Agentbaserte tjenester i finanssektoren

En undersøkelse av mulighetene for integrering av informasjon fra ulike kilder, samt presentasjon av en valgt løsning for sektoren

Hovedoppgave
ved
sivilingeniørutdanning i
informasjons- og kommunikasjonsteknologi

av
Anne-Gro Olsen

Grimstad, mai 1999

Forord

Hovedoppgaven „Agentbaserte tjenester i finanssektoren" bygger på prosjektet „Multimedia Banking" innenfor forskningstema 5 „Finansiell tjenesteyting i informasjonssnettverk" i Teleøkonomiprogrammet ved SNF. Et av delprosjektene under Multimedia Banking er kalt „Kunnskapsstøttede kjøpsbeslutninger ved smarte agenter". I dette delprosjektet gjennomføres et Agentprosjekt som igjen er delt i tre forskningsområder; en kartlegging av eksisterende agentteknologi, et demoprosjekt samt en atferds- og strukturell analyse av agentbaserte tjenester. Det er det her omtalte Agentprosjektet som danner utgangspunktet for denne oppgaven.

Hovedoppgaven er skrevet som et ledd i sivilingeniørutdanningen ved Høgskolen i Agder. Arbeidet har pågått i tidsrommet januar til og med mai 1999. Jeg ønsker å takke Jan P. Nytun for hjelp med datamodellerings-metoder og UML-skjemaer, og veilederen min, Per Egil Pedersen, for konstruktiv oppfølging.

Grimstad, 28.mai 1999

Anne-Gro Olsen

Sammendrag

Denne hovedoppgaven tar for seg informasjonsintegreringsproblemet på Internett. Problemene med å utnytte informasjonen som finnes på Internett er generelle, og også finanssektoren lider under dem, både kunder og bedrifter. I oppgaven foretar jeg en kartlegging av de foreliggende forskningsresultatene som skal lette integrering og innhenting av informasjon. Resultatene er mange siden problemet er stort, og mange har dedikert seg til denne forskningen. Av alle de mulighetene jeg her finner, er det derimot få som tas i bruk i dagens finanssektor på Internett. Den neste undersøkelsen jeg foretar i oppgaven viser nemlig at de fleste tilbydere av integrert finansinformasjon benytter seg av „tradisjonelle“ metoder under informasjons-innsamlingen som fax, telefon og e-mail.

Etter en vurdering av forskningsresultatene, har jeg valgt *fellesformat* som den løsningen jeg syns egner seg best for finanssektoren. Fellesformatet løser grunnleggende vanskeligheter ved å strukturere informasjonen på kildesidene. Bankene opererer i dag ikke med noen standard for hva som publiseres og hvordan dette skal gjøres. Jeg utvikler derfor et forslag til et entydig informasjonsinnhold for banker og finansinstitusjoner. Dette informasjons-innholdet tilrettelegger jeg så for det valgte fellesformatet, og viser hvordan man kan utvikle verdiøkende tjenester for finanssektoren med et strukturert informasjonsinnhold som grunnlag.

Eksempler på de konsekvensene jeg ser av en omlegging til et strukturert informasjonsformat er blant annet at man på produsent-siden lettere kan gå bort fra den integrerte verdikjeden som har vært vanlig i bank, til mer outsourcing av tjenester. Videre vil det støtte framveksten av nye mellomledd på Internett ettersom innsamling og integrering av informasjon vil forenkles, og til slutt vil kundene enklere kunne utveksle erfaringer, for eksempel i kundegrupper på Internett, ved at informasjonen de oppgir er entydig strukturert.

INNHOLD

TABELLER

FIGURER

1 Innledning

Det å selge finansprodukter på Internett er blitt et attraktivt marked, ikke bare for bankene, men også for andre aktører. Kundegruppen består av høyt utdannede, godt lønnede, yngre personer som har en høyere etterspørsel av finansielle tjenester enn den gjennomsnittlige bankkunde-gruppen. I dette markedsrommet finner man stor framvekst av tjenesteintegrerende systemer og tilbydere av enkeltstående finansvarer som for eksempel spesialiserte lån eller fond. Det blir stadig viktigere for finansinstitusjoner på Internett å tilby kundene verdiøkende tjenester, å følge med i den teknologiske utviklingen og å sammarbeide med for eksempel tidligere konkurrenter eller programvaretilbydere for å kunne sikre markedsandeler. Kunden forventer i større grad enn før et tilpasset finansielt tilbud og er mindre bundet av tradisjonelle lojalitetsbånd når handelen foregår på Internett. Fra kundens side er det viktig å finne fram til det beste tilbudet, selv om dette medfører mye leting.

Hensikten med Agentprosjektet i Multimedia Banking var, ifølge forstudien til prosjektet, å utvikle kunnskapsbaserte *søkeagenter* for kjøp av banktjenester på Internett. Dette skulle medføre en aktivisering av kunden med hensyn til utforming og skreddersying av finansprodukter, noe som igjen skulle reise spørsmål som hvordan man kan lage regler som forutsier kundens preferanser og valg, og hvordan kjøpsatferden endres ved bruk av søkeagenter [2].

Det finnes flere modeller som sier noe om hvilke mønstre kunder tradisjonelt følger i en kjøpssituasjon. I [1] beskrives CBB-modellen (Consumer Bying Behavior) som innbefatter hendelser og avgjørelser i forbindelse med handel og valg av varer og tjenester. Både CBB-modellen og andre liknende teorier for kundeoppførsel opererer med seks ulike trinn som leder kunden gjennom kjøpsprosessen. Disse trinnene strekker seg fra *behovsidentifisering*, altså fra kunden blir oppmerksom på et udekket behov, og til *service og evaluering*, når kunden evaluerer kjøpet og avgjørelsen. Internettagenter er spesielt egnet til å simulere de trinnene som innebærer informasjonsinnhenting og –filtrering, individuelle evalueringer, komplekse overveielser og tidsbaserte interaksjoner, det vil si trinnene *produktmekling*, *salgsmekling* og *forhandling* i CBB modellen.

Behovet for et slikt verktøy som er til hjelp under informasjonsinnhentingen, og som integrerer informasjonen for brukeren, skyldes det enorme, ustrukturerte informasjonstilbudet som finnes på Internett. Siden Internett er vokst opp uten noen indeks eller krav til definisjon av innhold, opplever man som bruker at det er svært vanskelig å finne fram til nøyaktig den informasjonen man leter etter. Kildene er mange og innholdet mangler en definert struktur. Det er både tidkrevende og noen ganger umulig å finne fram. Også agentene må tilpasse seg de allerede eksisterende, teknologiske forholdene på Internett, ettersom de er underlagt de samme begrensninger og manglende standarder som en vanlig bruker opplever. Agentene skal lette informasjonsinnhentingen og –integreringen for en vanlig bruker, samtidig møter agentene de samme problemene, om enn på en automatisert måte, i forsøket på å integrere informasjon fra ulike kilder. Problemene forbundet med identifisering og integrering av informasjon på Internett kalles med en fellesbetegnelse for *informasjonsintegreringsproblemet*.

Informasjonsintegreringsproblemet er et stort forskningsområde, med mange implementerte anvendelser som søker å løse problemet på ulike måter. Intelligente agenter er kun én slik anvendelse. Jeg ønsker i denne oppgaven å studere hele informasjonsintegreringsproblemet, og deretter vurdere alternative anvendelsesområder, før jeg velger den løsningen jeg finner egner seg best for å løse problemene i finanssektoren.

Jeg har tatt utgangspunkt i eksisterende litteratur omkring temaet for å skaffe meg en oversikt over dagens situasjon, noe som inkluderer både artikler og whitepapers publisert på Internett, bøker og prosjektbeskrivelser. I kapittel 2 presenterer jeg problemstillingen. Denne skal vise seg å være tredelt. I kapittel 3 redegjør jeg for de metodene jeg har benyttet gjennom arbeidet. Det har vist seg vanskelig å velge en enhetlig metode for framstillingen. Grunnen er at jeg både foretar *utredning* av problemstillingen og *utvikling* av løsning. Jeg har derfor valgt å dele metodedelen i tre slik at den følger inndelingen av problemstillingen. Hver problemdel behandles så for seg i kapittel 4, med behandling av teori, presentasjon av ulike løsninger på de gitte problemstillingene, vurdering av løsningene og beskrivelse av valgte alternativer. I kapittel 5 presenterer jeg de konklusjonene jeg har kommet til, samt vurderer gyldigheten deres. Til slutt ser jeg på hvilke konsekvenser konklusjonene mine kan få for finanssektoren.

2 Problembeskrivelse

Det generelle problemet som ligger til grunn, og som skaper vansker for integrering og utnyttelse av informasjon på Internett, betegnes altså som informasjons-integreringsproblemet, og bunner i mangelen på struktur og oversikt på Internett. Ettersom alle, det være seg bedrift eller kunde, som ønsker å utveksle eller utnytte Internett-informasjon, møter vanskeligheter, drives det naturlig nok mye forskning på dette temaet. Dermed gjelder det altså først å få et overblikk over forskningsområdet, og deretter, blant de tilgjengelige forskningsresultatene, å finne fram til en løsning som både kunde og bedrift kan ha nytte av, og som egner seg i den aktuelle sektoren.

En tradisjonell tanke innen databehandling har vært å standardisere og strukturere data slik at informasjon kan utveksles og utnyttes mellom brukere og systemer. For banker og finansinstitusjoner på Internett er informasjon hovedproduktet. Det er viktig at kundene får tak i informasjonen, og det er viktig å kunne utveksle informasjon med leverandører og partnere. Slik bildet er i dag har kunder og bedrifter ikke et entydig referansegrunnlag, og dette kompliserer situasjonen.

I tillegg til å gjøre informasjon tilgjengelig er det viktig for bedrifter å tilby kundene verdiøkende tjenester for å styrke relasjonene i mangelen av de tradisjonelle lojalitetsbåndene. Men mange av de applikasjoner og anvendelser som man allerede finner, fungerer dårlig på grunn av manglende struktur og standarder.

I de følgende delkapitlene ser jeg nærmere på de tre problemene; hvordan integrere Internett-informasjon, hvordan presentere informasjon på fellesformat og hvordan utnytte den strukturerte informasjonen i nye eller forbedrede tjenester.

2.1 Informasjonsintegreringsproblemet på Internett

Jeg ser for meg at selve informasjonsintegreringsproblemet på Internett er todelt; For det første finnes problemet som er knyttet til det å integrere informasjon fra ulike kildetyper innen samme bedrift for presentasjon på Internett. Et eksempel kan være skriftlig informasjon samlet i dokumenter og data gruppert i databaser som skal presenteres på samme web-område. For det andre finnes problemet som er knyttet til innhenting og integrering av informasjon fra ulike web-kilder på Internett. Informasjonen, som vanligvis finnes på HTML-format, inneholder lite

meta-data eller informasjon om informasjonen. Dette gjør det vanskelig å søke etter og behandle informasjon.

I hovedoppgaven konsentrerer jeg meg hovedsaklig om del to av informasjons-integreringsproblemet. I den forbindelse beskriver jeg først og fremst ulike underliggende teorier det i øyeblikket forskes rundt for å hente inn og behandle informasjon. Dernest ser jeg på ulike anvendelser av de teoretiske løsningene. Anvendelsene varierer fra applikasjoner som behandler en spesifikk del av integreringsproblemet, til systmer som betrakter hele problemområdet under ett og gir en enhetlig løsning.

Fra bankenes og finansinstitusjonenes synspunkt er det viktig at informasjonen som presenteres når ut til kundene. Gitt de ovennevnte problemene en kunde møter med å finne informasjon på Internett, bør informasjonstilbyderne i større grad konsentrere seg om å finne kundene. For å oppnå dette er det nødvendig at ulike aktører, enten de er tidligere konkurrenter i finanssektoren eller opererer i forskjellige bransjer. koordinerer seg, og samarbeider om å legge forholdene til rette, for på denne måten å vinne kundenes tillit og interesse. En måte å opprette et slikt samarbeid på, er å utarbeide et felles presentasjonsformat for finansinformasjon på Internett.

2.2 Hvordan presentere informasjon på fellesformat

Mange banker og finansinstitusjoner er i dag etablert på Internett med egne websider hvor man kan hente opplysninger. De fleste bankene oppgir, mer eller mindre utfyllende, den samme typen informasjonen. Slik situasjonen er i dag publiserer bankene sin egen informasjon uavhengig av de andre, det finnes ingen standard for hva og hvordan man skal oppgi dataene.

Dette byr ikke på problemer så lenge kunden bare er ute etter manuelt å ”surfe” gjennom bankens side og samle opplysninger. I dette tilfellet er ikke brukeren interessert i den underliggende datastrukturen, man er kun interessert i å finne de aktuelle opplysningene så enkelt som mulig. Dersom en sluttbruker derimot ønsker å foreta en mer grundig gjennomgang av dataene, for eksempel en sammenligning av tilbudene til flere banker, oppstår det problemer med dagens situasjon. Man kan tenke seg automatiserte prosedyrer eller agenter som har fått som oppgave å oppsøke et gitt antall banker og finansinstitusjoner for å sanke inn informasjon om tilbudene som gjelder lånerentene. Dette gjøres i dag ved at man ”parser” gjennom HTML sider med informasjon og enten finner fram til enkelte

ord i teksten som passer med spørringen, eller man går til bestemte posisjoner i HTML-dokumentet og trekker ut informasjonen som er lagret der. Det er klart at denne prosessen er vanskelig å vedlikeholde, ettersom det er tilstrekkelig at én av bankene endrer sin layout på websidene for at prosedyren må skrives om.

Det er derfor ønskelig at banker og finansinstitusjoner på Internett opererer med en entydig datastruktur over informasjonstilbudet. Dersom dette lar seg gjøre, kan automatiserte prosedyrer enkelt, og på en standardisert måte, finne fram til den aktuelle informasjonen. Denne tankegangen fører lett videre både til ideer om nye anvendelser og til bedre utnyttelse av de allerede eksisterende applikasjonene som finnes på Internett i dag.

2.3 Tjenesteutnyttelse av informasjonsstrukturen

Den store frykten for mange banker og finansinstitusjoner er i dag å bli hengende etter i den teknologiske utviklingen. Denne industrien er i rask endring, og man risikerer å miste sjansen til å komme med i markedet dersom man ikke følger med fra begynnelsen. Aktørene i finansmarkedet må sikre seg kunder, og den eneste måten de kan gjøre det på er å tilby kundene stadig mer verdiøkende tjenester. Verdiøkende tjenester kan være alt fra nye tjenester og kryss-salg, altså salg av relaterte finansprodukter og –tjenester, til det som kalles „service aggregation“ [28]. Med „service aggregation“ menes det å samle et stort antall tjenester og produkter på ett enkelt nettsted. Eksempler på verdiøkende tjenester er mulighetene til å tilby kundene rentevarsling eller nettverkskalkulatorer, å presentere sidene på en personliggjort måte eller å foreta informasjonsinnhenting på vegne av kundene for å finne det beste tilbudet.

