);
      asset.appendChild(reapId);
      //making evt context part of asset if present in offer
      boolean hasContext = false;
      Node offerAssetContextNode = null;
      NodeList offerAssetNodeList = offerAsset.getChildNodes();
      for (i=0; i<offerAssetNodeList.getLength(); i++){
       if (offerAssetNodeList.item(i).getNodeName().equals("context")){
        hasContext = true;
        offerAssetContextNode = offerAssetNodeList.item(i);
        break;
       }
      }
      if (hasContext){
       Element assetContext = Agreement.createElement("context");
       NodeList offerAssetContextNodeList =
                    offerAssetContextNode.getChildNodes();
       for (i=0; i<offerAssetContextNodeList.getLength(); i++){
        String nodeName = offerAssetContextNodeList.item(i).getNodeName();
        if (nodeName.equals("uid")){
         //getting the offers node for uid
         Node offerAssetContextUidNode = offerAssetContextNodeList.item(i);
         //getting the attribute and its value
         NamedNodeMap attributes = offerAssetContextUidNode.getAttributes();
         Node idscheme = attributes.getNamedItem("idscheme");
         String attributeValue = idscheme.getLastChild().getNodeValue();
         //getting the childvalue for the offers uid
         String nodeValue =
              offerAssetContextUidNode.getFirstChild().getNodeValue();
         //creating the corresponding construct in the agreement
         Element uid = Agreement.createElement("uid");
         uid.setAttribute("idscheme", attributeValue);
         Text uidText = Agreement.createTextNode(nodeValue);
         uid.appendChild(uidText);
         assetContext.appendChild(uid);
        }
        if (nodeName.equals("name")){
         //getting the name node from the offer and its value
         Node offerNameNode = offerAssetContextNodeList.item(i);
         String offerNameValue = offerNameNode.getFirstChild().getNodeValue();
         //creating the corresponding construct in the agreement
         Element name = Agreement.createElement("name");
         Text nameText = Agreement.createTextNode(offerNameValue);
         name.appendChild(nameText);
         assetContext.appendChild(name);
        }
        if (nodeName.equals("MIMEType")){
         //getting the name node from the offer and its value
         Node offerMimetypeNode = offerAssetContextNodeList.item(i);
         String offerMimetypeValue =
                    offerMimetypeNode.getFirstChild().getNodeValue();
         //creating the corresponding construct in the agreement
         Element MIMEType = Agreement.createElement("MIMEType");
         Text MIMETypeText = Agreement.createTextNode(offerMimetypeValue);
         MIMEType.appendChild(MIMETypeText);
         assetContext.appendChild(MIMEType);
        }
       }
       asset.appendChild(assetContext);
      }
     agreement.appendChild(asset);
   }

/*****************************************************************************/

   /**
```

```
 * <p>Description: Just a helper method for the generateAgreement method.<br>
 * Receives a requirement from the offer and a list of
 * requirements that the user has fullfilled and checks to see if the
 * requirement is in the list of requirements the user has fullfilled.
 * If the requirement has been fullfilled the method returns true, else false.
 * The parent permission parameter is required to make the comparison.</p>
 *
 * @param reqFullfilled A vektor of formatted strings telling which
 * requirements the user has fullfilled
 * @param requirement the requirement from the offer to be evaluated
 * @param parentPermission the parent permission of the requirement to be
 * evaluated
 * @return boolean value indicating if the user has fullfilled this requirement
 */
private boolean checkRequirement(Vector reqFullfilled,
                                 Node requirement,
                                 String parentPermission){

  boolean satisfied = false;
  String nodeName;
  String compareString = null;

  //traversing to the bottom of the requirement node and makes a string which
  //can be compared to the strings in the vector
  NodeList requirementChildren = requirement.getChildNodes();
  for (int i=0; i<requirementChildren.getLength(); i++){
    nodeName = requirementChildren.item(i).getNodeName();
    if (nodeName.equals("prepay")){
      Node prepayNode = requirementChildren.item(i);
      NodeList prepayChildren = prepayNode.getChildNodes();
      for (int j=0; j<prepayChildren.getLength();j++){
        nodeName = prepayChildren.item(j).getNodeName();
        if (nodeName.equals("payment")){
          Node paymentNode = prepayChildren.item(j);
          NodeList paymentChildren = paymentNode.getChildNodes();
          for(int k=0; k<paymentChildren.getLength(); k++){
            nodeName = paymentChildren.item(k).getNodeName();
            if (nodeName.equals("amount")){
              Node amountNode = paymentChildren.item(k);
              //getting the value of amount
              String amount =amountNode.getFirstChild().getNodeValue();
              //getting the currency attribute and its value
              NamedNodeMap attributes = amountNode.getAttributes();
              Node currencyAttribute = attributes.getNamedItem("currency");
              String currencyType = currencyAttribute.getFirstChild().getNodeValue();
              //genereating the string to be used for comparison
              compareString ="req=prepay,currency="+currencyType;
              compareString = compareString + ",amount="+amount;
              compareString = compareString +",parent="+parentPermission;
              break;
            }
          }
        break;
        }
      }
      break;
    }
  }

  //compares the String generated to the strings in the vektor,
  //if found identical string, then the requirement is fullfilled and the method
  //will return true

  Iterator iterator = reqFullfilled.iterator();
  String reqString;
  while (iterator.hasNext()){
    reqString = (String)iterator.next();
    if (reqString.equals(compareString)){
      //removes so the String can't be used to satisfy another identical requirement
      iterator.remove();
```

```
      satisfied = true;
    }
   }
   return satisfied;
  }

/*****************************************************************************/


  /**
   * <p>Description: Just a helper method for the generateAgreement method.<br>
   * Since constraints are not evaluated in the process of making an agreement
   * any constraints in the offer are just copied to the agreement so that they
   * can be evaluated when the user requests a ticket based on his agreement</p>
   *
   * @param Agreement reference to the agreement in making
   * @param offerConstraintNode the constraint from the offer to be added to the
   * agreement
   * @param agreementConstraintElement The Element of the agreement that is to
   * be extended with a constraint
   * @param parentPermission The Element representing the permission to be given
   * a constraint in the agreement
   */
  private void copyConstraint(Document Agreement,
                              Node offerConstraintNode,
                              Element agreementConstraintElement,
                              String parentPermission){

    int i, j;

    //traverses the constraintnode of the offer
    NodeList offerConstraintChildren = offerConstraintNode.getChildNodes();
    for (i=0; i<offerConstraintChildren.getLength();i++){
      String offerConstraintChildName =
          offerConstraintChildren.item(i).getNodeName();

      if (offerConstraintChildName.equals("count")){
        //evaluating the count element of the offer
        Node offerCount = offerConstraintChildren.item(i);
        NodeList offerCountChildren = offerCount.getChildNodes();
        for (j=0; j<offerCountChildren.getLength(); j++){
          String offerCountChildName = offerCountChildren.item(j).getNodeName();
          if (offerCountChildName.equals("max")){
            String max =
              offerCountChildren.item(j).getFirstChild().getNodeValue();
            //constructing the agreements count constraint
            Element agreementCount = Agreement.createElement("count");
            Element agreementCountMax = Agreement.createElement("max");
            Text agreementCountMaxValue = Agreement.createTextNode(max);
            agreementCountMax.appendChild(agreementCountMaxValue);
            agreementCount.appendChild(agreementCountMax);
            agreementConstraintElement.appendChild(agreementCount);
            if (!(parentPermission.equals("global"))){
              //makes the executed element to be used as a counter
              Element executed = Agreement.createElement("executed");
              Text executedValue = Agreement.createTextNode("0");
              executed.appendChild(executedValue);
              agreementCount.appendChild(executed);
            }
          }
        }
      }

      if (offerConstraintChildName.equals("dateRange")){
        //Evaluating the dateRange constraint of the offer
        String start = null;
        String end = null;
        Node offerDateRange = offerConstraintChildren.item(i);
        NodeList offerDateRangeChildren = offerDateRange.getChildNodes();
        for (j=0; j<offerDateRangeChildren.getLength(); j++){
```

```
      String offerDateRangeChildName =
        offerDateRangeChildren.item(j).getNodeName();
      if (offerDateRangeChildName.equals("start")){
        start = offerDateRangeChildren.item(j).getFirstChild().getNodeValue();
      }
      if (offerDateRangeChildName.equals("end")){
        end = offerDateRangeChildren.item(j).getFirstChild().getNodeValue();
      }
    }
    //building the agreements construct for dateRange and adding to constraint
    Element agreementDateRange = Agreement.createElement("dateRange");
    Element agreementDateRangeStart = Agreement.createElement("start");
    Text agreementDateRangeStartValue = Agreement.createTextNode(start);
    agreementDateRangeStart.appendChild(agreementDateRangeStartValue);
    Element agreementDateRangeEnd = Agreement.createElement("end");
    Text agreementDateRangeEndValue = Agreement.createTextNode(end);
    agreementDateRangeEnd.appendChild(agreementDateRangeEndValue);
    agreementDateRange.appendChild(agreementDateRangeStart);
    agreementDateRange.appendChild(agreementDateRangeEnd);
    agreementConstraintElement.appendChild(agreementDateRange);
  }

  if (offerConstraintChildName.equals("individual")){
    Node offerIndividual = offerConstraintChildren.item(i);
    String individualValue = "";
    if (offerIndividual.hasChildNodes()){
      individualValue = offerIndividual.getFirstChild().getNodeValue();
    }
    //Building the agreements presentation of the individual constraint
    Element agreementIndividual = Agreement.createElement("individual");
    Text agreementIndividualValue = Agreement.createTextNode(individualValue);
    agreementIndividual.appendChild(agreementIndividualValue);
    agreementConstraintElement.appendChild(agreementIndividual);
  }