På Internett finnes allerede en rekke applikasjoner og tjenester som lider under manglene som uteblivende standarder og strukturer utgjør, og blant dem er mange applikasjoner i finanssektoren. Med en entydig datastruktur for det informasjons-innholdet som banker og finansinstitusjoner presenterer, vil mange av disse applikasjonen kunne forbedres både sett fra brukerens og fra bankenes synspunkt. Man vil lettere kunne både samle inn og sammenlikne data, og mellom banker og finansinstitusjoner vil også overføring av informasjon kunne gjøres på en standardisert måte.

3 Metodedel

Under arbeidet med hovedoppgaven har jeg, som en stor del av arbeidet, gjennomført innsamling av informasjon. Det finnes et bredt antall måter man kan samle informasjon på, og jeg har i hver del av oppgaven valgt den metoden jeg har funnet mest hensiktsmessig. I dette kapitlet beskriver jeg hvordan jeg har gått fram for å hente inn informasjon som danner grunnlag for konklusjoner innen de ulike problemstillingene.

Kapitlet er delt i tre hoveddeler, og følger således probleminndelingen i oppgaven. Det første delkapitlet beskriver undersøkelsene jeg har gjennomført i forbindelse med teori og anvendelser rundt informasjonsintegreringsproblemet. Som et underkapittel gjennomgår jeg her hvordan jeg har gjennomført en kartlegging av eksisterende teknologibruk i finanssektoren. I det andre delkapitlet gjennomgår jeg valg av metode i forbindelse med presentasjon av data på et felles format. Denne delen er videre delt inn etter metode for valg av datamodelleringsmetode, metode for valg av fellesformat og metode ved tilrettelegging av informasjon på valgt fellesformat. I det siste underkapitlet gjennomgår jeg så hvordan jeg har gått fram under undersøkelsene i forbindelse med utnyttelser av det strukturerte informasjonsinnholdet.

I de kapitlene hvor jeg beskriver *litteraturstudiene* har jeg fulgt inndelingen *design*, *utvalg* og *måling*. I de kapitlene hvor undersøkelsesmetoden er *design-demonstration* er inndelingen hovedsaklig delt mellom *hva* jeg har laget og *hvilke hjelpemidler* jeg har benyttet meg av under implementeringen.

3.1 Metode ved kartlegging av teknologi for integrering av informasjon

Som første praktiske del av hovedoppgaven gjennomførte jeg en kartlegging av den tilgjengelige teknologien for innsamling og integrering av informasjon på Internett. Formålet med denne undersøkelsen var å beskrive de mulighetene man har, per i dag, innen finanssektoren når det gjelder denne typen informasjons-behandling på Internett, og jeg har da tatt et generelt utgangspunkt i eksisterende teknologi. I tillegg til å beskrive de teknologiske mulighetene man har til å utføre integrering av informasjon, var jeg også interessert i å finne ut hvordan teknologien faktisk ble utnyttet i den aktuelle sektoren. Metode og framgangsmåte ved denne undersøkelsen er beskrevet i kapittel 3.1.1.

Når det gjelder første del av undersøkelsen – kartlegging av tilgjengelig teknologi – var spørsmålet denne undersøkelsen skulle gi svar på altså:

- *Hvilke muligheter finnes for å lette innhenting og integrering av informasjon for brukeren?*

For å belyse denne problemstillingen har jeg gjennomført en sammenligning av problemløsninger jeg har funnet fram til ved *litteraturstudier*. Man skiller i litteraturstudier mellom *søke*-prosessen og *review*-prosessen. Under søkeprosessen identifiserte jeg derfor de kildene jeg ønsket å benytte. Kravet til kildene var at de skulle gi meg en generell oversikt over hvordan virkeligheten er, både med hensyn på teknologi og teori innenfor problemområdet. Her har Internett vært den største informasjonskilden, med en mengde tilgjengelige data på hva som er aktuelle teknologiske problemstillinger innen temaet.

Neste fase i søkeprosessen er da å snevre inn søkeområdet for å velge ut de mest interesante problemstillingene. Her har jeg derfor valgt å konsentrere meg om aktuelle forskningsområder. Ulempen med dette valget er at litteraturstudien min ikke gir et historisk perspektiv på utviklingen for teknologien på problemområdet. Fordelen er derimot at jeg kan konsentrere meg om det som kan være aktuelle løsninger på dagens problemer i finanssektoren.

I tråd med review-prosessen i litteraturstudiet, har jeg så evaluert og presentert de viktigste forskningsområdene det for øyeblikket arbeides med. Disse forskningsområdene har jeg kategorisert etter hvor de hører hjemme i inndelingen av informasjonsintegreringsproblemet. Ved å foreta en sammenligning av løsningene har jeg kommet fram til det alternativet jeg regner som best egnet for å løse problemene innen finanssektoren.

Ettersom den underliggende teknologien på Internett er felles for de fleste sektorer, regner jeg at kravet til realisme dekkes ved å studere problemstillingen generelt. Det vil si at jeg mener konklusjonene jeg trekker kun for finanssektoren er realistiske, selv om de baserer seg på generell teknologi for internettet.

Som utvalg for denne undersøkelsen har jeg som nevnt begrenset meg til informasjon om områder som er aktuelle forskningsområder, og som derfor muligens har en framtid på Internett. Jeg har derfor ikke tatt i betraktning, eller gått nærmere inn på, tidligere teknologi eller teknologi som historisk sett legger grunnlaget for der vi er i dag. Denne begrensningen i utvalg føler jeg er nødvendig for å kunne basere konklusjonene mine på oppdatert informasjon. Tidsrammen for

oppgaven har i tillegg satt grenser for omfanget på litteraturstudiet i denne, som i andre deler av oppgaven.

De spørsmålene jeg har stilt meg selv, og som har ført fram til det utvalget jeg presenterer i rapporten, er:

- Kan man skille mellom hva som gjøres på serversiden og hva som gjøres på klientsiden når det gjelder forskningsområdene. Det vil si, er det slik at noen konsentrerer seg om å løse problemet kun fra én side?
- Er det en mer naturlig inndeling å anta at løsningene retter seg mot en spesifikk del av informasjonsintegreringsproblemet, som for eksempel identifisering av kilder, ekstrahering av informasjon eller konvertering av informasjonen til et foretrukket format?
- På Internett benyttet jeg søkeord som ”Information Retrieval”, ”Information Exctraction”, ”Wrappers”, ”Spiders”, ”Agent technology” osv. for å finne informasjon om problemområdet. Gjennomlesing av en del publikasjoner satte meg så på nye leteord, som for eksempel ”virtual libraries”, ”social filtering” og ”multiple searches”.
- Finnes det systemer som i dag, på en enkel måte, foretar integrering av informasjon fra ulike kilder på Internett, og hvordan opererer de i så tilfelle?

3.1.1 Delundersøkelse: Dagens teknologibruk i finanssektoren

Som et ledd i kartleggingen av teknologi for integrering av informasjon, har jeg foretatt en delundersøkelse for å kartlegge teknologibruken i finanssektoren, og spørsmålet som ligger til grunn for valg av metode for denne undersøkelsen er:

- *Hvordan foretas informasjonsintegreringen ved de nettstedene som allerede presenterer finansinformasjon på Internett?*

Denne typen opplysninger er det ikke vanlig å publisere for offentlig tilgjengelighet på Internett, eller i andre medier. Jeg hadde derfor ingen mulig til å få tak i slik intern informasjon uten å ta direkte kontakt med firmaer som har sin virksomhet i den aktuelle sektoren.

For å få svar på spørsmålet gjennomførte jeg derfor en *survey-undersøkelse*. Jeg valgte å benytte en mail-survey – utsending av enkle, skriftlige spørsmål via elektronisk post – som framgangsmåte. Fordelene med en slik

survey-undersøkelse er at den er kostnadseffektiv og jeg kan selv spesifisere de enkelte dataene jeg ønsker svar på ut fra problemstillingen. Ulempen er at det er en forholdsvis krevende måte å hente inn informasjon på, noe undersøkelsen min viser: Størsteparten av dem jeg oppsøkte valgte å ignorere henvendelsen min; noen hadde ikke teknologisk kompetanse til å svare, og andre igjen valgte å ikke svare ettersom de regnet informasjonen som sensitiv. En slik undersøkelse kan uansett gi svar på flere forhold jeg ikke hadde mulighet til å oppdage kun ved å oppsøke nettsteder eller foreta litteraturstudier.

Ved å lete etter spesifikke søkeord på Internett (f.eks ”loan rates” eller ”mortgage rates”) fant jeg fram til en rekke, først og fremst amerikanske, nettsteder hvor man presenterer integrert informasjon fra låntilbydere og finansinstitusjoner. Selv om populasjonen i undersøkelsen er nettsteder i USA, mens jeg ellers i oppgaven tar utgangspunkt i norske forhold, regner jeg dette for et representativt gyldighetsområde ettersom undersøkelsen kun tar i betraktning teknologien som benyttes under innsamlingen, og ikke faktorer som demografiske eller kulturelle forhold. Jeg går ut fra at teknologibruken i forbindelse med Internett er tilnærmet den samme i Norge og i USA. Grunnen til valget av populasjon er at det foreløpig ikke finnes tilsvarende mange norske nettsteder som foretar integrering av finansinformasjon på Internett.

De konkrete spørsmålene jeg stilte for å få svar som kunne hjelpe meg å danne et bilde av teknologibruken i sektoren var:

- Har dere en Internett-side hvor dere beskriver hvilken underliggende teknologi som benyttes i informasjonsinnsamlingen fra banker og finansinstitusjoner?
- Hvis ikke, kan dere med få ord beskrive denne teknologien i et svar på mailen fra meg?

Disse spørsmålene var ikke formulert på en måte som egnet seg for ja/nei-svar, det vil si de var ustrukturerte og krevde derfor en litt større innsats fra intervju-objektene enn et vanlig spørreskjema ville gjøre. Svarene jeg fikk varierte stort i informasjonsinnhold. I del 4.1.2 gir jeg en detaljert beskrivelse av resultatene fra undersøkelsen.

3.2 Metode for presentasjon av data på fellesformat

Mange banker og finansinstitusjoner presenterer sitt informasjonsinnhold på Internett. Det finnes derimot ingen standard for hvordan dette gjøres, eller en mal for hva presentasjonen skal inneholde. Jeg har derfor gjennomført en systematisering av banker og finansinstitusjoners informasjonsinnhold med den hensikten å finne en *entydig datastruktur* som kan gjelde alle banker og finansinstitusjoner.

Denne delen av oppgaven består av tre underkapitler: Presentasjon av data på fellesformat innebærer først og fremst valg av metode for datamodellering som del 1, valg av fellesformat for datastrukturen som del 2, og som siste del en presentasjon av et konkret forslag til fellesformat for informasjonsinnholdet til banker og finansinstitusjoner på Internett.

3.2.1 Valg av metode for datamodelleringsproblemet

Undersøkelsen skal gi svar på spørsmålet:

- *Hvilke mulige veier kan man velge for å modellere et systems' informasjons-innhold?*

Ettersom formålet her er å få en oversikt over hva som finnes av kunnskap og muligheter på dette området, har jeg benyttet meg av *litteraturstudier* innen temaet *modellering av informasjon*.

I søkeprosessen utførte jeg spørringer på Internett for å identifisere publisert informasjon om datamodelleringsmetoder. Jeg snevret inn temaet etterhvert til de store metodene som CRC, OMT og UML. Metodene er alle anerkjente og mye brukte datamodelleringsmetoder, selv om de skiller seg fra hverandre på enkelte punkt og derfor egner seg godt innen ulike system-situasjoner. Disse punktene var det viktig for meg å kartlegge nøye, ettersom jeg var ute etter en metode eller et verktøy som kunne hjelpe meg under strukturering av et system av en viss type. En grundig analyse av de ulike metodene under review-prosessen av litteraturstudiet, førte derfor fram til visse hovedtrekk ved hver av disse metodene. Hovedtrekkene vurderte jeg deretter opp mot hverandre og la på denne måten grunnlaget for å kunne foreta et valg av datamodelleringsmetode.

Utvalget jeg har vurdert har bestått av noen av de største og mest brukte datamodelleringsmetodene. Jeg begrenset utvalget ytterligere til de metodene jeg

fant var best dokumentert på de feltene jeg hadde interesse for. De ulike datamodelleringsmetodene jeg har vurdert er beskrevet under punkt 4.2.1.

Ettersom utgangspunktet for å foreta et valg av datamodelleringsmetode var behovet for å strukturere bankers- og finansinstitusjoners informasjonsinnhold på Internett, forsøkte jeg først og fremst å skaffe meg informasjon om *strukturering av systemers informasjonsinnhold*, *systemutvikling*, og *datamodellering*. På Internett fant jeg flere artikler og nettsider som forklarte de ulike framgangsmåtene.

Som en følge av valget av datamodelleringsmetode, kompletterte jeg denne delen av oppgaven med å benytte den valgte datamodelleringsmetoden til å utføre en strukturering av informasjonsinnholdet til banker og finansinstitusjoner på Internett. Som utgangspunkt tok jeg informasjonen som presenteres på web-sidene til Norsk Familieøkonomi, og kompletterte bildet med informasjon fra nettsidene til DnB og Gjensidige.

Jeg benyttet programmet Visio Professional 5.0 for Microsoft Windows under framstillingen av strukturskjemaet. Programmet fungerer som et verktøy for å visualisere og strukturere systemer. Visio har et eget program som støtter utformingen av UML klasse-diagrammer, og dette benyttet jeg meg av under utviklingen.

3.2.2 Metode ved valg av fellesformat for informasjons-innholdet

Etter å ha valgt datamodelleringsmetode for informasjonsinnholdet til banker og finansinstitusjoner, måtte jeg nå velge et presentasjonsformat for den strukturerte informasjonen. Spørsmålet jeg nå ønsket svar på var:

- *Hvilke ulike presentasjonsformater kan jeg velge mellom, og hvordan egner de seg for domenet og problemstillingen?*

Framgangsmåten jeg valgte å benytte meg av for å finne svar på ovennevnte spørsmål var også her *litteraturstudier*. Under søkeprosessen identifiserte jeg derfor kilder på Internett som sa noe om ulike presentasjonsformater, i tillegg til bøker og ”online” brukerveiledninger.

Ettersom det av den informasjonen jeg studerte tydelig framgikk at et av presentasjonsformatene var det best egnede alternativet og i realiteten det eneste aktuelle for sektoren, valgte jeg å legge hovedvekten av konsentrasjonen min rundt det å kartlegge muligheter og begrensninger knyttet til dette

presentasjonsformatet. Fordelen med denne stramme innsnevringen er uten tvil at jeg har hatt muligheten til å sette meg relativt godt inn i det valgte presentasjonsformatet. Dette medfører jo automatisk ulempen med en tilsvarende overfladisk kjennskap til de andre, nevnte formatene. Begrunnelsen for dette valget er utdypet nærmere i del 4.2.4.