  if (offerConstraintChildName.equals("network")){
    //making networkElement in agreemnt and adding to constraint
    Element agreementNetwork = Agreement.createElement("network");
    agreementConstraintElement.appendChild(agreementNetwork);
    //Evaluating the network part of the offer
    Node offerNetwork = offerConstraintChildren.item(i);
    NodeList offerNetworkChildren = offerNetwork.getChildNodes();
    for (j=0; j<offerNetworkChildren.getLength(); j++){
      String offerNetworkChildName =
        offerNetworkChildren.item(j).getNodeName();
      if (offerNetworkChildName.equals("networkMask")){
        Node offerNetworkMask = offerNetworkChildren.item(j);
        String offerNetworkMaskValue = "";
        if (offerNetworkMask.hasChildNodes()){
          offerNetworkMaskValue =
            offerNetworkMask.getFirstChild().getNodeValue();
        }
        //building the IPMask construct into the agreement
        Element agreementIPMask = Agreement.createElement("networkMask");
        Text agreementIPMaskValue = Agreement.createTextNode(offerNetworkMaskValue);
        agreementIPMask.appendChild(agreementIPMaskValue);
        agreementNetwork.appendChild(agreementIPMask);
      }
      if (offerNetworkChildName.equals("startIPnr")){
        //getting the nodes for startIP and endIP in the offer
        Node offerStartIPnr = offerNetworkChildren.item(j);
        Node offerEndIPnr = offerStartIPnr.getNextSibling().getNextSibling();
        //getting the values for start and end IP
        String offerStartIPnrValue = "";
        String offerEndIPnrValue = "";
        if (offerStartIPnr.hasChildNodes()){
          offerStartIPnrValue = offerStartIPnr.getFirstChild().getNodeValue();
        }
        if (offerEndIPnr.hasChildNodes()){
          offerEndIPnrValue = offerEndIPnr.getFirstChild().getNodeValue();
```

```
          }
          //building the agreements presentation
          Element startIP = Agreement.createElement("startIPnr");
          Text startIPValue = Agreement.createTextNode(offerStartIPnrValue);
          startIP.appendChild(startIPValue);
          Element endIP = Agreement.createElement("endIPnr");
          Text endIPValue = Agreement.createTextNode(offerEndIPnrValue);
          endIP.appendChild(endIPValue);
          //binding to network element
          agreementNetwork.appendChild(startIP);
          agreementNetwork.appendChild(endIP);
        }
       }
     }
   }
 }

/*****************************************************************************/

 /**
  * <p>Description: Just a helper method for the generateAgreement method.<br>
  * Generates the permission part of the agreement based on the permission part
  * of the offer. Evaluates the global and local requirements and limits the
  * permissions to those for which the requirements (global and local) has been
  * satisfied. Global and local constraints are just copied into the agreement here.
  * They are evaluated when the user tries to access an permission in this agreement
  * </p>
  *
  * @param agreement The agreement Element of the agreement
  * @param Agreement Reference to the agreement in making
  * @param offerPermission The Element representing the permission part of the offer
  * @param fullfilledReq a vector containing the requirements
  * the user has fullfilled
  */
 private void generateAgreementPermissions(Element agreement,
                                 Document Agreement,
                                 Node offerPermission,
                                 Vector fullfilledReq){

   int i,j;
   String nodeName;
   Node requirement = null;
   Node constraint = null;
   boolean globalRequirementSatisfied = true;


   //makes permission Element and connects to agreement element
     Element permission = Agreement.createElement("permission");
     agreement.appendChild(permission);

   //getting the children of the offers permissionNode
     NodeList offerPermissionChildren = offerPermission.getChildNodes();

   //determines if any permissions can be given at all by checking
   //global requirement in the offer against the requirements the user has
   //fullfilled
     for (i=0; i<offerPermissionChildren.getLength(); i++){
       nodeName = offerPermissionChildren.item(i).getNodeName();
       if (nodeName.equals("requirement")){
         requirement = offerPermissionChildren.item(i);
         globalRequirementSatisfied =
           this.checkRequirement(fullfilledReq, requirement, "global");
       }
     }

   //if global requirements are satisfied, constructs the offers global
   //constraints in the agreement (will be evaluated at access time)
     if (globalRequirementSatisfied){
       for (i=0; i<offerPermissionChildren.getLength(); i++){
         nodeName = offerPermissionChildren.item(i).getNodeName();
```

```
      if (nodeName.equals("constraint")){
        //makes agreements constraintNode and adds to agreementNode
        Element agreementGlobalConstraint =
            Agreement.createElement("constraint");
        permission.appendChild(agreementGlobalConstraint);
        Node offerGlobalConstraint = offerPermissionChildren.item(i);
        //calls method to make make the constraint construct to be added to
        //the agreement
        this.copyConstraint(Agreement,
                            offerGlobalConstraint,
                            agreementGlobalConstraint,
                            "global");
      }
     }
    }

  //if global requirements are satisfied, can evaluate individual permissons
  if (globalRequirementSatisfied){
     for (i=0; i<offerPermissionChildren.getLength(); i++){
      nodeName = offerPermissionChildren.item(i).getNodeName();

      if (nodeName.equals("display")) {
        Node offerDisplay = offerPermissionChildren.item(i);
        NodeList offerDisplayChildren = offerDisplay.getChildNodes() ;
        //checking to see if requirements for this permission are satisfied
        //if there are no requirement to be evaluated, requirements are seen
        //as satisfied
        boolean displayReqSatisfied = true;
        for (j=0; j<offerDisplayChildren.getLength(); j++){
          if (offerDisplayChildren.item(j).getNodeName().equals("requirement")){
            Node displayReq = offerDisplayChildren.item(j);
            displayReqSatisfied =
              this.checkRequirement(fullfilledReq, displayReq, "display");
          }
        }
        if (displayReqSatisfied){
          //Makes the Display Element an attaches to agreement
          Element display = Agreement.createElement("display");
          permission.appendChild(display);

          //getting the constraint for this permission if exist
          Node offerDisplayConstraint = null;
          Element agreementDisplayConstraint = null;
          for (j=0; j<offerDisplayChildren.getLength(); j++){
            if (offerDisplayChildren.item(j).getNodeName().equals("constraint")){
              offerDisplayConstraint = offerDisplayChildren.item(j);
              agreementDisplayConstraint = Agreement.createElement("constraint");
              display.appendChild(agreementDisplayConstraint);
            }
          }
          //copies the constraints for the permission from the offer to
          //the agreement if they exist
          if (offerDisplayConstraint!=null){
            this.copyConstraint(Agreement,
                                offerDisplayConstraint,
                                agreementDisplayConstraint,
                                "display");
          }
        }
      }
      if (nodeName.equals("print")) {
        Node offerPrint = offerPermissionChildren.item(i);
        NodeList offerPrintChildren = offerPrint.getChildNodes() ;
        //checking to see if requirements for this permission are satisfied.
        //if there are no requirement to be evaluated, requirements are seen
        //as satisfied
        boolean printReqSatisfied = true;
        for (j=0; j<offerPrintChildren.getLength(); j++){
          if (offerPrintChildren.item(j).getNodeName().equals("requirement")){
            Node printReq = offerPrintChildren.item(j);
```

```
        printReqSatisfied =
          this.checkRequirement(fullfilledReq, printReq, "print");
        }
      }
      if (printReqSatisfied){
        //Makes the print Element an attaches to agreement
        Element print = Agreement.createElement("print");
        permission.appendChild(print);

        //getting the constraint for this permission if exist
        Node offerPrintConstraint = null;
        Element agreementPrintConstraint = null;
        for (j=0; j<offerPrintChildren.getLength(); j++){
          if (offerPrintChildren.item(j).getNodeName().equals("constraint")){
            offerPrintConstraint = offerPrintChildren.item(j);
            agreementPrintConstraint = Agreement.createElement("constraint");
            print.appendChild(agreementPrintConstraint);
          }
        }
        //copies the constraints for the permission from the offer to
        //the agreement if they exist
        if (offerPrintConstraint!=null){
          this.copyConstraint(Agreement,
                              offerPrintConstraint,
                              agreementPrintConstraint,
                              "print");
        }

      }
    }

    if (nodeName.equals("play")) {
      Node offerPlay = offerPermissionChildren.item(i);
      NodeList offerPlayChildren = offerPlay.getChildNodes() ;
      //checking to see if requirements for this permission are satisfied
      //if there are no requirement to be evaluated, requirements are seen
      //as satisfied
      boolean playReqSatisfied = true;
      for (j=0; j<offerPlayChildren.getLength(); j++){
        if (offerPlayChildren.item(j).getNodeName().equals("requirement")){
          Node playReq = offerPlayChildren.item(j);
          playReqSatisfied =
            this.checkRequirement(fullfilledReq, playReq, "play");
        }
      }
      if (playReqSatisfied){
        //Makes the Display Element an attaches to agreement
        Element play = Agreement.createElement("play");
        permission.appendChild(play);

        //getting the constraint for this permission if exist
        Node offerPlayConstraint = null;
        Element agreementPlayConstraint = null;
        for (j=0; j<offerPlayChildren.getLength(); j++){
          if (offerPlayChildren.item(j).getNodeName().equals("constraint")){
            offerPlayConstraint = offerPlayChildren.item(j);
            agreementPlayConstraint = Agreement.createElement("constraint");
            play.appendChild(agreementPlayConstraint);
          }
        }
        //copies the constraints for the permission from the offer to
        //the agreement if they exist
        if (offerPlayConstraint!=null){
          this.copyConstraint(Agreement,
                              offerPlayConstraint,
                              agreementPlayConstraint,
                              "play");
        }
      }
    }
```