Under review-prosessen av litteraturstudiet har jeg konsentrert meg om å trekke fram hovedpunkter ved det valgte presentasjonsformatet, og begrunne hvorfor formatet bør benyttes for å forenkle informasjonsintegreringen i finanssektoren.

Det finnes mye tilgjengelig publisert informasjon om det valgte fellesformatet. Ettersom det er en relativt ny og omdiskutert standard som flere store organisasjoner og programvareutviklere har valgt å støtte, er det mange som har villet si sin mening om temaet. Problemet har heller vært å velge ut den informasjonen som kan betegnes som gyldig. Både Microsoft og IBM har vært blant pådriverne, og jeg har hatt stor nytte av deres informasjonssider og opplæringssider. Her finnes i tillegg oppdatert informasjon om endringer i standarden, forslag til forbedringer og brukerveiledninger.

De faktorene jeg har valgt å sette fokus på i undersøkelsen har vært å:

- sammenligne presentasjonsformatet med gjeldende format for publisering av informasjon på Internett (HTML)
- Undersøke hvilke applikasjonstyper som vil være pådrivere for denne typen fellesformat, og se på hvordan dette kan relateres til finanssektoren.
- Se på konsekvensene valget av fellesformat vil få for den aktuelle sektoren

3.2.3 Metode ved tilrettelegging av informasjon på valgt fellesformat

Jeg er nå på et stadium i oppgaven hvor jeg har valgt en løsning på problemene forbundet med integrering av informasjon i finanssektoren, jeg har valgt metode for modellering av informasjonsinnholdet i sektoren og jeg har valgt presentasjonsformat for den aktuelle løsningen. Det jeg så ønsker å gjøre er å utvikle et konkret eksempel som viser hvordan finansinformasjonen kan tilrettelegges i det valgte fellesformatet. Det konkrete eksemplet er et XML-dokument basert på det allerede strukturerte informasjonsinnholdet. XML-

dokumentet skal støtte seg på en fil (Document Type Definition) som spesifiserer strukturen i dokumentet ut fra datastrukturen jeg fant fram til under del 4.2.3. Til denne utviklingsoppgaven velger jeg å benytte meg av en *design-demonstration* metode. Metoden går i korte trekk ut på å konstruere, teste og evaluere et program. I mitt tilfelle er programmet en XML-side med underliggende DTD.

Jeg ble, gjennom sammenligning av flere ulike kilder for informasjon om XML, imidlertid klar over at det finnes ulike muligheter, både når det gjelder typen definisjonsfil og det man kan kalle visualiseringsspråk. Man må velge mellom tradisjonell DTD og Microsofts' nye forslag til utvidbar DTD kalt XML-Schema for definisjonsfila, og mellom CSS og XSL for visualiseringsspråket. Jeg gjennomførte derfor sammenligninger av mulighetene og foretok valg basert på konklusjonene jeg trakk av analysene. Spesifikasjonsfila utviklet jeg så med utgangspunkt i det strukturerte informasjonsinnholdet fra del 4.2.3. Visualiseringsspråket benytter jeg i neste del, ved utvikling av tjenesteeksempler.

Både XML-dokument og DTD er rene tekst-filer, og jeg har benyttet Notepad ved skrivingen av dem. De siste versjonene av Internett-browserene Explorer og Netscape kan „parse" XML-dokumenter og validere dem mot en DTD, dersom dette er angitt i XML-fila. I del 4.2.5 presenterer jeg resultatene av utviklingsprosessen for DTD-fila, og i 4.2.6 beskriver jeg utviklingen av XML-dokumentet.

3.3 Metode for undersøkelser rundt ulike utnyttelser av bankenes informasjonsinnhold

Problemstillingen i denne siste delen av oppgaven har vært å definere hvilke applikasjonsområder på Internett som benytter seg av informasjonstilbudet til banker og finansinstitusjoner, ettersom en standardisering av strukturen på informasjons-innholdet vil ha betydning for alle applikasjoner som benytter seg av dette som underliggende informasjonstilbud. Hensikten med å foreta undersøkelsen er kanskje først og fremst å begrunne nødvendigheten av en entydig datastruktur på informasjonsinnholdet til banker og finansinstitusjoner. Dersom jeg kan identifisere forbedringsmuligheter og ideer til nye funksjoner, vil temaet sannsynligvis oppfattes som mer relevant.

Spørsmålene denne undersøkelsen skal gi svar på er:

- *Hvilke applikasjonstyper benytter seg av informasjonsinnholdet til banker og finansinstitusjoner?*

- *Hvilke nye muligheter har man med en standardisert informasjonsstruktur?*

En mulig framgangsmåte for å belyse disse problemstillingene ville her være å oppsøke et gitt antall nettsteder som enten driver med integrering og presentering av informasjon i finanssektoren, eller bankers egne nettsteder. På disse nettstedene kunne jeg så finne detaljert informasjon om hvilke tilbud de har per i dag, og eventuelt lete etter funksjoner som kan forbedres eller nye ideer som kan utvikles. Dette ville jeg valgt å utføre ved hjelp av en *case-studie*.

Jeg har imidlertid valgt å løse denne delen av oppgaven med å presentere et utvalg demonstrasjonseksempler som jeg ønsker skal belyse hvordan jeg tenker meg at det valgte presentasjonsformatet kan være med på å løse, eller i det minste forenkle, informasjonsintegreringsproblemet i finanssektoren. I siste del av rapporten, hvor jeg beskriver konklusjoner og konsekvenser, ser jeg nærmere på hvilke muligheter jeg ser med det valgte formatet som løsning for sektoren.

Jeg beskriver i den første delen av dette kapitlet tre eksempler på tjenester som utnytter bankenes og finansinstitusjonenes informasjonstilbud, og hvordan jeg ser for meg at XML kan gjøre disse typene informasjonsbehandling enklere. Under del to presenterer jeg så tre eksempler på anvendelser med DTD, XML og XSL. Under utviklingen av disse eksemplene har jeg fulgt *design-demonstration* metoden. I dette tilfellet ligger konstruksjonen i utviklingen av XSL stylesheets som skal benyttes for å presentere informasjonen i XML-dokumentet på forskjellige måter. Som utgangspunkt for alle eksemplene har jeg benyttet Microsoft‘s demonstrasjoner av anvendelser som ligger tilgjengelige for nedlastning på deres informasjonssider om XML.

- Det første eksemplet tar for seg sortering av informasjon ved å benytte ulike XSL-filer som hver sorterer innholdet etter et gitt kriterie.
- Eksempel to viser en mer effektiv variant av sorteringseksemplet. Her benyttes kun ei XSL-fil som inneholder ei prosedyre som kalles opp med den kolonnen man ønsker å sortere etter som parameter.
- Med det tredje eksemplet ønsket jeg å vise *pattern-matching* mulighetene med XML. Her filtreres informasjonen på bakgrunn av valgte parametre, og eksemplet viser hvordan man enkelt kan lokalisere eller ekskludere informasjon lagret i samme fil, bare ved å definere filter-parametre.

Verktøyet ved utviklingen av disse eksemplene var også her, som tidligere, Notepad og Internet Explorer 5.0.

Når det gjelder testing av resultatet som er neste steg i *design-demonstration* metoden, har jeg foretatt såkalt „debugging“ av eksemplene helt til de fungerte tilfredsstillende. Med hensyn til evalueringen ser jeg jo at eksemplene gjør det de forventet å skulle gjøre – de forenkler prosesser som ellers ville ha involvert serversiden i en internettforbindelse og de muliggjør nye presentasjonsformer av data som er samlet i en enkel underliggende datafil.

Eksemplene demonstrerer enkelte sider av problemet hvor fellesformat-løsningen egner seg godt. Jeg har derimot ikke kunnet utvikle omfattende demonstrasjoner som belyser de mer vidtrekkende konsekvensene av løsningen på grunn av tidsmangel. Disse løsningene prøver jeg likevel å beskrive med ord, og jeg tror det kommer godt fram i diskusjonen til slutt i kapitlet hvordan jeg anser at løsningen kan gripe om seg i finanssektoren. De konkrete resultatene av denne undersøkelsen kan sees under punkt 4.3.3.1 a), b) og c).

4 Behandling av delproblemer og resultater av undersøkelsene

I dette kapittelet presenterer jeg de resultatene jeg har kommet fram til under gjennomføringen av undersøkelsene som er beskrevet i forrige kapittel. Resultatene er organisert i tre deler etter inndelingen av problemområdet: I del 4.1 (Informasjonsintegreringsproblemet) gir jeg først en teoretisk gjennomgang av problemområdet, samt en beskrivelse av ulike anvendelser. Deretter foretar jeg en kartlegging av hvordan problemet løses i praksis i sektoren, og til slutt et valg av løsning for domenet. I del 4.2 (Fellesformatet) beskriver jeg ulike framgangsmåter for modellering av informasjonsinnhold, foretar et valg av modelleringsmetode og presenterer det resultatet jeg kommer fram til for finansinformasjonen. Deretter velger jeg fellesformat for sektoren og beskriver utviklingen av DTD og XML-dokument. I del 4.3 (Tjenesteutnytting) presenterer jeg tre eksempler på ulike tjenester i finanssektoren og vurderer så hvordan disse tjenestene kan dra nytte av det nye presentasjons-formatet. Deretter beskriver jeg utviklingen av tre demonstrasjons-eksempler på bruk av XML/XSL.

4.1 Informasjonsintegreringsproblemet

World Wide Web gir oss muligheten til å kommunisere med hele verden. Men for fullt å kunne utnytte mulighetene som web gir oss, er det viktig å benytte veldefinerte standarder. Foreløpig representerer HTML, HTTP, GIF og JScript noen slike standarder som gjør det mulig at en side opprettes et sted og så kan leses av mange forskjellige brukere til ulike tider. Dette standard-laget er imidlertid ikke nok til at data kan representeres og håndteres på en tilfredsstillende måte. Slik Internett framstår i dag er det mer et tilgangs-medium til tekst og bilder. Det finnes ingen standarder for intelligente søk, dataoverføring, tilpasset presentasjon og individuell tilpasning. I tillegg finnes det ingen entydig indeks over alle kilder som er tilknyttet nettet, ingen standard programmeringsgrensesnitt for å utføre spørringer mot kilden og svarene på spørringene er heller ikke strukturerte.

Dette medfører at søking etter informasjon på Internett byr på store utfordringer. Det er disse vanskelighetene, som direkte fører til en svært dårlig utnyttelse av informasjonen som finnes på Internett, som karakteriseres som informasjonsintegreringsproblemet.

4.1.1 Informasjonsintegreringsproblemets tre deler

Informasjonsintegreringsproblemet kan splittes i minst tre underliggende delproblemer [22]; *informasjonsidentifisering*, *-ekstrahering* og *–konvertering*. Jeg har valgt å la dette kapittelet følge denne inndelingen, og beskriver derfor de ulike teknologiske forskningsområdene jeg her kommer inn på etter hvordan de angriper informasjonsintegreringsproblemet, enten som et *identifiseringsproblem*, som et *ekstraheringsproblem* eller som et *konverterings-problem*.

Noen systemer er også under utvikling som prøver å angripe to eller tre av problemområdene samtidig, eller andre som velger nye tenkemåter for å unngå denne spesifikke problemstillingen. Jeg gir en kort presentasjon av disse forskningsområdene på slutten av kapitlet.

4.1.1.1 Informasjonsidentifisering

Informasjonsidentifisering betegner den prosessen man følger for å identifisere en ønsket kilde på Internett, hvor *kilde* kan være alt fra databaser og vanlige HTML-sider til strukturerte filer. Kildelokaliseringsprosessen kompliseres ved det enorme søkerommet, de utallige kildene og søkeverktøyene, og ved ulikhetene i søkemotorenes grensesnitt.

En måte å oppfatte identifiseringsfasen på [22], er at den også skal inkludere identifiseringen av de spesifikke informasjonselementene i de aktuelle dokumentene. Jeg har derimot valgt å betrakte dette under informasjonsekstraheringsfasen, ettersom jeg ser at dette er en vanlig oppfatning i litteraturen [19, 20, 29].

Kildeidentifiseringsprosessen betegnes også som *Information Retrieval*. I det følgende beskriver jeg ulike anvendelsesområder knyttet til Information Retrieval som prøver å løse informasjonsidentifiseringsproblemet:

a) Virtuelle bibliotek (Browse by Subject).

Disse er hovedsaklig organiserte samlinger av linker til ulike kilder på Internett. Bibliotekene møter imidlertid problemer som:

- Webkilder som biblioteket refererer til, slettes
- Vansker med å oppdatere biblioteket hurtig nok
- Ettersom biblioteket vokser, øker vanskene med å kategorisere nye referanser

- Hvordan kan man opprettholde en enkel indeks til innholdet i biblioteket?

Virtuelle bibliotek er avhengige av en kraftig markedsføring av sin egen eksistens for å kunne håpe på at informasjonstilbydere på Internett velger akkurat deres bibliotek som tilgangspunkt for sine kilder. De virtuelle bibliotekene er deretter ansvarlige for at de til syvende og sist avverterer den tilbudte kilden i listene over tilgjengelige websider.

Et virtuelt bibliotek er avhengig av et koordineringssenter. Administrasjonen av biblioteket kan enten være distribuert, sentral eller en kombinasjon av begge. Hensikten med et koordineringssenter er å samle og prosessere referanser til kildene. I tillegg til å drive reklame for sin egen eksistens, må de virtuelle bibliotekene evaluere alle nye dokumenter/kilder som de blir tilsendt. Dette kan enten gjøres manuelt eller ved å inkludere en indivudell evaluering av kilden som akkumuleres for flere brukere. Brukerpågangen kan i tillegg hjelpe til i bestemmelsen av hvor lenge kilden bør være tilgjengelig i listene.

De virtuelle bibliotekene må kunne håndtere utvikling av skjemaer som sendes til informasjonstilbydere. Her spør man etter tilleggsinformasjon som beskriver kategorien kilden går inn under. Bibliotekene må også kunne svare på spørringer.

b) Søkemotorer (Keyword Query).

Søkemotorene opprettholder en indeks til webkilder hvor man kan spørre etter gitte temaer. Disse indeksene kan enten hentes inn hos virtuelle bibliotek, fra spesifikke kilde-oversikter eller lages av *web wanderers, roboter, spiders, crawlers* og *worms*. Søkemotorene kan eliminere noen av problemene knyttet til virtuelle bibliotek som for eksempel at webkilder slettes, ved å dynamisk regenerere kilde-indeksen. Søkemotorer kan også, ved hjelp av *crawlers, spiders* og *worms* osv., finne fram til en mer komplett liste over kilder. Hovedproblemet ved bruk av roboter og crawlers er at de forlanger en altfor stor Internett båndbredde. Mange servere har opprettet protokoller som utelukker disse programmene fra webområdet.