```
        if (nodeName.equals("execute")) {
          Node offerExecute = offerPermissionChildren.item(i);
          NodeList offerExecuteChildren = offerExecute.getChildNodes() ;
          //checking to see if requirements for this permission are satisfied
          //if there are no requirement to be evaluated, requirements are seen
          //as satisfied
          boolean executeReqSatisfied = true;
          for (j=0; j<offerExecuteChildren.getLength(); j++){
            if (offerExecuteChildren.item(j).getNodeName().equals("requirement")){
              Node executeReq = offerExecuteChildren.item(j);
              executeReqSatisfied =
                this.checkRequirement(fullfilledReq, executeReq, "execute");
            }
          }
          if (executeReqSatisfied){
            //Makes the Display Element an attaches to agreement
            Element execute = Agreement.createElement("execute");
            permission.appendChild(execute);

            //getting the constraint for this permission if exist
            Node offerExecuteConstraint = null;
            Element agreementExecuteConstraint = null;
            for (j=0; j<offerExecuteChildren.getLength(); j++){
              if (offerExecuteChildren.item(j).getNodeName().equals("constraint")){
                offerExecuteConstraint = offerExecuteChildren.item(j);
                agreementExecuteConstraint = Agreement.createElement("constraint");
                execute.appendChild(agreementExecuteConstraint);
              }
            }
            //copies the constraints for the permission from the offer to
            //the agreement if they exist
            if (offerExecuteConstraint!=null){
              this.copyConstraint(Agreement,
                                  offerExecuteConstraint,
                                  agreementExecuteConstraint,
                                  "execute");
            }
          }
        }
      }
    }
  }
/*****************************************************************************/

  /**
   * <p>Description :Just a helper method for the generateAgreementParty method.<br>
   * Summarizes the payments the user has opted to pay</p>
   *
   * @param fullfilledReq The requirements the user has fullfilled
   * @return a vector with a totalPayment and a currency
   */
  private Vector calculatePayment (Vector fullfilledReq){

    String currency = null;
    int totalAmount = 0;;
    Vector calculated = new Vector();

    Iterator reqIterator = fullfilledReq.iterator() ;
    while (reqIterator.hasNext()){
      String requirement = (String)reqIterator.next();
      StringTokenizer reqTokenizer = new StringTokenizer(requirement,",");
      while (reqTokenizer.hasMoreTokens()){
        String keyValue = reqTokenizer.nextToken();
        StringTokenizer keyValueTokenizer = new StringTokenizer(keyValue, "=");
        int i=1;
        String key = null;
        String value = null;
        while (keyValueTokenizer.hasMoreTokens()){
          if (i==1){
```

```
          key = keyValueTokenizer.nextToken();
        }
        else{
          value = keyValueTokenizer.nextToken();
        }
        i++;
      }
      if (key.equals("currency")){
        currency = value;
      }
      if (key.equals("amount")){
        Integer integer = new Integer(value);
        totalAmount = totalAmount + integer.intValue();
      }
    }
  }
  calculated.add(currency);
  Integer tempInteger = new Integer(totalAmount);
  calculated.add(tempInteger);

  return calculated;
 }

/*****************************************************************************/

 /**
  * <p>Description: Just a helper method for the generateAgreement method.<br>
  * Generates the party part of the agreement, adding the user element and copying
  * the rightsholder elements from the offer to the agreement. Prints to a file
  * the details of the agreement created needed for later distribution of attribution
  * to the rightsholders by the system</p>
  *
  * @param agreement the agreement element of the agreement under wich the party part
  * of the agreement is to be added
  * @param userName The registered username of the user making the agreement
  * @param collectionId The collection the resource is in
  * @param itemId The identifier for the item in the collection
  * @param fullfilledReq The requirements the user has fullfilled
  * @param offerParty The party part of the offer
  * @param Agreement a reference to the agreement in making
  */
 private void generateAgreementParty(Element agreement,
                                     String userName,
                                     String collectionId,
                                     String itemId,
                                     Vector fullfilledReq,
                                     Node offerParty,
                                     Document Agreement){
  String tempText = null;
  try{
    //getting the total amount that is paid
      Vector payment = this.calculatePayment(fullfilledReq) ;
    //Opening the agreement report file for printing
      FileWriter writer = new FileWriter("c:\\report.txt",true);
      BufferedWriter buffWriter = new BufferedWriter(writer);

    //printing to report starting with an initial empty line
      tempText = "";
      buffWriter.write(tempText,0,tempText.length());
      buffWriter.newLine();

      tempText = "AGREEMENT";
      buffWriter.write(tempText,0,tempText.length());
      buffWriter.newLine();

      tempText = "  User = "+userName;
      buffWriter.write(tempText,0,tempText.length());
      buffWriter.newLine();

      tempText = "  collectionId = "+collectionId+" itemId = "+itemId;
```

```
    buffWriter.write(tempText,0,tempText.length());
    buffWriter.newLine();

    buffWriter.flush();

  //summing up the amount that is paid and is to be devided among rightsholders
    tempText = "  Total payment generated: ";
    Vector totalPaymentVector = this.calculatePayment(fullfilledReq);
    Iterator totPay = totalPaymentVector.iterator();
    while (totPay.hasNext()) {
      tempText = tempText + totPay.next()+" ";
    }
    buffWriter.write(tempText,0,tempText.length());
    buffWriter.newLine();
    buffWriter.flush();

  //Adding the party tag to the agrement
    Element party = Agreement.createElement("party");
    agreement.appendChild(party);

  //constructing the userpart of the agreement
    Element user = Agreement.createElement("user");
    Element userContext = Agreement.createElement("context");
    Element userContextName = Agreement.createElement("name");
    Text userContextNameValue = Agreement.createTextNode(userName);
    userContextName.appendChild(userContextNameValue);
    userContext.appendChild(userContextName);
    user.appendChild(userContext);
    party.appendChild(user);
  //constructing the rightsholders of the agreement
    NodeList offerPartyChildren = offerParty.getChildNodes();
    for (int i=0; i<offerPartyChildren.getLength(); i++){
      String offerPartyChildName = offerPartyChildren.item(i).getNodeName() ;
      if (offerPartyChildName.equals("rightsholder")){
        //generates the agreements rightsholder element and adds to agreement
        Element rightsholder = Agreement.createElement("rightsholder");
        party.appendChild(rightsholder);
        //printing to report
        tempText = "  Rightsholder ";
        buffWriter.write(tempText, 0, tempText.length());
        buffWriter.newLine();
        //traverses the rightsholder Node
        Node rightsholderNode = offerPartyChildren.item(i);
        NodeList rightsholderChildren = rightsholderNode.getChildNodes();
        for (int j=0; j<rightsholderChildren.getLength(); j++){
          String rightsholderChildName =
              rightsholderChildren.item(j).getNodeName();
          if (rightsholderChildName.equals("context")){
            Element agreementContext = Agreement.createElement("context");
            rightsholder.appendChild(agreementContext);
            //traverses the context Node
            Node context = rightsholderChildren.item(j);
            NodeList contextChildren = context.getChildNodes();
            for (int k=0;k<contextChildren.getLength();k++){
              String contextChildName = contextChildren.item(k).getNodeName();
              if (contextChildName.equals("name")){
                //making the name part of the context
                Element agreementContextName = Agreement.createElement("name");
                agreementContext.appendChild(agreementContextName);
                tempText = "     Name: ";
                //getting the value of name
                Node contextName = contextChildren.item(k);
                if (contextName.hasChildNodes()){
                  String contextNameValue =
                    contextName.getFirstChild().getNodeValue();
                  Text agreementContextNameValue =
                    Agreement.createTextNode(contextNameValue);
                  agreementContextName.appendChild(agreementContextNameValue);
                  tempText = tempText + contextNameValue;
                }
```

```
        buffWriter.write(tempText, 0 , tempText.length());
        buffWriter.newLine();
        buffWriter.flush();
       }
       if (contextChildName.equals("uid")){
        tempText="    uid ";
        Node contextUid = contextChildren.item(k);
        NamedNodeMap attributes = contextUid.getAttributes();
        Node idscheme = attributes.getNamedItem("idscheme");
        String attributeValue = idscheme.getLastChild().getNodeValue();
        tempText = tempText+"idscheme "+attributeValue+" : ";
        //getting the child value for the offers uid
        String nodeValue = contextUid.getFirstChild().getNodeValue();
        tempText = tempText + nodeValue;
        //creating the corresponding construct in the agreement
        Element uid = Agreement.createElement("uid");
        uid.setAttribute("idscheme", attributeValue);
        Text uidText = Agreement.createTextNode(nodeValue);
        uid.appendChild(uidText);
        agreementContext.appendChild(uid);
        buffWriter.write(tempText,0,tempText.length());
        buffWriter.newLine();
        buffWriter.flush();
       }
       if (contextChildName.equals("URLreference")){
        //making the name part of the context
        Element agreementContextURL = Agreement.createElement("URLreference");
        agreementContext.appendChild(agreementContextURL);
        tempText = "    URLreference: ";
        //getting the value of URLreference
        Node contextURL = contextChildren.item(k);
        if (contextURL.hasChildNodes()){
          String contextURLValue =
            contextURL.getFirstChild().getNodeValue();
          Text agreementContextURLValue =
            Agreement.createTextNode(contextURLValue);
          agreementContextURL.appendChild(agreementContextURLValue);
          tempText = tempText + contextURLValue;
        }
        buffWriter.write(tempText, 0 , tempText.length());
        buffWriter.newLine();
        buffWriter.flush();
       }
      }
     }
     if (rightsholderChildName.equals("attribution")){

      tempText = "    attribution";
      buffWriter.write(tempText, 0, tempText.length());
      buffWriter.newLine();
      buffWriter.flush();
      //making and adding the attribution Element
      Element attributionElement = Agreement.createElement("attribution");
      rightsholder.appendChild(attributionElement);
      //traversing the offers attribution Part
      Node attribution = rightsholderChildren.item(j);
      NodeList attributionChildren = attribution.getChildNodes();
      for (int l=0; l<attributionChildren.getLength(); l++){
        String attributionChildName =
           attributionChildren.item(l).getNodeName();
        if (attributionChildName.equals("percentage")){
          //making the percentage part of the attribution
          Element agreementPercentage = Agreement.createElement("percentage");
          attributionElement.appendChild(agreementPercentage);
          //for printing to the report
          tempText = "      percentage:";
          //getting the value of percentage
          Node percentageNode = attributionChildren.item(l);
          if (percentageNode.hasChildNodes()){
            String percentageValue =
```