Mange av søkemotorene har imidlertid innsett at det ikke er nok å tilby en mengde informasjon uten noen form for guide, de har derfor kommet med tiltak som skal tilby ekstra innhold til informasjonen. Infoseek og Excite har for

eksempel ”channels” eller ”guides” hvor fagpersoner identifiserer kvalitetsrike webområder og leder brukeren til disse, istedenfor kun å la dem skrive inn et ord og lete på egenhånd. Søkemotorene går altså mot å bli innholdstilbydere, ikke bare en indeks til andre kilder.

I tillegg til problemet rundt opprettholdelsen av en dekkende indeks, har man vanskelighetene som oppstår idet man endrer valg av søkemotor og brukergrensesnittet endres. Det forskes i dag rundt mulighetene til å utvikle et standard grensesnitt for søkemotorer.

c) Multiple Searches/Repetitive Queries.

Ettersom Internett ikke har en entydig indeks, opererer alle søkemotorene med ulike, ukomplette indekser. Systemer som CUSI (Configurable Search Interface) benyttes for å kjøre den samme spørringen mot flere ulike søkemotorer for å oppnå et mer komplett svar. Dette brukergrensesnittet hjalp i utgangspunktet brukere til å opprette enkle websider som utførte spørringer mot en rekke søkemotorer. Brukeren kunne skrive inn spørringen, velge en søkemotor fra ei liste og så kjøre spørringen for prosessering. Etter å ha sett resultatene kunne brukeren beholde spørringen og velge en annen søkemotor fra lista for kjøring.

Prosessen benytter World Wide Web forms og cgi scripts. Forbedringer ble tidlig oppdaget: Man kunne automatisk kjøre den samme spørringen mot alle søkemotorene i parallell. Denne ideen resulterte i utviklingen av multi-threaded gateways (MT), som er en moderne programmeringsmetode som tillater at applikasjoner skifter mellom seriell prosessering og parallell prosessering.

d) Social Filtering

Social Filering, også kalt Community Based Navigation eller Collaborative Filtering bygger på ideen om å indeksere ruter som brukere følger når de navigerer på Internett. På denne måten bygger man opp sider som inneholder linker til de mest populære og mest besøkte sidene på Internett.

Den mer vanlige bruken av Social Filtering er imidlertid å la brukerne evaluere nettstedene de oppsøker, og så la denne evauleringen være disponibel for andre brukere. Dette gjør at en kunde som for eksempel oppgir å like en gitt type bøker, kan bli opplyst om en bok som sannsynligvis vil falle i smak ettersom brukere med liknende preferanser har evaluert denne boken høyt.

e) Nye protokoller for informasjonsutveksling

I forskningsmiljøet rundt Virtuelle bibliotek har standarden Z39.50 Information Retrieval Protocol blitt utviklet. Z39.50 gir endel nye muligheter, for eksempel inneholder standarden API'er for applikasjonsutvikling og muligheten til å indeksere dokument-samlinger og gjøre dem tilgjengelige på nettet. Imidlertid finnes ingen mulighet til å konvertere eksisterende nettverksdatabaser til Z39.50. Fra et teknisk synspunkt møter Z39.50 standarden de fleste kravene til et Internet Information retrieval system, men likevel har ikke standarden blitt tatt i utbredt bruk utenfor virtuelle bibliotek fordi den er for kompleks og stor til å kunne implementeres i kommersielle applikasjoner.

Z39.50 Lite baserer seg på den opprinnelige Z39.50 protokollen, men skal være en mindre og lettere pakke. Hensikten er å øke aksepten for protokollen i Internett-samfunnet. STARTS (The Stanford Protocol for Internet Retrieval and Search) er en annen protokoll som ved hjelp av en uformell IR standard prøver å imøtekomme kravene til Internett-samfunnet. Mange store software-produsenter har vært med i utviklingen av denne protokollen; Infoseek, Fulcrun, Verity, Microsoft, Netscape, HP med flere. Hovedproblemet ved STARTS-protokollen er at den kun kan benyttes til enkle dokument-søk. I tillegg er protokollen ikke kompatibel med den kraftige Z39.50 standarden, noe som utelukker muligheten til å inkludere i framtiden noen av de sofistikerte metodene som Z39.50 kan tilby.

4.1.1.2 Informasjonsekstrahering

Informasjonsekstraheringsfasen av integreringsproblemet ser jeg på som et todelt problemområde:

- *Elementlokalisering* innenfor de dokumentene man har identifisert i identifiseringsfasen
- *Ekstrahering* av informasjon

Jeg ser nå først på elementlokaliseringsprosessen (som også kalles Information Extraction) med aktuelle anvendelser av teorien, og deretter på ekstrahering av informasjon med sine tilhørende anvendelser.

Elementlokaliseringsprosessen kompliseres ved at dataene i kildene stort sett er semistrukturerte. Med semistrukturerte data menes data som enten mangler struktur på et fysisk nivå (eksempel kan være HTML-filer) eller mangler struktur på logisk nivå (datamengden kan være for stor og uregelmessig til å kunne

modelleres i en entydig struktur). Ettersom kildene allerede eksisterer, og man ikke har noen måte å endre på den allerede eksisterende strukturen i kildene man vil inkludere i systemet, må man ta hensyn til hver enkelt kildestruktur [16]. Dette medfører problemer som:

- *Semantiske ulikheter mellom kildene*. Ettersom hver datakilde har blitt opprettet med forskjellige formål og for ulike organisasjoner, er informasjonen ikke modellert på samme måten. For eksempel kan en informasjon som er lagret som attributt ett sted, være lagret som element et annet sted, navnene på attributter og tabeller vil ikke være like osv.
- *Ulike betegnelser*. Kilder kan bruke forskjellige navn for å referere til det samme objektet.

Man har derfor vanligvis ingen entydig framgangsmåte når man skal identifisere elementer i et dokument. Man er avhengig av ulike former for *pattern matching*.

I litteraturen jeg har benyttet kalles de ulike systemene som er utviklet for å foreta informasjonsidentifisering for *Information Extraction* (IE) systemer. *Information extraction* systemer analyserer tekst-dokumenter for å finne spesielle typer informasjon [29]. Systemet prøver ikke å forstå hele teksten i et dokument, men leter seg fram til de deler av dokumentet som er relevant for analysen. Relevans er gitt av forutbestemte kriterier som spesifiserer så nøyaktig som mulig hva systemet skal se etter. IE systemer opererer på flere nivåer; fra gjenkjenning av ord til setningsanalyse og eventuelt forståelse av emne på dokument-nivå.

Ordbok-problemet som består i å ha et vokabular som dekker alle mulige sjargonger og uttrykk som kan finnes i et ikke-standardisert dokument, er en av de store utfordringene i tekstbaserte information extraction systemer. Den vanligste framgangsmåten hittil er å benytte en enkel ordgjenkjennings-teknikk. Man ser som oftest bort fra syntaksen i dokumentet. Dette gjør at prosedyren går glipp av viktig informasjon eller ren forståelse av innholdet i dokumentet. På den annen side, dersom man inkluderer syntaktisk analyse i prosessen blir den veldig ineffektiv ettersom parseren må behandle større mengder data.

Det forskes uansett mye i dag rundt mulighetene til å forbedre extraction prosedyrene fra enkle ord-gjenkjennings prosedyrer til komplekse analytiske prosesser over innhold og tema. I [6] beskrives noen forbedringer; NLP (Natural

Language Processing) som prosesserer sider hvor innholdet danner hele setninger, har eksempelvis blitt forbedret i en ny web-parser, Webfoot, hvor informasjonen som trekkes ut er *relasjonen* mellom ulike fakta. Crystal er et NLP system som automatisk innfører et sett med område-spesifikk informasjon som den lærer fra eksempler og som brukes som regler når informasjon skal hentes. Fra Webfoot får Crystal et sett med instanser som består av tekstsegmenter kalt *felter*, pluss en *ordbok* som brukes til å kategorisere ordene i teksten. På denne måten lager Crystal et system for informasjonen i et dokument som gjør det mulig å trekke ut relevant informasjon.

Ekstrahering av informasjon eller *informasjonsekstraheringsproblemet* består i å overføre den identifiserte informasjonen til en uttrykt eller implisitt datamodell [22]. Det vil i praksis si å hente ut de elementer med attributter og verdier som man tidligere i prosessen har identifisert. Med implisitt datamodell menes at strukturen på informasjonen som trekkes ut er kjent, og derfor ikke trenger nærmere spesifisering. Med eksplisitt datamodell menes at man utvikler en beskrivende struktur for hvordan informasjonen i de aktuelle kildene opptrer, og så trekker ut informasjonen i henhold til denne modellen.

Ekstrahering av informasjon gjøres tradisjonelt ved å utvikle skreddersydde prosedyrer for hvert enkelt tilfelle eller domene. I dette tilfellet snakker man om en implisitt datamodell og man vet, fordi informasjonsplasseringen allerede er identifisert, nøyaktig hvor man skal gå for å få tak i de aktuelle elementene. En annen metode er å benytte en eksplisitt datamodell som gjelder for flere nettsteder, og enten utvikle en prosedyre som henvender seg på samme måten til alle nettstedene (fordi informasjonen er plassert etter samme modell i alle nettsidene), eller benytte eksempelbasert læring til å utvikle et system som „husker“ eller lærer seg hvor det skal hente ut den aktuelle informasjonen [17].

„Wrappers“ er en type systemer som er spesielt designet for å ta seg av informasjonsekstraheringen i integrasjonsproblemet. Man skiller mellom generelle og spesielle wrappere [19]. Generelle wrappere kan defineres uavhengig av skjemaet som kildene er bygget opp etter, de baserer seg på en generell objekt-modell som er gyldig for alle de aktuelle kildene. Spesielle wrappere trengs derimot for kilder med en uregelmessig og heterogen form. Her må wrapperen ta

hensyn til hver enkel kildes format. Formålet med wrapperen er å få kildene til å likne vanlige databaser som brukeren kan utføre spørringer mot ved å benytte *query languages* som f.eks. SQL.

Wrapping-teknologien innebærer at man genererer informasjon om kilden, som for eksempel hvilke spørsmålstyper som kan besvares og hvilke datatyper svaret vil inneholde, og i tillegg svarer wrapperen på spørringene som kommer til websiden. Wrapperne kan gjerne ha et grafisk grensesnitt der eieren av informasjonskilden spesifiserer websidens muligheter og wrapperens funksjonalitet, eller de kan være en del av et større informasjonsintegrerende system (se punkt 4.1.1.3).

I [5] beskrves forskning omkring wrapping-teknologien, spesielt rettet mot de følgende tre feltene:

- en måte å definere web-kilders kapasitet, samt definisjon av en protokoll som kan publisere denne kapasiteten
- Utvikling av et verktøysett som tilbyr et grafisk grensesnitt og et språk til å spesifisere kildens egenskaper og wrapperens funksjonalitet
- Utvikling av teknologi til en generator som produserer web wrappere

I Ariadne prosjektet [17] beskrives et „inductive-learning system“, Stalker, som er et verktøy for å lage og opprettholde wrappere. Stalker lærer et sett med ekstraheringsregler for å trekke ut informasjon fra ei side.

4.1.1.3 Informasjonskonvertering

Informasjonskonvertering vil si at den informasjonen man nå har identifisert og ekstrahert, må konverteres over på det formatet som den aktuelle applikasjonen eller brukeren ønsker. Etter ekstraheringen av dataene opptrer de vanligvis i det formatet som er gitt av datamodellen, enten den er implisitt eller eksplisitt. Men dette er ikke nødvendigvis det formatet som er ønsket av sluttbruker. Det er nødvendig å konvertere dataene til et ønsket presentasjonsformat.

Det å lage „mappingen“ mellom informasjonen etter at den er ekstrahert og det endelige presentasjonsformatet er vanligvis ikke noe problem, ettersom informasjonen kun skal overføres fra et kjent format til et annet. Når denne „mappingen“ er på plass kan man lage en enkel prosedyre som konverterer dataene ifølge „kartet“. Disse prosessene er relativt enkle, problemet ligger ofte i å velge et egnet presentasjonsformat. Man kan her tenke seg flere forskjellige formater. For eksempel kan dataene benyttes til å fylle en relasjonsdatabase. I

dette tilfellet må dataene struktureres etter relasjonene i databasen; de må legges i de rette kolonnene i de riktige tabellene.

Det er derimot mer vanlig å benytte åpne objekt-modeller som presentasjonsformat. Et eksempel er OEM (Object Exchange Model), som er en selv-beskrivende (eller „tagged“) objekt-modell hvor objektene har etikett, type, verdi og et valgfritt identifikasjonsnavn. OEM modellen er utviklet i samsvar med liknende, åpne objekt-modeller som har vært i utstrakt bruk i informasjonskonvertering. OEM er utviklet spesielt for TSIMMIS prosjektet og er beskrevet nærmere i [25].

En liknende standard er XML. XML-strukturen er formet som et tre hvor det øverste elementet er roten. Det kan kun være et rot-element i et XML-dokument. Hvert element består av attributter og verdier, og eventuelt nye elementer. Dette presentasjonsformatet har fått stor oppmerksomhet den senere tiden, og nyter støtte fra mange av de store programvareleverandørene som f.eks. Microsoft og IBM.

Det man i den senere tid har kommet fram til under forskningen rundt presentasjonsformater, er at de ikke nødvendigvis kun må fungere som et endelig presentasjonsformat *etter* en informasjonsinnhenting, men at man derimot kan benytte dem som et publikasjonsformat direkte på Internett. Det vil si at man snur problemstillingen og ønsker å standardisere informasjonen i kildene. På denne måten vil man kunne unngå mange av problemene forbundet med elementidentifisering og informasjonskonvertering.

Noen velger imidlertid å behandle de tre delproblemene i informasjons-integreringsproblemet på en litt annen måte; de benytter seg av resultatene i de tre forskningsområdene som underliggende teknologi, og henvender seg deretter til brukeren med et enhetlig system for informasjonsintegrering. To ulike systemer av denne typen er *intelligente agenter* og *informasjonsintegrerende systemer*:

Intelligente Agenter eller agentbaserte tjenester på Internett er et relativt nytt konsept som bygger på studier rundt kunstig intelligens. Internett-agentene har, fra kundens synspunkt, til hensikt å sanke informasjon fra ulike webservere og sile denne etter gitte kriterier. Informasjonen presenteres så for kunden på et brukervennlig format. Det er vanskelig å snakke om intelligente agenter som en

egen teknologi. Det er enklere å definere dem som systemer som benytter seg av underliggende teknologiske løsninger for å kunne utføre bestemte oppgaver.

Agentene fungerer som et enhetlig grensesnitt mot brukeren, men „under overflaten“ kan man finne ulike kombinasjoner av teknologi. Man kan for eksempel tenke seg agenter som benytter seg av søkemotorer eller „spiders“ for å finne kildene, NLP for å identifisere informasjonselementene i kildene, og ekstrahering av informasjon etter OEM-modellen. Disse underliggende operasjonene og teknologiene vil være transparente for sluttbrukeren, som kun vil møte et grensesnitt for spørringer og resultater.

Agenter skiller seg fra annen software ved at man individuelt kan tilpasse dem til de behov man har, de kjører kontinuerlig og er delvis selvstyrt. Disse egenskapene gjør agenter svært nyttige ved behandling og prosessering av store mengder data, noe som blant annet gjør dem egnet til Internett og elektronisk handel.

En mye brukt anvendelse av intelligente agenter er i *produkt-anbefalingssystemer*. Denne typen systemer benytter tre ulike typer filtrering som underliggende teknologi ved informasjonssprosesseringen: *Innholdsbasert filtrering* (Content-based), *samarbeidsbasert filtrering* (Collaborative-based) eller *restriksjonsbasert filtrering* (Constraint-based).

Innholdsbasert filtrering: Systemet prosesserer informasjon fra mange ulike kilder og prøver å ekstrahere nyttige egenskaper og elementer fra innholdet. Nøkkelord-baserte søk er den enkleste teknikken. Denne formen innebærer å sammenlikne kombinasjoner av nøkkelord. En mer avansert form baserer seg på å trekke ut betydningen av et dokumentinnhold (se kap. 4.1.1.2 Information Extraction).

Samarbeidsbasert filtrering: Systemet benytter tilbakemelding og vurdering fra andre brukere til å filtrere ut relevant informasjon. Denne typen system prøver ikke å forstå produktbeskrivelsene, men bruker kundenes evaluering til å lage en indeks over hvor godt produktet faller i smak (se kap. 4.1.1.1.d Social Filtering).

Restriksjonsbasert filtrering: Systemet benytter, som i innholdsbasert filtrering, egenskaper ved produktet til å bestemme dets relevans. Denne formen for filtrering krever i tillegg at kunden spesifiserer problem og løsning i form av

variable, områder og grenser. Ved hjelp av disse parametrene kan deretter systemet filtrere ut informasjon som ligger innenfor de ønskede grensene.

Informasjonsintegrerende systemer: Et system som skal foreta integrering av informasjon (*Information Integration system*) må ta utgangspunkt i et sett med eksisterende datakilder. Disse kildene kan variere mellom ulike typer, som for eksempel databaser, vanlige HTML-sider eller strukturerte filer. Oppgaven til systembyggeren i dette tilfellet er manuelt å designe et *mellomledd* mellom bruker og kilder. Mellomleddet må inkludere en beskrivelse av alle kildene som er med i kildesettet.

Spørringene mot et slikt mellomledd kan gjøres på et entydig språk, og uavhengige av forhold som f.eks språket kildene benytter for å foreta spørringer, at informasjonen er distribuert over mange kilder og plasseringen til kildene. Mellomleddet bestemmer hvilke kilder som skal benyttes, hvordan man skal få tak i den ønskede informasjonen, hvordan og hvor man skal midlertidig lagre data, og hvordan man på en effektiv måte kan hente inn data fra kildene [16].

Hver informasjonsintegrerende applikasjon opererer med en underliggende modell for domenet. Denne modellen syr sammen de ulike kildemodellene, som hver beskriver innholdet i en enkelt informasjonskilde. Ved en spørring mot applikasjonen velger systemet dynamisk et aktuelt sett med kilder, og genererer så en plan for effektivt å produsere de ønskede dataene. For brukeren av systemet ser det ut som om alle kildene tilhører en eneste database. Systemet skjuler fasene for planlegging og innhenting for brukeren.

Når en bruker utfører spørringer mot et slikt mellomledd for informasjonsintegrering, krever dette en omformulering fra systemets side. Det er nødvendig å tilpasse spørringen til kildenes individuelle strukturer. Dette gjøres ved at man først utfører en omformulering av spørringen til et format som passer til skjemaet i mellomleddet. Deretter kjøres spørringen gjennom en optimaliserer som genererer en *query-execution plan*. Til slutt kjøres spørringen mot et sett med *wrappers* på kildesidene (se punkt 4.1.1.2). Hver kilde har en wrapper som tilbyr en beskrivelse av informasjonen i den gitte kilden og som omformer kildedataene til et format som integrasjons-systemets prosessor kan behandle.

Informasjonsintegrerende systemer bygges ofte på en semistrukturert datamodell. Fordelen med å bygge informasjonsintegrerende systemer på en slik semistrukturert datamodell er at i mange tilfeller er kildene i seg selv ustrukturerte

med hensyn på logisk nivå. I tillegg kan systemer som bygger på denne typen struktur på en enkel måte inkludere data fra mange forskjellige datamodeller.

Når jeg nå har gjennomført en kartlegging av den tilgjengelige teknologien og de oppnådde forskningsresultatene innenfor temaet informasjonsintegrering, er jeg interessert i å undersøke hvordan problemet behandles „i virkeligheten“, det vil si hvordan finansinstitusjoner per i dag løser informasjons-integreringsproblemet.

4.1.2 Hvordan informasjonsinnhentingen foregår "i virkeligheten"

Det er i det siste blitt vanligere på Internett med egne sider innenfor gitte sektorer som tilbyr hjelp til brukere med hensyn til informasjonsinnhenting og beslutningsstøtte. Denne formen for mellomledd blir stadig viktigere på Internett ettersom informasjonstilbudet er økende, mens det samtidig mangler en entydig indeks over det komplette innholdet.

Som forrige kapittel viser, kan brukeren velge flere veier for å finne den ønskede informasjonen. Den mest vanlige metoden er som nevnt under punkt 4.1.1.1 å benytte søkemotorer, og så lete seg fram gjennom et uttall av søkeresultater. Intelligente agenter blir også etter hvert vanligere. Alternativt kan man altså velge å henvende seg til et mellomledd for informasjonsinnhenting. Sidene kan tilby brukeren jevnlig oppdatert informasjon som danner grunnlag for et valg brukeren skal ta. Brukeren kan velge seg fram til den typen informasjon som er ønsket, og man kan eventuelt oppgi personlige preferanser eller opplysninger som danner grunnlag for det søkeresultat som blir lagt fram.

For brukeren kan denne typen informasjonsbehandling og søkehjelp framstå som en intelligent agent. Den underliggende teknologien er imidlertid ikke alltid knyttet opp mot det som betegnes som intelligente agenter. I den følgende kartleggingen av nettsteder i USA hvor brukere kan få informasjon om oppdaterte lånerenter fra flere låntilbydere på samme side, beskriver jeg den teknologien som benyttes av hvert enkelt nettsted for å kunne presentere den innsamlede informasjonen.

Kartleggingen viser, at det blant de som svarte til i dag ikke blir tatt i bruk teknologi som kan karakteriseres som intelligente agenter. Kun ett av nettstedene oppgir at de har vurdert bruken av „spiders“ og/eller „crawlers“ under

informasjonsinnhentingen fra bankene/finansinstitusjonene. Grunnen til at denne metoden framstår som veldig vanskelig for det aktuelle firmaet, er at ingen av de underliggende bankene eller finansinstitusjonene har valgt å framlegge informasjonen på den samme måten, selv om alle gjerne oppbevarer informasjonen i en eller annen form for tabell.

Det oppgis at markeds-teamet henter inn informasjon fra institusjonene og oppsøker så de tilhørende webområdene. Her undersøker de om informasjonen stemmer overens med den innsamlede informasjonen, og at den blir oppdatert på en jevn basis. Etter dette skrives den aktuelle URL til institusjonens nettside inn i en robot database som firmaet opprettholder. Robotene monitorerer de sidene som er listet i databasen. I de første to ukene blir den informasjonen som roboten kommer fram til nøye overvåket. Dersom informasjonen viser seg å oppfylle kravene til jevnlig oppdatert informasjon, kan den robot-baserte innhentingen overta ved denne banken eller institusjonen. Kontakten min i firmaet oppgir at selve innhentingen av HTML-tabellen enkelt kan importeres inn i Lotus 123 eller MS Excel. Derfra kan så informasjonen legges i en database på en web-server.

Brukeren av denne typen websider kan ikke skille mellom den informasjon som er hentet inn av en ansatt på markedsavdelingen, og den informasjonen som en automatisert robot finner fram til på egenhånd. For brukeren framstår den informasjonsinnhentingen som *brukeren* må foreta, som assistert av en intelligent agent idet man ved enkle tastetrykk finner fram til informasjonen som er den som er best tilpasset den enkelte bruker. Dette kan gjelde tilpasninger som stat-tilhørighet i USA, hvor stort lån man har fra før eller hvilke type lån man ønsker å ta opp. Istedenfor å søke på sidene til 200 låntilbydere, blir de 200 låntilbyderne samlet på ett sted og man får hjelp til å foreta både undersøkelser og valg.

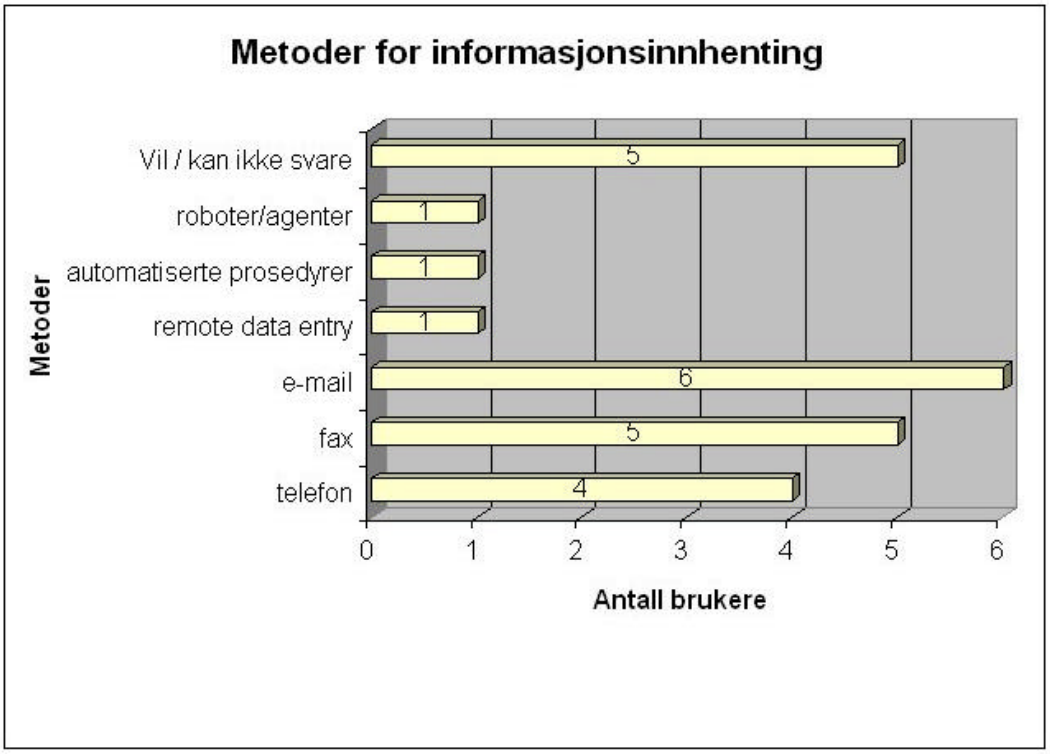
Ideen om å presentere og tilby kundetilpassede løsninger innen finansmarkedet basert på innsamling av informasjon fra banker og finansinstitusjoner, har foreløpig vært benyttet spesielt i USA. Jeg har derfor tatt kontakt med noen av de største amerikanske nettstedene for å kartlegge hvilken underliggende teknologi de benytter ved innsamlingen av data. Tabellen under viser noen av webområdene jeg har oppsøkt, og hvilken teknologi webområdet benytter for å hente inn den ønksede informasjonen. Her har jeg kun tatt med de mest signifikante svarene jeg fikk. Den komplette tabellen finnes derimot som vedlegg 1.

Web-område: **Underliggende teknikk/teknologi**
www.loanshop.com / www.mortgage.com: Sidene oppdateres med daglig informasjon. Informasjonen blir publisert fra finansinstitusjonene i form av et Excel spreadsheet via elektronisk mail. Ved Mortgage.com ekstraheres ratene fra spreadsheetet og sendes via en ny e-mail til en spesiell mailbox. Derfra går dataene automatisk inn i SQL serveren hvor de kan aksesseres av web-brukerne. I tillegg finnes et brukergrensesnitt hvor data kan legges inn manuelt.
www.interest.com / www.mmis.com: Dette webområdet er avhengige av at låntilbyderne faxer inn deres tall hvor de deretter tastes inn manuelt i en database.
www.bankrate.com: Bankrate ringer opp bankene for å få med siste endringer i ratene.
www.homefair.com: Låntilbyderne publiserer sine rater på webområdene til Homefair på et gitt format. Et program på Homefairs' server finner disse sidene, ekstraherer ratene og lagrer dem i en database. Formatet som benyttes for bankene er en komma-separert fil med et innholdsformat som Homefair spesifiserer.
www.themortgagenetwork.com: The Mortgage Network opererer med manualer fra alle lånetilbyderne som er tilknyttet dem, som beskriver produktene og rettningslinjene deres. Hver morgen sender disse låntilbyderne faxer med lister over dagens rater. Ved The Mortgage Network går de så gjennom alle ratelistene og finner de beste ratene for et bestemt låneprodukt. Denne informasjonen blir så publisert på websidene.
Www.loanz.com: Opererer gjennom et underliggende firma Laser Rates, som mottar faxer med rater fra alle låntilbyderne som Loanz.com benytter. Disse ratene legges manuelt inn i Laser Rates' sitt system. Deretter sendes ratene til Loanz.com sitt web-område.

www.rate.net.com: Rate.Net opererer med fem forskjellige prosedyrer for informasjonsinnhentingen: 1. *Telefon.* Markedsundersøkere ringer opp forskjellige markeder og samler inn data som manuelt legges inn i en database 2. *Fax*. Automatiske faxer sendes ut hver dag til forskjellige finans-institusjoner med de aktuelle ratene og dataene. Dersom disse dataene trenger oppdatering gjøres dette, før faxene sendes tilbake. Her blir den oppdaterte informasjonen manuelt lagt inn i en database. 3. *Email*. Automatiske e-mail sendes ut til forskjellige institusjoner som har tilgang til e-mail, hvor informasjonen studeres av de aktuelle personene som oppdaterer dataene og sender så mailen tilbake. På rate.net sida er denne prosessen automatisert. De returnerte e-mailene blir analysert, feilsjekket og dataene automatisk lagt inn i en database. 4. *Robots* eller *Spiders*. Rate.net har roboter som drar rundt til webområdene, samler inn data og rater og oppdaterer databasen. Rate.net oppgir dette som en god idé med god programmering, men veldig få av bankenes webområder er pålitelige nok til å tilby denne typen informasjon fordi de ikke oppdateres på regelmessig basis. 5. *Remote data entry*. Remote data entry er en av favorittmekanismene til rate.net, hvor finansinstitusjonene logger på en av websidene til rate.net og oppdaterer deres informasjon direkte.
www.E-Loan.com: Mottar informasjon fra låntilbyderne på elektronisk vis (antar e-mail), og har en automatisert prosedyre som daglig laster disse ratene inn i en database slik at de kan finnes på websiden deres.
Www.yourmortgage.com: Informasjonen samles og oppdateres på flere forskjellige måter. Dette inkluderer telefonoppringninger til institusjonene og faxer med informasjonen tilbake til institusjonene slik at de kan bekrefte riktigheten av ratene. I tillegg kontrollerer de låntilbydernes websider. Samtidig ber de låntilbyderne om å kontrollere yourmortgage's sine websider og maile, faxe eller ringe om eventuelle endringer. De sier at de alltid har planlagt et „online" system som kan aksesseres via nettet som tillater et „online" finansielt system (antar de mener at de finansielle institusjonene selv kan oppdatere egen informasjon direkte på websiden), dette har imidlertid blitt utsatt.