```
                percentageNode.getFirstChild().getNodeValue();
               Text percentageNodeText =
                Agreement.createTextNode(percentageValue);
               agreementPercentage.appendChild(percentageNodeText);
               tempText = tempText + percentageValue;
              }
              buffWriter.write(tempText, 0 , tempText.length());
              buffWriter.newLine();
              buffWriter.flush();
             }

             if (attributionChildName.equals("fixedamount")){
              Node fixedAmountNode = attributionChildren.item(l);
              //making the agreements representation of fixed amount
              Element fixedAmountElement =
                 Agreement.createElement("fixedamount");
              attributionElement.appendChild(fixedAmountElement);
              //writing to report
              tempText = "      fixedamount ";
              //traversing the fixedamount node of the offer
              NodeList fixedAmountChildren = fixedAmountNode.getChildNodes();
              for (int m=0; m<fixedAmountChildren.getLength();m++){
               if (fixedAmountChildren.item(m).getNodeName().equals("payment")){
                Element paymentElement = Agreement.createElement("payment");
                fixedAmountElement.appendChild(paymentElement);
                Node paymentNode = fixedAmountChildren.item(m);
                NodeList paymentChildren = paymentNode.getChildNodes();
                for (int n=0; n<paymentChildren.getLength();n++){
                 String paymentChildName =
                  paymentChildren.item(n).getNodeName();
                 if (paymentChildName.equals("amount")){
              Node amountNode = paymentChildren.item(m);
              NamedNodeMap attributes = amountNode.getAttributes();
              Node currency = attributes.getNamedItem("currency");
              String attributeValue = currency.getLastChild().getNodeValue();
              tempText = tempText+"currency "+attributeValue+" : ";
              //getting the child value for the amount
              String nodeValue = amountNode.getFirstChild().getNodeValue();
              tempText = tempText + nodeValue;
              buffWriter.write(tempText, 0, tempText.length());
              buffWriter.newLine();
              //creating the corresponding construct in the agreement
              Element amount = Agreement.createElement("amount");
              amount.setAttribute("currency", attributeValue);
              Text amountText = Agreement.createTextNode(nodeValue);
              amount.appendChild(amountText);
              paymentElement.appendChild(amount);
                   }
                  }
                 }
                }
               }
              }
             }
             buffWriter.flush();
            }
           }

   //prints remaining text in buffer and closes BufferedWriter
    buffWriter.flush();
    buffWriter.close();
  }
  catch (Exception e){
   System.out.println("Exception thrown");
   e.printStackTrace(System.out);
   System.out.println(e.getMessage());
  }
 }
```

```
/*****************************************************************************/

 /**
  * <p>Description: generates an agreement and stores it in a database</p>
  *
  * @param username the registered username of the user making the agreement
  * @param collectionId The collection the resource is n
  * @param itemId The identifier for the item in the collection
  * @param fullfilledReq The requirements the user has fullfilled
  * @throws Exception Any exceptions is thrown back to the user to be handled
  * there
  */
 public void generateAgreement(String username,
                               String collectionId,
                               String itemId,
                               Vector fullfilledReq)
                               throws Exception{

   NodeList nodeList = null;
   int i,j;
   String nodeName = null;
   Node offerNode = null;
   Node offerContext = null;
   Node offerAsset = null;
   Node offerPermission = null;
   Node offerParty = null;

   try{

     //making a copy of the vector that can be manipulated with,
     //keeping the original untouched for later use
     Vector fullfilledReq_Copy = new Vector();
     //used for traversing the original
     Iterator origIterator = fullfilledReq.iterator();
     //copying entry by entry
     while (origIterator.hasNext()){
       fullfilledReq_Copy.add(origIterator.next());
     }

     //making a connection to the database
       xmlDatabaseFrontend xmldb = new xmlDatabaseFrontend();

     //do I need the profile to make an agreement
     //getting the users profile
       Document profile = xmldb.get_Profile(username);

     //getting the initial offer for the material
       Document offer = xmldb.get_Offer(collectionId, itemId);
       Element documentRoot = offer.getDocumentElement();

       //getting the various mandatory parts of the offer
       nodeList = documentRoot.getChildNodes();
       for (i=0; i<nodeList.getLength();i++){
         nodeName = nodeList.item(i).getNodeName();

         if (nodeName.equals("offer")){
           //offerNode found. getting the children as Nodes
           offerNode = nodeList.item(i);
           nodeList = offerNode.getChildNodes();
           for(j=0;j<nodeList.getLength();j++){
             nodeName = nodeList.item(j).getNodeName();
             if (nodeName.equals("context")){
               offerContext = nodeList.item(j);
             }
             if (nodeName.equals("asset")){
               offerAsset = nodeList.item(j);
             }
             if (nodeName.equals("permission")){
               offerPermission = nodeList.item(j);
             }
```

```
            if (nodeName.equals("party")){
              offerParty = nodeList.item(j);
            }
           }
          }
         }//end for

        //making a new Dom Document to hold the generated agreement
        //no namespace, root=="rights", no DOCTYPE
         Document Agreement = null;
         DOMImplementation domImpl = null;
         Element root = null;
         domImpl = new DOMImplementationImpl();
         Agreement = domImpl.createDocument(null, "rights", null);
         root = Agreement.getDocumentElement();

        //creating the agreement element and adding to root
         Element agreement = Agreement.createElement("agreement");
         root.appendChild(agreement);

        //making the context part.
         this.generateAgreementContext(agreement, Agreement);

        //Making asset part
         this.generateAgreementAsset(agreement, Agreement, collectionId, itemId, offerAsset);

        //making permission part (sending the copy of the vektor as this
        //copy will be manipulated with
         this.generateAgreementPermissions(agreement,
                                  Agreement,
                                  offerPermission,
                                  fullfilledReq_Copy);
        //making the party type
        //sending the original vector here since the copy is tampered with
         this.generateAgreementParty(agreement,
                                username,
                                collectionId,
                                itemId,
                                fullfilledReq,
                                offerParty,
                                Agreement);

        //stores the generated agreement in the database
          xmldb.store_Agreement(username, collectionId, itemId, Agreement);
       }
       catch(Exception e){
          throw e;
       }
     }
/*****************************************************************************/

     /**
      * <p>Description: Helpermethod for the generateTicket method<br>
      * Evaluates if a constraint can be met by the user taking into consideration
      * the users profile and his current connection tp the Internet</p>
      *
      * @param constraintNode The constraint from the agreement to be evaluated
      * @param username The registered username of the user
      * @param IPAdress The IPadress of the users current Internet connection
      * @param ticket The vector to be given an error message if the constraint
      * can't be met.
      * @param parentPermission The parent permission of the constraint to be
      * evaluated
      * @return boolean value indication if the constraint has been met
      */
     private boolean evaluateConstraints(Node constraintNode,
                                String username,
                                String IPAdress,
                                Vector ticket,
                                String parentPermission){
```

```
//the value to be returned
  boolean constraintsSatisfied = true;
//getting the varius constraints
  NodeList constraintChildren = constraintNode.getChildNodes();
  for (int i=0; i<constraintChildren.getLength(); i++){
    String constraintChildName = constraintChildren.item(i).getNodeName();

    if (constraintChildName.equals("count")){
      //getting the max value
      String max_String;
      int max_int = 0;
      Node countNode = constraintChildren.item(i);
      NodeList countNodeChildren = countNode.getChildNodes();
      for (int j=0; j<countNodeChildren.getLength(); j++){
        String countNodeChildName = countNodeChildren.item(j).getNodeName();
        if (countNodeChildName.equals("max")){
          //getting the max value
          Node maxNode = countNodeChildren.item(j);
          max_String = maxNode.getFirstChild().getNodeValue();
          //converting to an int
          Integer integer = new Integer(max_String);
          max_int = integer.intValue();
        }
      }
      if (parentPermission.equals("global")){
        //the value of a global constraint for count has to be evaluated
        //against the number of times the individual permission has been
        //executed. This is stored under the individual permission in the
        //agreement. Therefore the value is just stored in a global variable
        //until this method is called again to evaluate the actual
        //permission (local constraint)
        globalConstraintCountValue = max_int;
      }
      else{
        //the value for count max compared to the value of executed has to
        //be the most restricted of the value defined globally and the one
        //declared locally, unless the value saved in
        //globalConstraintCountValue==0 in case there were no declaration of
        //a global count
        boolean globallyDeclared;
        int compareValue;
        if (globalConstraintCountValue==0){
          //there were no global count restriction, using the local count as
          //comparevalue
          globallyDeclared = false;
          compareValue = max_int;
        }
        else{
          if (globalConstraintCountValue<max_int){
            compareValue = globalConstraintCountValue;
            globallyDeclared = true;
          }
          else {
            compareValue = max_int;
            globallyDeclared = false;
          }
        }
        //getting the value of executed
        String executed_String = "";;
        int executed_int = 0; //will always be overwritten, just initializing
        NodeList countNodeChildren2 = countNode.getChildNodes();
        for (int j=0; j<countNodeChildren2.getLength(); j++){
          String countNodeChild2Name = countNodeChildren2.item(j).getNodeName();
          if (countNodeChild2Name.equals("executed")){
            Node executedNode = countNodeChildren2.item(j);
            executed_String = executedNode.getFirstChild().getNodeValue();
            //converting to int
            Integer integer = new Integer(executed_String);
            executed_int = integer.intValue();
```