Tabell 1 : kartlegging av teknologi-bruken for informasjonsintegrering

Tabellen under her viser hvordan svarene jeg har fått indikerer at informasjons-innhentingen foregår i de firmaene jeg har kontaktet. Av 40 utsendte forespørsler har jeg fått 15 svar. Noen benytter flere metoder:



Figur 1: Metoder for informasjonsinnhenting

- Med telefon mener jeg både når låntilbyderne blir oppringt og spurt om endringer eller når låntilbyderne selv oppgir endringer via telefon
- Med fax mener jeg både når låntilbyderne får fax der de blir spurt om endringer eller når låntilbyderne selv sender endringer via fax
- Med email mener jeg både når låntilbyderne får mail der de blir spurt om endringer eller når låntilbyderne selv sender endringer via mail
- Med remote data entry mener jeg applikasjoner som gjør det mulig for låntilbyderne å endre sine opplysninger „online“
- Med automatiserte prosedyrer mener jeg programmer som henter inn på spesielle (nett)steder informasjon til spesielle tider
- Med roboter/agenter mener jeg applikasjoner som bygger på kunstig intelligens og som benyttes i informasjonsinnhentingen

a) Avhengighetsforhold

Mange av sidene på Internett som presenterer informasjon til brukeren om oppdaterte lånerater, benytter andre firmaer som underliggende leverandører. På denne måten foretas gjerne selve informasjonsinnhentingen og

informasjonsintegreringen fra de ulike banker og finansinstitusjoner av et enkelt firma, mens mange andre kjøper resultatene og oppgir dem på sine nettsider. Jeg har laget en oversikt over de avhengighetsforhold jeg har oppdaget under kartleggingen. Denne oversikten kan sees under vedlegg 2.

4.1.3 Valg av løsning for sektoren

I dette kapitlet har jeg beskrevet informasjonsintegreringsproblemet. Problemet har jeg tidligere delt i tre; identifiseringsproblemet, ekstraheringsproblemet og konverteringsproblemet. Disse tre områdene angriper kjerneproblemet (informasjonsintegreringen) fra ulike kanter og konsentrerer seg om forskjellige deler av problembildet for å løse oppgaven.

Mange av de anvendelsene jeg har skildret, løser vanskelighetene som utgjør delproblemer i informasjonsintegreringsproblemet. Men hver for seg er det ingen av disse teknologiene som klarer å gjøre noe med det grunnleggende problemet – en manglende entydig struktur på informasjonen på Internett.

Applikasjoner som beskrevet under punkt 4.1.1.3 – informasjonsintegrerende systemer – betrakter hele problemområdet under ett, og lager en implementasjon som skal få Internett til å se ut som en enkel database for sluttbrukeren. De underliggende problemene forblir likevel de samme; semantikk, syntaks, identifisering av nye kilder, mulighet til å navigere i spørreresultatene, tilgang til kildebeskrivelser osv.

Under konverteringsproblemet har jeg diskutert presentasjonsformat som en måte å nærme seg informasjonsintegreringsproblemet på. Men det er ikke først og fremst tanken på å *konvertere* informasjon, fra et format som ekstraheres fra informasjonskildene og til et felles presentasjonsformat, som gjør dette så interessant; det er mer den muligheten man har til å publisere informasjon direkte på et standard format. Et slikt presentasjonsformatet kunne, teoretisk sett, gi struktur til all publisert informasjon på Internett. Dersom man publiserer kildene direkte på et kjent fellesformat, vil man kunne unngå deler av informasjonsintegreringsproblemet: Dersom jeg betrakter det tredelte informasjons-integreringsproblemet i lys av en slik løsning, ser jeg for eksempel at identifiseringsproblemet ikke vil bli løst. Det vil fortsatt være vanskelig å finne fram til den aktuelle kilden selv om informasjonen i kilden er strukturert. Men dersom hver kilde publiseres med en tilhørende beskrivelse av innholdet, vil dette kunne gjøre problemet mindre. Man vil ikke trenge å „parse“ en kilde for å sjekke

om den inneholder aktuell informasjon, man henvender seg bare til web-områdets definisjonsfil.

Elementidentifiseringen i ekstraheringsproblemet vil praktisk talt forsvinne. Hvert element i kildedokumentet vil være merket med identifikasjons-merkelapper. Selve ekstraheringsproblemet vil gjøres om til en enkel prosedyre for siling av den aktuelle informasjonen. Konverteringsproblemet vil også forsvinne dersom presentasjonsformatet på brukersida er identisk med presentasjonsformatet i kilden, og forenkles dersom brukeren benytter et annet presentasjonsformat; dataene er likevel så strukturerte at konverteringen ikke bør være noe problem.

Ideen om å benytte et felles presentasjonsformat ved publisering av informasjon på Internett, begynner å få stor oppslutning. Som nevnt under punkt 4.1.1.3 er mange store programvareleverandører interessert i tenkemåten, og dette setter fart på en utvikling i denne retningen.

Jeg har valgt å konsentrere meg om felles presentasjonsformat som løsning for informasjonsintegreringsproblemene i finanssektoren, ettersom jeg ser dette som det foreløpig beste alternativet.

4.2 Fellesformatet

Formålet med å utvikle et fellesformat for informasjonsinnholdet til banker og finansinstitusjoner, er å definere en standard for hvordan strukturen på informasjonsinnholdet skal være når det publiseres på Internett. I denne delen av oppgaven ser jeg først på hvordan jeg kan plassere informasjonsinnholdet jeg finner at banker og finansinstitusjoner publiserer, i *en entydig datastruktur*. Klassediagrammet som jeg her kommer fram til skal publiseres på et presentasjonsformat. Jeg velger derfor et egnet format, og viser til slutt et eksempel på den strukturerte informasjonen presentert på valgt fellesformat.

4.2.1 Ulike løsninger på modellering av informasjonsinnhold

Før jeg starter arbeidet med å komme fram til en entydig informasjonsstruktur, er jeg avhengig av å finne en datamodelleringsteknikk som egner seg i mitt tilfelle.

Jeg har vurdert flere ulike teknikker og metoder, og deretter valgt den jeg mener passer best [35]:

a) OMT (Object Modeling Technique)

OMT-teknikken består av fire *faser*.

- *Analyse,* hvor resultatet er objekt-, dynamisk- og funskjonell modell.
- *System design,* hvor resultatet er en struktur av basis-arkitekturen i systemet.
- *Objekt design,* som produserer et design-dokument med detaljerte objekt-, dynamiske- og funskjonelle modeller.
- *Implementasjon,* hvor koden produseres.

OMT deler *datamodellering* i tre ulike deler:

- *Objektmodell,* beskrives ved
  - Objektmodell-diagram
  - Ordbok for dataene
- *Dynamisk Modell,* beskrives ved
  - Tilstandsdiagram
  - Globale hendelses- og flyt-diagram
- *Funksjonell Modell,* beskrives ved
  - Dataflyt-diagram
  - Begrensninger

Objektmodellering i OMT brukes til å beskrive den statiske strukturen til et system med klasser og relasjonene mellom dem. Dynamisk modellering er her brukt til å uttrykke relasjoner og oppførsel som endres over tid. Den dynamiske strukturen beskrives først og fremst ved tilstandsdiagrammer, mens funksjonell modellering benytter dataflyt diagrammer.

b) CRC (Class Responsibility Collaborator)

CRC behandler først og fremst design-fasen i software-utvikling. Prosessen kan deles i tre deler:

- *Identifisere klasser og ansvar (responsibilities).* Klasser identifiseres og grupperes basert på felles attributter. På tilsvarende måte dannes også superklasser. Hver klasse skrives så opp på et CRC-kort, hvor man i tillegg noterer ned subklasser og/eller super-klasser. Hensikten er å vise

klassestrukturen i systemet. Man analyserer så for å finne hendelser og informasjon som assosieres med hver klasse. På bakgrunn av denne informasjonen kan man så definere ulike typer ansvar i systemet.

- *Tildeling av ansvar*. I denne fasen tildeles klassene de ansvarstypene man har oppdaget. Det er viktig at ansvarstypene er så generelle som mulig, og at de plasseres så høyt i systemet som mulig. Dette for at subklasser skal kunne arve ansvar fra superklassene.
- *Identifisering av samarbeid*. Denne fasen identifiserer hvordan klassene samarbeider. De kortene som inneholder klasser som samarbeider tett, plasseres nært opptil hverandre.

CRC regnes mer som en teknikk enn som en modelleringsmetode. Teknikken tas ofte i bruk av de andre metodene for å definere systemkravene i klasser.

c) OOSE (Object Oriented Software Engineering) / Objectory

Denne metoden har som hovedområde å støtte utviklingen av store sanntidssystemer, men inneholder ulike modeller for utvikling av delprosjekter:

- *Use Case Model* definerer det som ligger utenfor systemets oppførsel ved hjelp av *actors*, og systemets innside ved hjelp av *use cases*.
- *Domain Object Model*. Her blir objektene fra ”den virkelig verden” kartlagt.
- *Analysis Object Model* inneholder en ideell beskrivelse av systemets struktur. Hensikten med denne modellen er å presentere en robust system-struktur.
- *Design Object Model* beskriver hvordan kildekoden eller implementasjonen bør utvikles og skrives.
- *Implementation Model* representerer implementasjonen av systemet.
- *Test Model* består av planene for testing, spesifikasjoner og rapporter.

Objectory metoden er på sett og vis forløperen til UML, da i særlig grad Use Case notasjonen i UML, ettersom personen bak Objectory, Ivar Jacobsen, ble en av grunnleggerene av UML.

d) UML (Unified Modeling Language)

UML er et modelleringsspråk som benytter diagrammer for å strukturere ideer til analyse og design [15, 35]. De ulike diagrammene er:

- *Use Case diagram* er en beskrivelse av mulige buisness situasjoner som en applikasjon kan behandle. Et use case beskriver måten en person, en organisasjon eller et eksternt system interagerer med det gitte systemet. Use case diagrammer viser personene (actors) som er involvert, hendelsene (use cases) de er involvert i og skillet mellom system og verden utenfor. Kombinasjonen av use cases og deres korresponderende use case diagram representerer en use case modell.
- *Klasse-diagram*, viser klassene i et system og relasjonene mellom dem (inkludert arv, aggregering og assosiasjon). Klassediagrammene er selve grunnlaget i objekt orientert modellering og brukes til å vise den statiske strukturen i systemet . Hver klasse består av et klasse-navn, attributter og metoder. Attributtene defineres ved en beskrivelse av hva de inneholder, typen og eventuelt en indikasjon av mulige variabel-verdier. Metodene dokumenteres med en beskrivelse av logikken.
- *Komponent-diagram* definerer de software-komponentene som inngår i en større programbit. Diagrammet viser også grensesnittene mellom disse komponentene og hvordan sammenhengene mellom dem er.
- *Deployment diagram* viser konfigurasjonen av enhetene i systemet, det vil si de ulike hardware komponentene samt hvilken software som vil kjøre på hver av dem.
- *Tilstands-diagram* brukes for å vise hvordan et objekt opererer. Diagrammet viser de ulike tilstandene et objekt kan befinne seg i og hvordan overgangen fra en tilstand til en annen forekommer. Tilstandene er gitt av attributt-verdiene til objektet, mens overgangene fra en tilstand til en annen styres av objektets metoder.
- *Interaksjons-diagrammer:*
  - *Sekvens-diagram* brukes til å definere logikken i et use case scenario, og til å vise logikk-flyten i systemet. Sekvensdiagrammene viser de objekttypene som er involvert i hvert use case, meldingene som går mellom dem og responsverdiene som er assosieres med disse meldingene. Objektene er instanser av klassene som defineres i klassediagrammet. Objektene skilles fra klassene ved at navnet understrekes.
  - *Collaboration-diagram* viser hvordan klassene i systemet samarbeider, det vil si hvordan objektene i systemet sender meldinger mellom hverandre

(objektene er instanser av klassene). Collaboration diagrammet tegnes vanligvis samtidig med klassediagrammet.

4.2.2 Vurdering av egnethet for domenet og valg av løsning

UML er en standard notasjon for modellering av objekt-orienterte systemer. Hovedpoenget med UML er at det er et modelleringsspråk, ikke en metode eller metodologi (selv om en metode er underveis også her, den blir derimot kalt The Unified Process). UML definerer et visst antall diagrammer som man kan benytte for å beskrive et system, sammen med en beskrivelse av hva hvert diagram betyr. Språket beskriver derimot ikke prosessen man må følge for å utvikle programvare. En slik prosess-beskrivelse, eller metode, inkluderer en liste over oppgaver som må utføres, i hvilken rekkefølge de bør utføres, hva som produseres, hvilke egenskaper som trengs for å utføre hver oppgave, hvem som gjør hva osv. De tradisjonelle metodologiene består både av notasjon og metode.

Før UML ble opprettet som en standard, fantes mange uavhengige modellerings-metoder, men ingen felles notasjon dem imellom. Det fantes derfor ingen kompatibilitet og mulighet for kommunikasjon mellom systemer som var utviklet etter de ulike metodene. Det er dette problemet man har provd å løse ved å opprette standarden UML. Med en standardisert notasjonen kan også programvare-utviklere kommunisere seg imellom ved at det underliggende systemet beskrives av et felles språk. Utviklerene kan deretter velge den metoden de ønsker for å utvikle systemet.