```
        }
       }
       //ready to do the comparison
       if (executed_int<compareValue){
        //permission can be granted
        /*
        //incrementing the value of executed
        executed_int = executed_int + 1;
        Integer testInteger = new Integer(executed_int);
        String newValue =  testInteger.toString();
        NodeList countNodeChildren3 = countNode.getChildNodes();
        for (int j=0; j<countNodeChildren3.getLength(); j++){
         String countNodeChild3Name =
          countNodeChildren3.item(j).getNodeName();
         if (countNodeChild3Name.equals("executed")){
          Node executedNode = countNodeChildren3.item(j);
          executedNode.getFirstChild().setNodeValue(newValue);
         }
        }
        */
       }
       else{
        //writing error to vector
        String error;
        if (globallyDeclared){
         error = "error=You have exeeded the maximum number of times ";
         error = error + "you can execute this permission as defined in ";
         error = error + "a global constraint";
         ticket.add(error);
        }
        else{
         error = "error=You have exeeded the maximum number of times ";
         error = error + "you can execute this permission as defined for ";
         error = error + "this permission";
         ticket.add(error);
        }
        //setting the return value to false
        constraintsSatisfied = false;
       }
      }
     }

     if (constraintChildName.equals("dateRange")){
      //getting start and enddates
      String startDate = "";
      String endDate = "";
      Node dateRangeNode = constraintChildren.item(i);
      NodeList dateRangeChildren =  dateRangeNode.getChildNodes();
      for (int j=0; j<dateRangeChildren.getLength(); j++){
       String dateRangeChildName = dateRangeChildren.item(j).getNodeName();
       if (dateRangeChildName.equals("start")){
        Node dateRangeStart = dateRangeChildren.item(j);
        startDate = dateRangeStart.getFirstChild().getNodeValue();
       }
       if (dateRangeChildName.equals("end")){
        Node dateRangeEnd = dateRangeChildren.item(j);
        endDate = dateRangeEnd.getFirstChild().getNodeValue();
       }
      }
      //generate startDate as a GregorianCalendar
      StringTokenizer startDateTokenizer = new StringTokenizer(startDate, "-");
      String startyear = null;
      String startmonth = null;
      String startdate = null;
      int startyear_int = 0;
      int startmonth_int = 0;
      int startdate_int = 0;
      int j = 1;
      while (startDateTokenizer.hasMoreTokens()){
       String temp = startDateTokenizer.nextToken();
```

```
  if (j==1){
    startyear = temp;
    Integer startyearInteger = new Integer(startyear);
    startyear_int = startyearInteger.intValue();
  }
  if (j==2){
    startmonth = temp;
    Integer startmonthInteger = new Integer(startmonth);
    startmonth_int = startmonthInteger.intValue();
  }
  if (j==3){
    startdate= temp;
    Integer startdateInteger = new Integer(startdate);
    startdate_int = startdateInteger.intValue();
  }
  j++;
}
GregorianCalendar startDateCal =
  new GregorianCalendar(startyear_int, startmonth_int, startdate_int);

//generates endDate as a GregorianCalendar
StringTokenizer endDateTokenizer = new StringTokenizer(endDate, "-");
String endyear = null;
String endmonth = null;
String enddate = null;
int endyear_int = 0;
int endmonth_int = 0;
int enddate_int = 0;
j = 1;
while (endDateTokenizer.hasMoreTokens()){
  String temp = endDateTokenizer.nextToken();
  if (j==1){
    endyear = temp;
    Integer endyearInteger = new Integer(endyear);
    endyear_int = endyearInteger.intValue();
  }
  if (j==2){
    endmonth = temp;
    Integer endmonthInteger = new Integer(endmonth);
    endmonth_int = endmonthInteger.intValue();
  }
  if (j==3){
    enddate= temp;
    Integer enddateInteger = new Integer(enddate);
    enddate_int = enddateInteger.intValue();
  }
  j++;
}
GregorianCalendar endDateCal =
  new GregorianCalendar(endyear_int, endmonth_int, enddate_int);

//getting the value of this date (today)
Calendar rightNow = Calendar.getInstance();
//is today before end and after start??
boolean beforeEnd;
beforeEnd = rightNow.before(endDateCal);
boolean afterStart;
afterStart = rightNow.after(startDateCal);

//determining if satisfied
if (!(beforeEnd && afterStart)){
  //permission can't be granted
  String error;
  if (parentPermission.equals("global")){
    error = "error=You are not within the dateRange in which permissions";
    error = error + "can be exerciced according to your agreement";
    ticket.add(error);
  }
  else{
    error = "error=You are not within the dateRange in which this permission ";
```

```
            error = error + "can be exerciced according to your agreement";
            ticket.add(error);
           }
           constraintsSatisfied = false;
          }
        }

        if (constraintChildName.equals("network")){
          //System.out.println("constraint network found");
          //System.out.println("no evaluation method for this constraint");
        }

        if (constraintChildName.equals("individual")){
          Node individualNode = constraintChildren.item(i);
          String restrictedToName = individualNode.getFirstChild().getNodeValue();
          if (!restrictedToName.equals(username)){
            String error;
            error = "error=This permission has been restricted to a single user ";
            error = error + "and you are not the one";
            ticket.add(error);
            constraintsSatisfied = false;
          }
        }

      }
    return constraintsSatisfied;
  }

/*****************************************************************************/

  /**
   *<p>Description: generates a ticket which gives the user access to a permission
   *if all constraints of the agreement are met.</p>
   * @param collectionId The collectionId of the resource
   * @param itemId The itemId of the resource
   * @param permission The permission the user wants to exercise
   * @param username The registered username of the user
   * @param IPAdress The IPAdress of the users current Internet connection
   * @return a vector implementing the ticket
   * @throws Exception Any exeption is thrown back to the calling class to be
   * handled there
   */
  public Vector generateTicket(String collectionId,
                               String itemId,
                               String permission,
                               String username,
                               String IPAdress)throws Exception{

    int i,j;
    Node agreementNode = null;
    NodeList agreementChildren = null;
    Node permissionNode = null;
    NodeList permissionChildren = null;
    Node permissionElementNode = null;
    String nodeName;
    Document agreement = null;

    //making the ticket
    Vector ticket = new Vector();

    //getting the offer from the database
      xmlDatabaseFrontend xmldb = null;
      try{
        xmldb = new xmlDatabaseFrontend();
        agreement = xmldb.get_Agreement(username, collectionId, itemId);
      }
      catch(Exception e){
        throw e;
      }
```

```
//getting the root of the agreement
  Element documentRoot = agreement.getDocumentElement();

//getting the permission part of the agreeement
  NodeList nodeList = documentRoot.getChildNodes();
  for (i=0; i<nodeList.getLength();i++){
    nodeName = nodeList.item(i).getNodeName();
    if (nodeName.equals("agreement")){
      agreementNode = nodeList.item(i);
      agreementChildren = agreementNode.getChildNodes();
      for(j=0;j<agreementChildren.getLength();j++){
        nodeName = agreementChildren.item(j).getNodeName();
        if (nodeName.equals("permission")){
          permissionNode = agreementChildren.item(j); //holds tag permission
        }
      }
    }
  }//end for


//checking if the permission is part of the agreement
  boolean permissionExists = false;
  permissionChildren = permissionNode.getChildNodes() ;
  for (i=0; i<permissionChildren.getLength(); i++){
    nodeName = permissionChildren.item(i).getNodeName();
    if (nodeName.equals(permission)){
      permissionExists = true;
    }
  }

//if the permission is not part of the agreement, prints that to the vector
//and just returns it.
if (!permissionExists){
  String statusElement = "ticketGranted=no";
  String errorElement =
      "error=The permission "+permission+" is not part of your agreement";
  ticket.add(statusElement);
  ticket.add(errorElement);
}
else{
  //checks to se if global and local constraints are satisfied
  boolean allConstraintsOK = true;

  //checking global constraints if they exist
  boolean globalConstraintsOK;
  for (i=0; i<permissionChildren.getLength(); i++){
    nodeName = permissionChildren.item(i).getNodeName();
    if (nodeName.equals("constraint")){
      Node globalConstraintNode = permissionChildren.item(i);
      //global constraints exists. Evaluating them. If the method returns false
      //one or more of the defined constraints was not met, which means the
      //permission cant be granted. Reasons are recorded in the vektor passed
      //by reference to the method
      globalConstraintsOK = this.evaluateConstraints(globalConstraintNode,
                                        username,
                                        IPAdress,
                                        ticket,
                                        "global");
      //if global constraints are not met, this permission cant be given
      //sets the allConstraintsOK== false to indicate this for later
      if (!globalConstraintsOK)
        allConstraintsOK = false;
    }
  }

  //checking local constraints if they exist
  boolean localConstraintsOK = false;
  for (i=0; i<permissionChildren.getLength(); i++){
    nodeName = permissionChildren.item(i).getNodeName();
    if (nodeName.equals(permission)){
```

```
      permissionElementNode = permissionChildren.item(i);
      NodeList permissionElementChildren =
        permissionElementNode.getChildNodes();
      for (j=0; j<permissionElementChildren.getLength();j++){
        String permissionElementChildName =
          permissionElementChildren.item(j).getNodeName();
        if (permissionElementChildName.equals("constraint")){
          Node localConstraintNode = permissionElementChildren.item(j);
          //global constraints exists. Evaluating them. If the method returns false
          //one or more of the defined constraints was not met, which means the
          //permission cant be granted. Reasons are recorded in the vektor passed
          //by reference to the method
          localConstraintsOK = this.evaluateConstraints(localConstraintNode,
                                                        username,
                                                        IPAdress,
                                                        ticket,
                                                        permission);
          //sets the allConstraintsOK== false to indicate this for later
          if (!localConstraintsOK){
            allConstraintsOK = false;
          }
        }
      }
    }
  }

  if (allConstraintsOK){
    //recording in the agreement that the permission has been executed once
    System.out.println("updating the agreement for permission "+permissionElementNode.getNodeName() );

    boolean agreementUpdated = false;