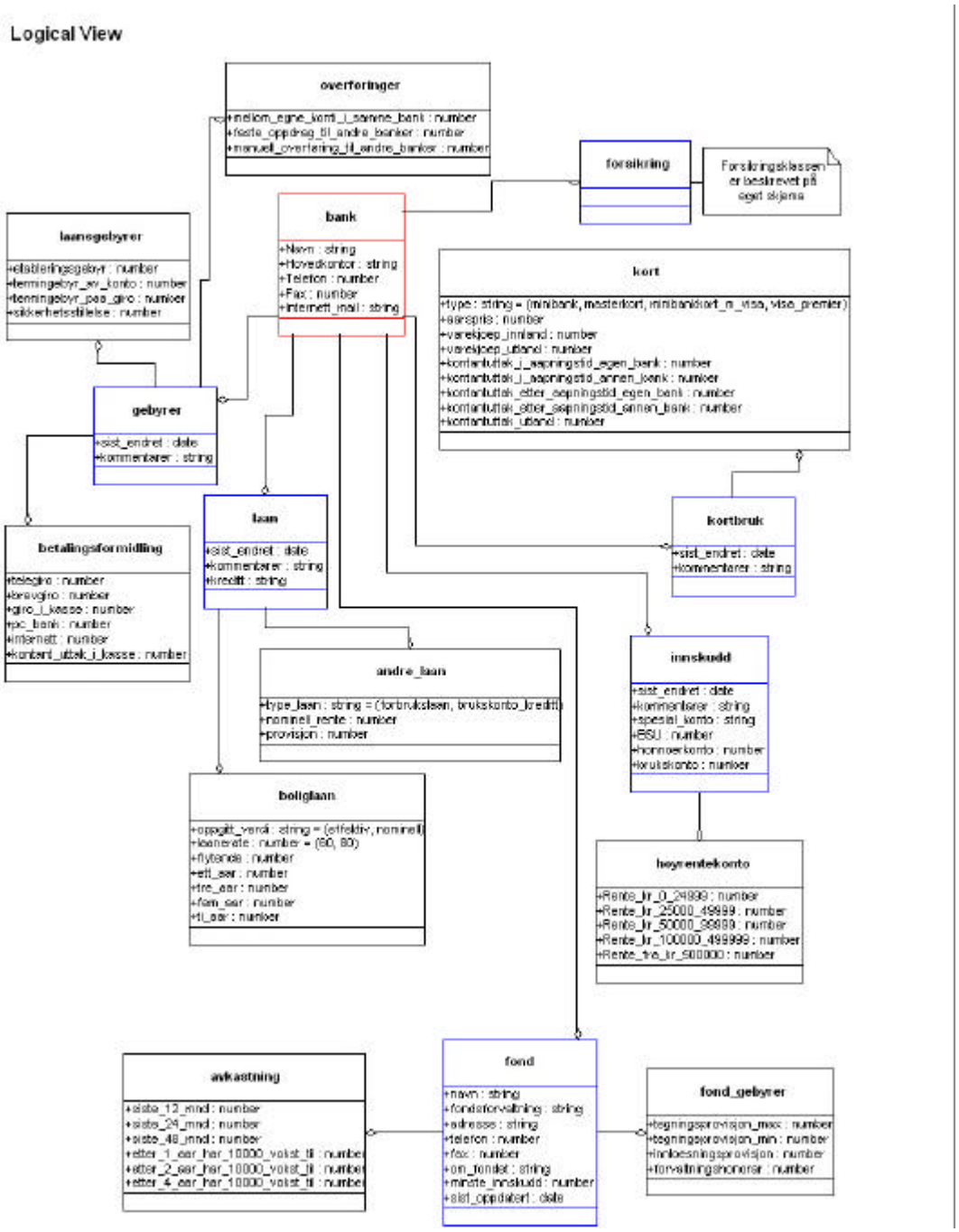
UML ble utviklet gjennom et samarbeid mellom mange av store forkjempere av kjente modelleringsmetoder som for eksempel Rational, Objectory og OMT med støtte fra Object Management Group (OMG). Det var enighet omkring behovet for et standard notasjonsspråk innenfor objekt orientert utvikling, og UML ble resultatet.

Mitt behov for en datamodelleringsmetode bestod i å kunne definere datastrukturen i et system. Jeg var ikke ute etter å implementere et komplett system eller utvikle programvare. UML skiller seg derfor ut som det best egnede alternativet, ettersom det representerer et språk for beskrivelse av et system i motsetning til en metode hvor man følger en prosess for å utvikle et system. Innenfor de ulike framgangsmåtene for kartlegging ved hjelp av diagrammer har jeg valgt å benytte meg av klassediagrammer. Jeg ser at bankenes

informasjonsstruktur kan deles inn i ulike klasser, med relasjoner dem imellom, som arv, aggregering og assosiasjon. Dette er den anbefalte diagramtypen dersom man skal analysere et system med hensyn på hvordan systemets struktur er bygd opp.

4.2.3 Presentasjon av klassediagrammet for domenet

Med utgangspunkt i datastrukturen på sidene til Norsk Familieøkonomi, og med klassediagrammene i UML som virkemiddel, har jeg kommet fram til en oversikt som kan gjelde som en generell datastruktur for banker og finansielle institusjoner. Jeg har delt informasjonen inn i klasser og tildelt dem attributter med eventuelle predefinerte verdi-valg. Det endelige UML-skjemaet er vist i figur 2 her under.



Figur 2: UML skjemaet for informasjonsstrukturen

Forsikring er én av seks hovedkategorier jeg har tatt med i informasjons-strukturen til bankene. Det komplette skjemaet som viser strukturen over forsikrings-informasjonen er vist som vedlegg 3.

4.2.4 Valg av felles presentasjonsformat

I del 4.1.1.3 presenterer jeg ulike alternativer til felles presentasjonsformat for data. De mest aktuelle er OEM (Object Exchange Model) og XML (eXstensible Markup Language).

OEM er en representasjonsmodell som egner seg godt for semistrukturerte data, og er en enkel og åpen standard. I praksis er den veldig lik XML. XML har derimot fått en mye bredere aksept i utviklingsmiljøer, ikke minst på grunn av støtten fra Microsoft og IBM. XML har etterhvert blitt definert som en internettstandard, og jeg velger derfor å konsentrere meg om XML i det videre arbeidet med fellesformat for bankenes- og finansinstitusjonenes informasjons-innhold. Jeg ser i det følgende på hva XML-standarden innebærer i detalj, hvilke typer applikasjoner som er pådrivere for en slik standard-endring og hva det vil si, for eksempel for meldingsutvekslingen mellom bedrifter, at man endrer publiseringsstandard fra HTML til XML.

World Wide Web Consortium (W3C) har definert Standard Generalized Markup Language (SGML) som er spesialisert for dataoverføring over web. Både HTML og XML er subsett av denne standarden. XML tilbyr fordeler med hensyn til hvordan dataene beskrives og utveksles mellom web-baserte applikasjoner, ved å benytte et enkelt standard-basert format. Hypertext markup language (HTML) tilbyr universelle metoder å vise data på, mens XML tilbyr universelle måter å arbeide direkte med dataene på.

Forskningen rundt XML er viktig ettersom bedriftene i stadig større grad går over fra den klassiske, to-delte klient-server applikasjonsmodellen, til tre-delte modeller hvor browseren arbeider mot en webserver som i sin tur arbeider mot en database hvor informasjonen lagres. XML gir bedriftene muligheter til å kombinere data fra flere ulike kilder og presentere dem på felles format.

XML skiller seg fra HTML hovedsaklig ved de tre følgende punkter:

- Tilbydere av informasjon kan definere nye tags og attributt-navn etter ønske.
- Dokumentstrukturer kan utvikles til et hvilket som helst kompleksitetsnivå.
- Ethvert XML dokument kan inneholde en valgfri beskrivelse av gramatikken til bruk for applikasjoner som trenger å foreta gyldighetstest av strukturen.

HTML som standard lider følgelig av enkelte mangler sammenlignet med XML. De viktigste kan sammenfattes i tre punkter; *extensibility*, *structure* og *validation* [9]:

- *Extensibility*. HTML tillater ikke brukeren å spesifisere egne tags eller attributter for å parametrisere eller på annen måte semantisk kvalifisere dataene sine. Dette gjør det vanskelig å søke etter spesifikk informasjon på

Internett. HTML gir få muligheter til merking av metadata, og denne merkingen er uansett inkludert i selve HTML-dokumentet, slik at hele dokumentet må lastes ned for å undersøke innholdet av en bestemt tag. Andre typer informasjons-filer på Internett har ingen mulighet for merking av data. XML er et tekst-basert format som tillater at utviklerne beskriver, leverer og utveksler strukturerte data mellom en rekke applikasjoner hos klienter som kan visualisere og manipulere disse dataene. XML dokumenter kan enten være selv-beskrivende eller det kan følge med et XML-Skjema eller DTD (Document Type Definitions). XML gjør det altså mulig å opprette en fil på et webområde som inneholder en meta-spesifikasjon av alt som finnes på området. Dette gjør søking av informasjon mye mer effektiv.

- *Strcture*. HTML tilbyr ikke spesifikasjon av dyp struktur som er nødvendig for å representere database-skjemaer eller objekt-orienterte hierarkier. Ved HTML blir vanligvis dataene hentet inn i en tabell som vises på web-siden. Dersom dataene leveres som et XML dokument som tar vare på den opprinnelige informasjonen slik som kolonnenavn og datatyper, kan informasjonen benyttes av brukeren på en helt annen måte enn før. Man kan blant annet manipulere dataene og endre oppsettet på dem etter ønske. Strukturen i et XML dokument er hovedsakelig et tre hvor rot-elementet er det øverste elementet. XML. XSL-(eXtensible Stylesheet) prosessoren gjør det mulig for utviklere å transformere XML data til HTML gjennom et *stylesheet* som definerer presentasjonsregler. Et XSL *stylesheet* er en samling programmeringsregler for hvordan informasjon skal trekkes ut av XML dokumentet og inn i et annet format, for eksempel HTML.
- *Validation*. HTML tilbyr ingen type spesifikasjon av språket som tillater at bruker-appliakasjoner sjekker dataene for en gyldig struktur ved import.

De applikasjonene som kommer til å være pådrivere av godkjenningen av XML er de som ikke kan fungere innenfor begrensningene som HTML legger. Disse applikasjonene kan deles i fire grupper[9]:

- Applikasjoner som krever at web-klienten henter informasjon fra to eller flere heterogene databaser.
- Applikasjoner som prøver å distribuere en stor del av prosesserings-byrden over fra web-serveren til web-klienten.

- Applikasjoner som krever at web-klienten presenterer ulike framstillinger av de samme dataene til forskjellige brukere.
- Applikasjoner hvor intelligente web-agenter prøver å skreddersy informasjonsinnhenting etter hver brukers individuelle behov.

Alternativet til XML for disse typene applikasjoner er tilleggsprogramvare i form av script-biter eller java-applets som leveres sammen med HTML-dokumentet.

Det finnes hovedsaklig to måter man kan implementere utveksling av data mellom ulike bedrifter innen samme fagområde. Den ene er å kreve at alle bedriftene følger samme standard systemer. På denne måten produserer alle bedriftene data i det samme formatet, og når de skal overføres møter man ikke format-problemer på mottaker-siden fordi formatet er på den avtalte formen. Den andre måten er å utvikle et standard format for datautvekslingen. Et slikt format sikrer at all informasjon inn til et system ved import av data er på det gitte formatet, samtidig som all informasjon ut fra et system blir omdannet til gitt format ved eksport. Dette var opprinnelig hensikten med SGML, og XML overtar nå denne rollen.

Muligheten for meldingsutveksling mellom bedrifter er noe av det viktigste XML har å tilby, hvor man med meldingsutveksling mener utveksling av data mellom enten organisasjoner eller mellom ulike applikasjoner innen samme organisasjon. Meldingsutveksling mellom bedrifter tas i dag hånd om av EDI (Electronic Data Interchange). EDI har vært med på å automatisere buisness-to-buisness transaksjoner, men for mange små bedrifter er det ikke økonomisk mulig å ta i bruk et EDI-system og mange av disse bedriftene benytter derfor fortsatt tradisjonelle metoder for informasjonsoverføring som telefon og fax.

Mange EDI-systemer krever tilkopling til et VAN (Value-Added Network), ikke bare Internett. Dette kreves på grunn av sikkerhet, pålitelighet og tilgjengelighet men er mer kostbart enn Internett. Og Internett trenger man jo gjerne for å benytte e-mail eller andre operasjoner som går utover VAN'ets nettverksgrenser. Installasjon av EDI-systemet er heller ikke noen enkel prosess; man må kontakte fagkyndige operatører som kan opprette og drifte systemet innen bedriften. Muligheten for å drive buisness-to-buisness interaksjoner over Internett virker stadig mer forlokkende på bedrifter, både på grunn av lavere kostnader og muligheten til å knytte til seg nye samarbeidspartnere. Man kaller dette Internett

EDI. To flaskehalser vanskeliggjør Internett EDI: sikkerheten og meldingsformatet. Og det er under meldingsformatet at XML kan være til nytte.

Generelt er kunnskapen om HTML større enn EDI forståelsen. XML likner HTML og gjør det derfor enklere å lære seg det nye XML formatet. Med XML kan derfor terskelen for å starte meldingsutveksling bli ganske lav.

Det som altså bør inkluderes i dagens Internett, er innføringen av en universell forståelse, en standard måte å representere data slik at det blir lettere for software programmene å manipulere dataene. Og her ser jeg foreløpig XML som den beste løsningen.

4.2.5 Utvikling av Dokument Type Definition

Et XML-dokument er strukturert og kan godt opptre uten nærmere spesifikasjon. Innholdet i fila er likevel bare *instanser* av objekter, og dersom man ønsker å kjenne strukturen, uten å involvere dataene, kan det være hensiktsmessig å opprette ei fil som beskriver datastrukturen.

4.2.5.1 Valget mellom DTD og XML Schema

Man har to muligheter til å definere strukturen i et XML-dokument: XML-skjema eller DTD. Definisjonen av XML-Schema er fortsatt under utvikling hos W3C-gruppen „The W3C XML Schema Working Group". På sidene hvor de beskriver de nåværende aktivitetene deres sier de at selv om XML 1.0 standarden kommer med Document Type Definition (DTD) som metode for å deklarere dokument-strukturen, trengs en kraftigere mekanisme etterhvert som automatiserte prosedyrer skal prosessere XML-dokumenter. De delene som må forbedres er blant annet hvordan komponent-delene i en applikasjon hører sammen, dokument-strukturen, attributtene og datatyping.

XML-skjemaet er selv en XML-fil og gir derfor muligheter som å ekspandere dokumentet. Selv om XML-skjema inneholder alle mulighetene DTD'en gir, og i tillegg tilbyr en kraftigere måte å definere XML-strukturen på, har jeg valgt å benytte meg av DTD, ettersom dette er en etablert standard (XML Schema er fortsatt under utvikling).

4.2.5.2 Document Type Definition (DTD) i detalj

Et XML-dokument har i utgangspunktet ingen <koder> som er ferdig definert, slik som i HTML. Alle slike koder må deklareres i en såkalt "Document Type Definition" (DTD). En DTD er rett og slett en del av XML-dokumentet, og skal alltid komme først i XML-dokumentet eller finnes som et separat dokument som XML-dokumentet refererer til.