    //tries first to update an existing value if it exits
    NodeList permissionElementChildren = permissionElementNode.getChildNodes();
    for (i=0; i<permissionElementChildren.getLength(); i++){
      nodeName = permissionElementChildren.item(i).getNodeName();
      if (nodeName.equals("constraint")){
        System.out.println("nodeName "+nodeName);
        Node constraintNode = permissionElementChildren.item(i);
        NodeList constraintChildren = constraintNode.getChildNodes();
        for (j=0; j<constraintChildren.getLength(); j++){
          nodeName = constraintChildren.item(j).getNodeName();
          if (nodeName.equals("count")){
            Node countNode = constraintChildren.item(j);
            NodeList countChildren = countNode.getChildNodes();
            for (int k=0; k<countChildren.getLength(); k++){
              nodeName = countChildren.item(k).getNodeName();
              if (nodeName.equals("executed")){
                Node executedNode = countChildren.item(k);
                //getting the value as string and converting to an int
                String oldValue_String =
                  executedNode.getFirstChild().getNodeValue();
                Integer convertInteger = new Integer(oldValue_String);
                int oldValue_int = convertInteger.intValue();
                //making the new value and converting to String
                int newValue_int = oldValue_int + 1;
                Integer convertInteger2 = new Integer(newValue_int);
                String newValue_String = convertInteger2.toString();
                //overwriting with new value
                executedNode.getFirstChild().setNodeValue(newValue_String);
                agreementUpdated = true;
              }
            }
          }
        }
      }
    }
    if (!agreementUpdated){
      //there was no count construct for this permission, so one has to be
      //created so that it can be incremented when issuing tickets
```

```
        Element constraint = agreement.createElement("constraint");
      }
    }

    //finishing up the vector by adding element telling if the permission is granted
    if (allConstraintsOK){
      ticket.add("ticketGranted=yes");
    }
    else{
      ticket.add("ticketGranted=no");
    }
  }

  //storing the agreement back to the database to record the changes
  xmldb.store_Agreement(username, collectionId, itemId, agreement);


  //returning the ticket to calling class
  return ticket;
 }

/*****************************************************************************/
}
```

reap.DRMPackager.DRMPackager.java

```
package reap.DRMPackager;

//importing utility classes
import reap.util.database.xml.xmlDatabaseFrontend;
import reap.util.database.resource.resourceDatabaseFrontend;
import reap.util.DRMPackage;
import reap.util.FileEncapsulator;

//import for handeling xml Dom
import org.w3c.dom.*;

/*****************************************************************************/

/**
 * <p>Title: DRMPackager</p>
 * <p>Description: This is the class acting as the DRMPackager which delivers
 * a Content Package containing the material to the DRM Controller
 * (And in future versions could insert material metadata and rights into
 * the collections</p>
 * <p>Copyright: Copyright (c) 2002</p>
 * @author Øyvind Vestavik
 * @version 1.0
 */

public class DRMPackager{

/*****************************************************************************/

  /**
   * <p>Description: Default constructor</p>
   */
  public DRMPackager(){
  }

/*****************************************************************************/



  /**
   * <p>Description: Making a DRMPackage an returning it to the caller </p>
   * @param collectionId
   * @param itemId
```

```
   * @param userId
   * @return DRMPackage
   */
  public DRMPackage getPackage(String collectionId,
                               String itemId,
                               String userId) {

    //making drmPackage to be returned
    DRMPackage contentPackage = new DRMPackage();

    //Putting the material into the package
    try{
      resourceDatabaseFrontend resourceDatabase =
                                 new resourceDatabaseFrontend();
      FileEncapsulator encapsulator = new FileEncapsulator();
      encapsulator = resourceDatabase.getMaterial(collectionId, itemId);
      contentPackage.material = encapsulator;
      contentPackage.containsMaterial = true;
    }
    catch(Exception e){
      contentPackage.containsMaterial = false;
    }

    /*
    //Putting XML Agreement into the package
    try {
      xmlDatabaseFrontend xmlDatabase = new xmlDatabaseFrontend();
      Document offer = xmlDatabase.get_Offer(collectionId, itemId);
      drmPackage.rights = offer;
      drmPackage.containsOffer = true;
    }
    catch(Exception e){
      drmPackage.containsOffer = false;
    }
    */

    return contentPackage;
  }

}
```

reap.util.DRMPackage.java

```
package reap.util;

import java.io.*;
import org.w3c.dom.*;

  /**
   * <p>Description: a DRMPackage contains the material and the rights of
   * a material. In this prototype this is just an encapsulator for the material
   * and some information about itself and its material</p>
   * <p>Copyright: Copyright (c) 2002</p>
   * <p>Company: </p>
   * @author Øyvind Vestavik
   * @version 1.0
   */
  public class DRMPackage{
    /**
     * <p>Description: indicates if this package contains any material</p>
     */
    public boolean containsMaterial=false;
    /**
     * <p>Description: The rightsmodel for the material. In this prototype this
     * field is not in use</p>
     */
    public Document rights = null;
    /**
```

```
    * <p>Description: This field contains the material itself as a sequence of bits</p>
    */
   public FileEncapsulator material = null;
   /**
    * <p>Description: This field indicates the size of the file in bits </p>
    */
   int fileSize =0;
   /**
    * <p>Description: This field contains the filename of the resource</p>
    */
   String fileName=null;
  }
```

reap.util.FileEncapsulator.java

```
package reap.util;

/**
 * <p>Description: This class encapsulates a resource/file and some information
 * about the file</p>
 * <p>Copyright: Copyright (c) 2002</p>
 * <p>Company: </p>
 * @author Øyvind Vestavik
 * @version 1.0
 */

public class FileEncapsulator{
  /**
   * <p>Description: This field contains the filename of the resource</p>
   */
  public String fileName=new String("fileName not set");
  /**
   * <p>Description: This field contains the filesize of the resource</p>
   */
  public int fileSize;
  /**
   * <p>Description: This array contains the contents of the file as an array of bytes</p>
   */
  public int []content;
}
```

reap.util.ProfileHandler.java

```
package reap.util;

import java.util.*;
import java.io.*;
import reap.util.database.xml.xmlDatabaseFrontend;
import org.w3c.dom.*;
import org.apache.xerces.dom.DOMImplementationImpl;

/**
 * <p>Title: </p>
 * <p>Description: This class receives user info from registration servlet and
 * makes a profile in the form of an xml DOM tree for the user.
 * It generates a password to insert it along with the username in the profile
 * and sends the profile to the xmlDataBaseFrontEnd class to be stored there.
 * Then the generated password is returned to the registraiton servlet</p>
 * <p>Copyright: Copyright (c) 2002 Øyvind Vestavik</p>
 * @author Øyvind Vestavik
 * @version 1.0
 */
public class ProfileHandler {

  /**
```

```
   * <p>Description: generates an eight character long password</p>
   *
   * @return String pwd. The generated password
   */
  private String generate_pwd(){
    String characters =
      new String("0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz");
    //generates password containing ASCII charachters from the String above
    Random numbergenerator = new Random();
    String pwd ="";
    int gen_number=0;
    for (int i=0; i<8; i++){
      gen_number = numbergenerator.nextInt(62);
      pwd = pwd + characters.charAt(gen_number);
    }
    return pwd;
  }

/*****************************************************************************/

  /**
   * <p>Description: Receives details about a user collected by servlet
   * reap.servlets.register and creates a profile in the database. Returns an
   * object containing the status of the delivered service and the generated
   * password</p>
   *
   * @param email The users email adress. Functions as username
   * @param fname First name of the user
   * @param lname Last name of the user
   * @return UsernamePwdHolder
   * @throws Exception
   */
  public UsernamePwdHolder make_profile(String email,
                        String fname,
                        String lname)
  {
    //global reference to object to be returned
    UsernamePwdHolder up = new UsernamePwdHolder();
    //generating password
    String pwd = generate_pwd();

    //making xml profile
    Document profile = null;
    DOMImplementation domImpl = null;
    Element root = null;
    try{
      //making a document with a root. no namespace, root=="profile", no DOCTYPE
      domImpl = new DOMImplementationImpl();
      profile = domImpl.createDocument(null, "profile", null);
      root = profile.getDocumentElement();

      //adding elements to document
      Element element = null;
      Text text = null;
      //adding element for username/mail to root
      element = profile.createElement("username");
      text = profile.createTextNode(email);
      element.appendChild(text);
      root.appendChild(element);
      //adding element for password to root
      element = profile.createElement("password");
      text = profile.createTextNode(pwd);
      element.appendChild(text);
      root.appendChild(element);
      //adding element for first name to root
      element = profile.createElement("fname");
      text = profile.createTextNode(fname);
      element.appendChild(text);
      root.appendChild(element);
      //adding element for last name to root
```

```
    element = profile.createElement("lname");
    text = profile.createTextNode(lname);
    element.appendChild(text);
    root.appendChild(element);

    //sending profile to storage
    xmlDatabaseFrontend xmldb = new xmlDatabaseFrontend();
    xmldb.store_Profile(email, profile);

    //setting propeties of the object returning the status of the operation
    up.setUsername(email);
    up.setPassword(pwd);
    up.setMessage("profile_stored");

   } //try

   catch (Exception e){
    up.setMessage("profile_not_stored");
   }
   //returning status of operation
   return up;
  }

/*****************************************************************************/

}
```

reap.util.UsernamePwdHolder.java

```
package reap.util;

/**
 * <p>Title: </p>
 * <p>Description: Just a keeper for a username and a password
 * with an optional message field that can be used for passing messages
 * between classes</p>
 * <p>Copyright: Copyright (c) 2002</p>
 * <p>Company: </p>
 * @author Øyvind Vestavik
 * @version 1.0
 */

public class UsernamePwdHolder {

  /**
   * <p>Description: contains the users username</p>
   */
  private String username = null;
  /**
   * <p>Description: contains the users password</p>
   */
  private String password = null;
  /**
   * <p>Description: contains an optional message</p>
   */
  private String message = null;

/*****************************************************************************/

  /**
   * <p>Description: Default constructor</p>
   */
  public UsernamePwdHolder() {
  }

/*****************************************************************************/

  /**
```

```
   * <p>Description: This contstruktor receives the users username, password and
   * optionally a message to be stored in the object</p>
   * @param username
   * @param password
   * @param message
   */
  public UsernamePwdHolder(String username, String password, String message) {
    this.username = username;
    this.password = password;
    this.message = message;
  }

/*****************************************************************************/

  /**
   * <p>Description: getMethod for username</p>
   * @return username
   */
  public String getUsername(){
    return username;
  }

  /**
   * <p>Description: getMethod for password</p>
   * @return password
   */
  public String getPassword(){
    return password;
  }

  /**
   * <p>Description: getMethod for message</p>
   * @return message
   */
  public String getMessage(){
    return message;
  }

  /**
   * <p>Description: setMethod for username</p>
   * @param username
   */
  public void setUsername(String username){
    this.username = username;
  }

  /**
   * <p>Description: setMethod for password</p>
   * @param password
   */
  public void setPassword(String password){
    this.password = password;
  }

  /**
   * <p>Description: setMethod for message</p>
   * @param message
   */
  public void setMessage(String message){
    this.message = message;
  }

}
```

reap.util.database.resource.IdentifierResolver.java

```
package reap.util.database.resource;
```

```
import java.io.*;
import java.util.*;

/**
 *
 * <p>Description: This class contains functionality for converting from a REAP
 * identification scheme containing a CollectionId and ItemId to a physical location.</p>
 * <p>Copyright: Copyright (c) 2002</p>
 * @author Øyvind Vestavik
 * @version 1.0
 */
public class IdentifierResolver {

  /**
   * <p>Description: Takes a collectionId and an itemId and returns the physical
   * location from where the resource can be retrieved</p>
   *
   * @param collectionId
   * @param itemId
   * @return A String containing a filepath to the material
   */
  public String getPhysicalLocation(String collectionId, String itemId){

    String read_collectionId=null;
    String read_itemId=null;
    String read_location=null;
    String physicalLocation = null;
    boolean found = false;

    try{
      //Constucting a filereader for the file containing physical locations
      File fileref =
        new File("C:\\utvikling\\jbproject\\testCollection\\itemregistry.txt");
      FileReader reader = new FileReader(fileref);
      BufferedReader buffread = new BufferedReader(reader);

      //Reading line by line from the tab separatedfile and gets the location
      //from the line containing the right entry
      String readLine = buffread.readLine();
      while (readLine!=null){
        StringTokenizer tokenizer = new StringTokenizer(readLine);
        read_collectionId = (String)tokenizer.nextElement();
        read_itemId = (String)tokenizer.nextElement();
        read_location = (String)tokenizer.nextElement();
        //if found==true, breaks out of loop
        found = ((read_itemId.equals(itemId))&&(read_collectionId.equals(collectionId)));
        if(found){
          physicalLocation = read_location;
          break;
        }
        readLine = buffread.readLine();
      }
    }

    catch (IOException e){
      System.out.println("file  not found");
    }

    //Insert escapechar for "\" incase windows path.
    String temp="";
    for (int i=0 ; i<physicalLocation.length(); i++){
      Character currentChar = new Character(physicalLocation.charAt(i));
      if (currentChar.toString().equals("\\")){
        temp = temp + (currentChar.toString())+"\\";
      }
      else{
        temp = temp + currentChar.toString();
      }
    }
    physicalLocation = temp;
```

```
    return physicalLocation;
  }
}
```

reap.util.database.resource.resourceDatabaseFrontend.java

```
package reap.util.database.resource;

import java.io.*;
import reap.util.FileEncapsulator;

/**
 * <p>Description: This class is responible for collecting the material the
 * user is requesting from a physical storage. It gets the location of the storage
 * from the IdentifierResolver class</p>
 * <p>Copyright: Copyright (c) 2002</p>
 * @author Øyvind Vestavik
 * @version 1.0
 */
public class resourceDatabaseFrontend {

  /**
   * <p>Retrieves the resource from its physical location and packs it into a
   * reap.util.FileEncapsulator object before returning this object to the caller</p>
   *
   * @param collectionId
   * @param itemId
   * @return reap.util.FileEncapsulator
   */
  public FileEncapsulator getMaterial(String collectionId, String itemId){

    //getting a reference to the material
    String physicalLocation = null;
    IdentifierResolver resolver = new IdentifierResolver();
    physicalLocation = resolver.getPhysicalLocation(collectionId, itemId);

    //making a fileobject to function as referense to the resource
    File myFile = new File(physicalLocation);

    //making the object to contain file content and file information
    FileEncapsulator encapsulator =
                    new FileEncapsulator();

    if (myFile.exists()){

      //getting the filename
      encapsulator.fileName = myFile.getName();

      //finding the files size and converting to an int
      //possible bug: What happens if the number of bytes in in the file is
      //larger than the range of an int?? Would have to be a huge(!) file, but..
      long fileSize_long = myFile.length();
      Long longWrapper = new Long(fileSize_long);
      int fileSize = longWrapper.intValue();

      //initiates the fileSize att of the object to be returned
      encapsulator.fileSize = fileSize;

      //int fileSize is used for allocating slots to the array
      //keeping the contents of the file
      encapsulator.content = new int[fileSize];

      //reading contents of file, storing one byte at a time to the
      //ResourceEncapsulator
      try{
        FileInputStream in =
          new FileInputStream(myFile);
```

```
      BufferedInputStream buff = new BufferedInputStream(in);
      int i=0;
      int nextByte = buff.read();
      while (nextByte!=(-1)){
        encapsulator.content[i] = nextByte;
        nextByte = buff.read();
        i++;
      }
      //closes inputStreams
      in.close();
     }
     catch(IOException e){
      e.printStackTrace();
     }
    }
    else{
     encapsulator.fileName = "No such file";
     encapsulator.fileSize = 0;
    }
    return encapsulator;
  }
/*****************************************************************************/
}
```

reap.util.database.xml.xmldatabaseFrontend.java

```
package reap.util.database.xml;

import org.w3c.dom.*;
import org.xmldb.api.base.*;
import org.xmldb.api.modules.*;
import org.xmldb.api.*;


/**
 * <p>Description: <blockquote>This class is responsible for storing and retrieving xml
 * documents like profiles, Offers and Agreements from an Xindice database
 * instance.<br><br>
 * flaw: There is no check on existence of profiles. If a user registers using
 * an already registered mail adress, the old profile stored under this username
 * will simply be written over..</P>
 *
 * <p>Copyright: Copyright (c) 2002. Øyvind Vestavik</p>
 *
 * @author Øyvind Vestavik
 * @version 1.0
 */

public class xmlDatabaseFrontend {


/*****************************************************************************/

  /**
   * <p>Description: Opens a collection in the xml Database based on the
   * parameter collectionName. This method is a local method not required by the
   * xmlDatabaseFrontEndInterface </p>
   *
   * @param collectionName The name of the collction to be opened
   * @return col a reference to the collection opened
   * @throws Exception If collection can't be opened, an Exception is thrown
   *         back to the caller for handeling there
   */
  private Collection openDB(String collectionName) throws Exception {

    //setting up reference to the right colllection in db and returning it
    Collection col = null;
      try {
```

```
        String driver = "org.apache.xindice.client.xmldb.DatabaseImpl";
        Class c = Class.forName(driver);

        Database database = (Database) c.newInstance();
        DatabaseManager.registerDatabase(database);

        col = DatabaseManager.getCollection(collectionName);
      return col;
    }
    catch (Exception e) {
      throw e;
    }
  }

/*****************************************************************************/

  /**
   * <p>Description: retrieves the users profile from a database and returns
   * it as an Dom Document. </p>
   *
   * @param username : Identifies which profile to retrieve
   * @return A Document (xml DOM tree) with all information stored about a user.
   * @throws Exception
   */
  public Document get_Profile(String username) throws Exception  {
    Document profile = null;
    Collection col = null;
    try {
      //Opening collection
      col = this.openDB("xmldb:xindice:///db/REAP/UserProfiles");
      //retrieving the right profile based on the username as key.
      XMLResource resource = (XMLResource) col.getResource(username.trim());
      profile = (Document) resource.getContentAsDOM();
    }
    catch (Exception e) {
      //indicates to calling class/method that profile could not be retrieved
      throw e;
    }
    finally {
      if (col != null) {
        col.close();
      }
    }
    //returning the profile
    return profile;
  } // end get_profile


/*****************************************************************************************************/

  /**
   * <p>Description: Stores a profile for a user </p>
   *
   * @param username : Identifies the profile
   * @param profile : Dom Document with all info about a user
   * @throws Exception
   */
  public void store_Profile(String username, Document profile) throws Exception {
    boolean known_user = false;
    Collection col = null;
    try {
      //opening collection
      col = this.openDB("xmldb:xindice:///db/REAP/UserProfiles");
      //storing profile.
      XMLResource resource =
          (XMLResource) col.createResource(username.trim(), XMLResource.RESOURCE_TYPE);
      resource.setContentAsDOM(profile);
      col.storeResource(resource);
    }
    catch (Exception e) {
```

```
    //indicates to calling class/method that profile could not be stored
    throw e;
   }
   finally {
    if (col != null) {
     col.close();
    }
   }
  } // end store_profile

/*****************************************************************************/

  /**
   * <p>Description: Returns an Agreement already negotiated governing
   *           the allowed consumption of a given material by a given user</p>
   *
   * @param username : Indicates the user that made the agreement
   * @param collectionId : Indicates the collection the Item is in.
   * @param itemId : Indicates the Item the agreement is about
   * @return A Dom Document containing the rights the user has obtained over
   *         this material including the constraints and obligations these rights
   *         are given and can be executed under.
   * @throws Exception
   */
  public Document get_Agreement(String username,
                          String collectionId,
                          String itemId)
                          throws Exception{
   Document agreement = null;
   Collection col = null;
   //setting up the ID for the agreement based on the parameters
   //note the spaces inserted for easy use with commandline tool of the
   //databaseimplemention..
   String id = ("user "+username.trim()+" col "+
            collectionId.trim()+" item "+itemId.trim());
   //getting agreement with this string
   try {
    //Opening collection
    col = this.openDB("xmldb:xindice:///db/REAP/Agreements");
    //retrieving the right agreement based on the String id as key.
    XMLResource resource = (XMLResource) col.getResource(id.trim());
    agreement = (Document) resource.getContentAsDOM();
   }
   catch (Exception e) {
    throw e;
   }
   finally {
    if (col != null) {
     col.close();
    }
   }
   return agreement;
  } // end get_Agreement

/*****************************************************************************/

  /**
   * <p>Description : Receives an identification of the user (username) and an
   * identification of the material (collectionId and itemId) and returns true
   * if the user has an agreement or else returns false<br>
   * problem: Can throw local exeption if db not started. Not visible outside method...
   * </p>
   *
   * @param username
   * @param collectionId
   * @param itemId
   * @return boolean value telling if the resource exists in the database
   * @throws Exception Throws back exceptions from the database to the calling
   * class to be handled there
   */
```

```
 public boolean agreement_exists(String username,
                          String collectionId,
                          String itemId)
                          throws Exception
 {
  boolean exists;
  Collection col = null;
  //setting up the ID for the agreement the agreement would be stored under
  //based on the parameters
  //note the spaces inserted for easy use with commandline tool of the
  //databaseimplemention..
  String id = ("user "+username.trim()+" col "+
           collectionId.trim()+" item "+itemId.trim());
  //System.out.println("id = "+id);
  //storing agreement with string as key
  try {
    //Opening collection
    col = this.openDB("xmldb:xindice:///db/REAP/Agreements");
    //checking to see if there exists an agreement for material/user
    try{
      XMLResource resource = (XMLResource) col.getResource(id.trim());
      if (resource==null)
        exists=false;
      else
        exists=true;
      resource = null;
    }
    catch (Exception e){
      exists = false;
    }
  }
  //If database is unavailable this exception is thrown back
  catch(Exception  e){
    throw e;
  }
  finally{
    if (col != null){
      col.close();
    }
  }
  return exists;
 }

/*****************************************************************************/
  /**
   *<p>Description: Stores an agreement governing how a user may consume a material
   *               based on a received xml DOM Document.
   *
   * @param username the user that has made the agreement
   * @param collectionId The collection of the Item
   * @param itemId The number of the item within the collection
   * @param Agreement A dom Document containing the actual agreement
   * @throws Exception
   */
  public void store_Agreement(String username,
                          String collectionId,
                          String itemId,
                          Document Agreement)
                          throws Exception {
    Collection col = null;
    //setting up the ID for the agreement based on the parameters
    //note the spaces inserted for easy use with commandline tool of the
    //databaseimplemention..
    String id = ("user "+username.trim()+" col "+
             collectionId.trim()+" item "+itemId.trim());
    //storing agreement with string id as key
    try {
      //Opening collection
      col = this.openDB("xmldb:xindice:///db/REAP/Agreements");
      //storing the agreement using the string id as key.
```

```
    XMLResource resource =
     (XMLResource) col.createResource(id.trim(), XMLResource.RESOURCE_TYPE);
    resource.setContentAsDOM(Agreement);
    col.storeResource(resource);
   }
   catch (Exception e) {
    throw e;
   }
   finally {
    if (col != null) {
     col.close();
    }
   }
 } // end store_Agreement

/*****************************************************************************/

  /**
   * <p>Description: Simply removes the old Agreement from the db and stores the
   *                 new version under the same key as the old one</P>
   *
   * @param username email/username for the user who has generated the agreement
   * @param collectionId Together with itemId the identifier for the material in question
   * @param itemId Together with collectionId the identifier for the material in question
   * @param new_Agreement The updated version of the agreement
   * @throws Exception
   */
  public void update_Agreement(String username,
                               String collectionId,
                               String itemId,
                               Document new_Agreement)
                               throws Exception {
   Collection col = null;
   //setting up the ID for the agreement based on the parameters
   //note the spaces inserted for easy use with commandline tool of the
   //databaseimplemention..
   String id = ("user "+username.trim()+" col "+
            collectionId.trim()+" item "+itemId.trim());
   //replacing agreement with new agreement
   try {
    //Opening collection
    col = this.openDB("xmldb:xindice:///db/REAP/Agreements");
    //deleting the old agreement based on username as key
    col.removeResource(col.getResource(id.trim()));
    //storing the new agreement using the username as key.
    XMLResource resource =
     (XMLResource) col.createResource(id.trim(), XMLResource.RESOURCE_TYPE);
    resource.setContentAsDOM(new_Agreement);
    col.storeResource(resource);
   }
   catch (Exception e) {
    throw e;
   }
   finally {
    if (col != null) {
     col.close();
    }
   }
 } // end update_Agreement

/*****************************************************************************/

  /**
   *<p>Description: Returns an Offer for an item identified by a
   *            combination of collectionId and itemId </p>
   *
   * @param collectionId : Indicates the collection the Item is in.
   * @param itemId Indicates the Item in the collection that the offer is about
   * @return Offer A DOM Document with information about rights over the material
   *             that can be given to users and the conditions and constraints
```

```
   *              for doing so
   * @throws Exception
   */
  public Document get_Offer(String collectionId, String itemId) throws Exception{
    Document offer = null;
    Collection col = null;
    //setting up the ID for the offer based on the parameters
    String id = ("colId "+collectionId+" itemId "+itemId);
    //getting Offer with this string as key
    try {
      //Opening collection
      col = this.openDB("xmldb:xindice:///db/REAP/Offers");
      //retrieving the right offer based on the String id as key.
      XMLResource resource = (XMLResource) col.getResource(id.trim());
      offer = (Document) resource.getContentAsDOM();
    }
    catch (Exception e) {
      throw e;
    }
    finally {
      if (col != null) {
        col.close();
      }
    }
    return offer;
  }

/*****************************************************************************************************************/

  /**
   * Does : stores an ODRL Offer for a given Item/Material
   *        This method will not be implemented in this prototype. The declaration is included for consistency and to
   *        imply how this prototype can be extended to support the making of ODRL offers and publishing of material
   *        in possible future versions.
   *
   * @param collectionId : Indicating which collection the Item is in
   * @param itemId : Indicating the item in question
   * @param Offer : The Offer to be stored (Dom Document)
   * @throws Exception
   */
  public void store_Offer(String collectionId, String itemId, Document Offer) throws Exception {
    Collection col = null;
    //setting up the ID for the agreement based on the parameters
    String id = ("colId "+collectionId.trim()+" itemId "+itemId.trim());
    //storing offer with string as key
    try {
      //Opening collection
      col = this.openDB("xmldb:xindice:///db/REAP/Offers");
      //storing the agreement using the string id as key.
      XMLResource resource =
        (XMLResource) col.createResource(id.trim(), XMLResource.RESOURCE_TYPE);
      resource.setContentAsDOM(Offer);
      col.storeResource(resource);
    }
    catch (Exception e) {
      throw e;
    }
    finally {
      if (col != null) {
        col.close();
      }
    }
  } // end store_Offer

/*****************************************************************************/

  /**
   * <p>Description: This method is just taken in for clarity.
   *              It will not be used in version one of this prototype.
   *              Simply removes the old Offer from the db and stores the
```

```
   *                new version under the same key as the old one</P>
   * @param collectionId
   * @param itemId
   * @param new_Offer
   * @throws Exception
   */
  public void update_Offer(String collectionId,
                           String itemId,
                           Document new_Offer)
                           throws Exception {

    Collection col = null;
    //setting up the ID for the offer based on the parameters
    String id = (collectionId.trim()+"&"+itemId.trim());
    //replacing offer with new offer
    try {
      //Opening collection
      col = this.openDB("xmldb:xindice:///db/REAP/Offers");
      //deleting the old offer based on String id as key
      col.removeResource(col.getResource(id.trim()));
      //storing the new offer using the String id as key.
      XMLResource resource =
        (XMLResource) col.createResource(id.trim(), XMLResource.RESOURCE_TYPE);
      resource.setContentAsDOM(new_Offer);
      col.storeResource(resource);
    }
    catch (Exception e) {
      throw e;
    }
    finally {
      if (col != null) {
        col.close();
      }
    }
  } // end update_Agreement

/*****************************************************************************/

  /**
   * Does : Receives a username and a password and checks if there is a matching
   * profile stored in the database. If such a profile is found the method returns 1, else it returns
   * 0. Returned value 2 means something is wrong, (two or more users with same username and
   * password) but this should never occure.
   *
   * @return A Boolean being true if user exist and has this pwd, else returns false
   * @param username Username entered by user
   * @param pwd password entered by user
   * @throws Exception
   */
  public int checkLogin(String username, String pwd) throws Exception {
    Collection col = null;
    int result = 0;
    try {
      //Opening collection
      col = this.openDB("xmldb:xindice:///db/REAP/UserProfiles");

      //defining a query and executing it
      String xpath = "//profile[username="'+username+"' and password="'+pwd+"']";
      XPathQueryService service =
      (XPathQueryService) col.getService("XPathQueryService", "1.0");
      ResourceSet resultSet = service.query(xpath);

      //evaluating and returning result. If resultset contains one hit,
      //the username and password were correct
      int size = (int) resultSet.getSize();
      switch (size){
        case 0: result = 0; break; //user not found
        case 1: result = 1; break; //user found
        default : result = 2; //some undefined error.
      }
```

```
    }
    catch (Exception e) {
      //throws the exception back to the servlet to be handled there
      throw e;
    }
    finally {
      if (col != null) {
        col.close();
      }
    }
    return result;
  }
}
```

Appendix E: Kildekode for REAP