Med en DTD spesifiserer man hva XML-dokumentet *kan* inneholde av koder, hva det *må* inneholde, og om det er påkrevet rekkefølge av disse kodene. En DTD er altså en syntaktisk deklarasjon av en nærmere bestemt dokumenttype, og en DTD kan brukes som en slags kontrakt mellom sender og mottaker av et XML-dokument for å avgjøre om dokumentet er gyldig eller ikke. Denne prosessen kan eventuelt automatiseres slik at et program gjøres ansvarlig for valideringen av XML-dokumentet basert på DTD'en. Det finnes derimot tilfeller der validering av XML-dokumentet ikke er viktig, og her trengs ingen DTD. XML syntaksen i ethvert XML-dokument må være velformet, noe som innebærer at dokumentet må følge en rekke syntaktiske krav (som er strengere enn dem som stilles til HTML-dokumenter). Velformet betyr at dokumentet følger disse reglene:

- Alle koder som omslutter et innhold (dvs. har en <start> og en </start>-kode) skal nøstes inni hverandre
- Alle parameterverdier (eks. <lånetype="parameterverdi"/>) skal omsluttes med anførselstegn. Alle koder og parameternavn er "case-sensitive".

Konseptet velformet XML gjør det altså mulig for brukeren å klare seg uten DTD, noe som igjen gjør at et XML-dokument enklere kan spres til et stort antall brukere, samtidig som det stilles lavere krav til de applikasjonene som skal behandle informasjonen.

Microsoft har vist stor interesse for XML, og satser tungt på bruken av denne standarden, både i fremtidige Web-baserte applikasjoner og som lagringsformat i Office-produktene sine. Det siste vil i tilfelle bety at man med en XML-prosessor kan behandle informasjon fra et Office-dokument. Man vil også enklere kunne integrere informasjon fra ulike kilder; både fra ulike web-kilder, fra web-kilder og lagrede Office-dokumenter og mellom lagrede Office-dokumenter.

4.2.5.3 DTD basert på bankenes informasjonsstruktur

Under utviklingen av en egnet DTD, som kan dekke den generelle informasjonsstrukturen til banker og finansinstitusjoner, har jeg tatt utgangspunkt i UML-skjemaet beskrevet under punkt 3.2.3. Den komplette DTD'en er lagt ved denne rapporten som vedlegg 4. Jeg viser her kun en del av DTD'en, den delen som spesifiserer lånetilbudene:

```
<!ELEMENT laan (boliglaan, andre_laan)>
<!ATTLIST laan      sist_endret CDATA #IMPLIED
                    kommentarer CDATA #IMPLIED
                    kreditt CDATA #IMPLIED>
<!ELEMENT boliglaan (effektiv, nominell)>
<!ELEMENT effektiv (flytende, ett_aar, tre_aar, fem_aar, ti_aar)*>
<!ATTLIST effektiv  laanerate (60 | 80) "60">
<!ELEMENT nominell (flytende, ett_aar, tre_aar, fem_aar, ti_aar)*>
<!ATTLIST nominell  laanerate (60 | 80) "60">
<!ELEMENT flytende (#PCDATA)>
<!ELEMENT ett_aar (#PCDATA)>
<!ELEMENT tre_aar (#PCDATA)>
<!ELEMENT fem_aar (#PCDATA)>
<!ELEMENT ti_aar (#PCDATA)>
<!ELEMENT andre_laan (forbrukslaan, brukskonto_kreditt)>
<!ELEMENT forbrukslaan (nominell_rente?, provisjon?)>
<!ELEMENT brukskonto_kreditt (nominell_rente?, provisjon?)>
<!ELEMENT nominell_rente (#PCDATA)>
<!ELEMENT provisjon (#PCDATA)>
```

4.2.6 Utvikling av XML-dokument

Med utgangspunkt i DTD'en kan jeg nå sette opp et faktisk XML-dokument som inneholder data, altså instanser av elementene som er spesifisert i DTD'en.
I dette kapitlet viser jeg hvordan jeg går fram for å implementere et slikt eksempel på XML-dokument som følger datastrukturen i den ferdige DTD'en.

4.2.6.1 XML-dokument basert på DTD

I XML-dokumentet plasserer jeg dataene i en tre-struktur med elementet <informasjon> som rot. Rot-elementet kan kun forekomme én gang. Roten har under seg mange <Bank>-elementer som i sin tur har elementer av typen <laan>, <innskudd> osv. Når det gjelder de reelle dataene, som renteverdier og informasjon om fond eller banker, har jeg i denne XML-fila tastet inn fiktive data

for hånd. Den komplette XML-fila kan sees under vedlegg 5. Her viser jeg kun et lite utdrag av XML-dokumentet som viser innskuddstilbudet til en tenkt bank:

```
<innskudd>
 <brukskonto>0.027</brukskonto>
 <honnoerkonto>0.0</honnoerkonto>
 <BSU>0.067</BSU>
 <hoeyrentekonto>
  <Rente_kr_0_24999>0.0625</Rente_kr_0_24999>
  <Rente_kr_25000_49999>0.0625</Rente_kr_25000_49999>
  <Rente_kr_50000_99999>0.0625</Rente_kr_50000_99999>
  <Rente_kr_100000_499999>0.067</Rente_kr_100000_499999>
  <Rente_fra_kr_500000>0.068</Rente_fra_kr_500000>
 </hoeyrentekonto>
 <sist_endret>01.04.99</sist_endret>
 <kommentarer>Hva som helst</kommentarer>
 <spesial_konto>nytt</spesial_konto>
</innskudd>
```

I forbindelse med prosjektet ”Multimedia Banking” eksisterer det allerede en database som inneholder innsamlet informasjon fra banker og finansinstitusjoner. Databasen, som er en Microsoft Access fil, oppdateres hver dag. Jeg har valgt å benytte meg av den informasjonen som er samlet i denne databasen i det videre arbeidet med å utvikle demonstrasjoner på bruk av XML.

For å trekke informasjonen ut av databasen og inn i en XML-fil, har jeg benyttet meg av konverteringsverktøyet DB2XML som man kan laste ned gratis fra Internett. Dette programmet transformerer data fra ulike typer databaser til XML-filer, og dersom det er nødvendig, opprettes også DTD’en på bakgrunn av meta-informasjonen i databasen. Jeg har kun benyttet meg av XML-filer som produkt fra DB2XML-programmet. Et eksempel på XML-fil som dannes i DB2XML kan sees som vedlegg 6.

XML har, som jeg har vist, muligheten til å benytte ulike måter å definere strukturen på; man må velge mellom DTD og XML-skjema. Jeg har videre sett at et liknende valg må tas for det som gjelder visualiseringsspråket. Visualiseringsspråket benyttes til å utvikle formatterende filtre for informasjonen i XML-dokumentene. Man trekker ut de dataene man ønsker å se, og gir dem den fasongen man ønsker. Disse filtrene er de såkalte „stylesheets“.

4.2.6.2 Valget mellom CSS og XSL

Det finnes for øyeblikket tre ulike tilgjengelige språk som kan benyttes til å utvikle Microsoft Internet Explorer kompatible *stylesheets*; CSS (Cascading Style Sheets), DSSSL (Document Style Semantics and Specification Language) og XSL (eXtensible Style Language). DSSSL er den mest kompliserte standarden av de tre, og er en del av SGML-prosjektet. DSSSL-definisjonen er på over 250 sider og det snakkes lite om denne som en realistisk standard for XML-fremvisning. CSS og XSL er derimot de to språkene som er mest brukt og lettest tilgjengelige. Det viktigste punktet som skiller disse to standardene er at med CSS er man avhengig av at XML strukturen er identisk til den strukturen man presenterer dataene med. Siden et av hovedpoengene med XML jo er å skille informasjonsinnholdet fra måten dataene blir presentert på, sier det seg selv at bruk av CSS har sine begrensninger.

CSS er et enklere språk enn XSL og er dermed lettere å lære. Dette snur seg derimot ofte til en ulempe ettersom man må kompensere det enkle språket med en desto mer kompleks XML-struktur. Et eksempel kan være dersom man i Bank-strukturen ønsker å visualisere Lånetilbudet foran Innskudd. Med CSS måtte man eventuelt ha endret XML-fila ved å la de to element-gruppene bytte plass. Med XSL er dette unødvendig – man kan aksessere alle elementene uavhengig av hverandre.

XSL tilbyr også mange tilleggsfunksjoner som CSS ikke har:

- Generere overskrifter til datafeltene
- Man kan lage mer sofistikert layout ved å inkludere HTML tabeller
- Data-felter kan forekomme flere enn en gang i et *style-sheet*
- Man har tilgang til informasjon som er lagret som attributt-verdier
- Man kan endre rekkefølgen på elementene i XML-fila

Jeg velger derfor å konsentrere meg om XSL som utviklingsspråk for stylesheets i det videre arbeidet.

4.2.6.3 XSL i detalj

XSL tillater at formatteringsinformasjon assosieres med elementer i kildedokumentet (XML-dokumentet). Selve prosessen med å omforme et XML-

dokument, fra et strukturert tre til et format som kan visualiseres, kan deles i to underprosesser;

- Omforming av kilde-treet til et nytt tre. Man kan filtrere og sortere nodene i det opprinnelige treet, og strukturen i det endelige treet behøver ikke være lik den opprinnelige. Man kan generere tilleggs-informasjon som for eksempel overskrifter eller innholdsfortegnelser. Denne transformasjons-prosessen legger også til informasjon i det endelige treet som trengs for formatteringen. Man kan kalle denne omformingen for *mønster-gjenkjenning* ettersom man trekker ut de nodene som passer til det mønsteret man spesifiserer i XSL-dokumentet.
- Formattering. Her blir nodene i det endelige treet oppfattet som formatterings-objekter eller flyt-objekter. Syntaktisk beskrives formatteringsobjektene som XML-elementer hvor egenskapene representeres ved attributt/verdi par. Her gir man for eksempel farge eller font-størrelse til elementene. Man kan kalle formatteringen for anvendelse av *regler* på det resulterende treet etter omformingsprosessen.

4.3 Tjenesteutnytting

Tanken bak dette kapitlet er først og fremst å illustrere enkelte tjenester innen finanssektoren på Internett, som baserer seg på informasjonstilbudet til banker og finansinstitusjoner, og som kan dra fordel av et felles presentasjonsformat som XML. Jeg har valgt ut tre eksempler (blant mange mulige). Jeg beskriver først hvordan tjenestene fungerer i dag, og deretter hvordan jeg tror XML vil kunne tilby nye muligheter og forbedringer innen de aktuelle tjenestene. Til slutt i kapitlet viser jeg tre praktiske eksempler på bruk av XML i finanssektoren.

4.3.1 Tjenesteutnyttelse av informasjonsinnholdet

Den typen informasjon som jeg har laget en generell struktur for, benyttes i flere ulike tjenester innen finanssektoren. Under punkt 4.2.4 viser jeg en inndeling av de applikasjonene som kommer til å være pådrivere av godkjenningen av XML. Finanssektoren på Internett består av flere typer applikasjoner som faller innenfor denne inndelingen, hvor hovedtrekkene er delt inn i applikasjoner som:

- krever at web-klienten henter informasjon fra flere heterogene databaser
- vil distribuere prosesserings-byrden over fra serveren til klienten
- krever at klienten presenterer ulike framstillinger av de samme dataene til forskjellige brukere
- hvor web-agenter skreddersyr informasjonsinnhenting etter brukerens individuelle behov

Alternativet til XML for disse typene applikasjoner er tilleggsprogramvare i form av script-biter eller java-applets som leveres sammen med HTML-dokumentet.

4.3.1.1 Brukerstøttet informasjonsinnhenting

Ved hjelp av intelligente agenter og mellomledd som samler informasjon fra mange banker og finansinstitusjoner prøver man å hjelpe brukeren til å finne et mer oversiktlig bilde av virkeligheten. Hovedmålet er å lete fram det beste tilgjengelige tilbudet for en bank-kunde ved å skreddersy informasjon fra flere ulike informasjonskilder og presentere den etter brukerens behov og ønske. Informasjonsinnhentingen i finanssektoren i dag vanskeliggjøres av de ulike formattypene og av det enorme informasjonstilbudet. Denne typen applikasjoner faller både innenfor applikasjonstype 4 og type 1 i inndelingen ovenfor.

Ved en overgang til XML vil man her kunne benytte seg av at informasjonen er standardisert og definert i en DTD. På denne måten vil den samme informasjonen være presentert på samme vis uansett hvilken bank man henvender seg til. Dersom en bruker for eksempel er interessert i det beste rente-tilbudet for høyrentekontoer med innskudd mellom 25.000 og 50.000, vil innhenting av denne typen informasjon være enkel for en intelligent agent (eller en hvilken som helst automatisert prosess) dersom dataene fra bankene er publisert som oppgitt i DTD'en. Man leter seg fram til det aktuelle elementet – i dette tilfellet

```
<!ELEMENT Rente_kr_25000_49999 (#PCDATA)>
```

og presenterer dette sammen med bankens navn, eventuelt ved å sile informasjonen slik at kun det beste tilbudet visualiseres for kunden.

Den intelligente agenten som foretar søket, kan også inneholde en prosedyre som validerer innholdet i bankenes sider etter DTD'en. Den typen standardisering som en DTD tilbyr ved agent-operasjoner, er av fundamental betydning. Slik situasjonen er i dag, med HTML som format for publiserte data,

kan ikke agentene operere på en tilfredsstillende måte, fordi informasjonen ikke enkelt kan sammelignes ettersom HTML ikke er et entydig format.
Alternativet til XML ved denne typen applikasjoner er "script elements" i HTML-dokumenter sammen med plug-ins og/eller Java applets.

4.3.1.2 Presentasjon

Finansinformasjon på Internett gir en kunde mulighet til å få informasjon om finansielle institusjoner og endringer i deres tilbud. For kunden er det imidlertid viktig at denne typen informasjon er enkel å finne, er oversiktlig og tar kort tid å laste ned. En bruker forventer å kunne finne, med få tastetrykk, den informasjonen som gjelder i hvert enkelt tilfelle. De forskjellige brukerne av finanskildene har gjerne ulike behov til hvordan informasjonen skal se ut. En kunde som er interessert i den laveste lånerenta bankene kan tilby, vil for eksempel ikke være interessert i å se informasjon som omhandler innskudd, gebyrer eller andre deler av bankenes komplette informasjonstilbud. I tillegg, innenfor temaet lånerente, kan den samme brukeren være interessert i å visualisere dataene etter ulike kriterier. I noen tilfeller ønsker man altså å kunne utføre små operasjoner på dataene som gjør dem enklere og mer oversiktlige, som for eksempel å sortere informasjonen etter visse kriterier eller filtrere bort uønsket informasjon.

Slik situasjonen er i dag må man enten laste ned et helt nytt, ferdig HTML-dokument, eller man er avhengig av tilleggsprogramvare som plug-ins eller script-biter som java applets for å utføre denne typen operasjoner. Dette medfører relativt lange ventetider på grunn av nedlastingstider og prosesseringstider på serveren. Denne typen applikasjoner hører til under applikasjonstype nummer 3 i inndelingen ovenfor.

Finanskilder på Internett som visualiserer data kan dra nytte av XML formatet og datastrukturen til å visualisere den samme informasjonen på ulike måter. Ettersom de forskjellige brukerne av finanskildene har ulike behov til hvordan informasjonen skal se ut, kan brukeren som leter etter laveste lånerente, og som ikke vil se informasjon som innskudd, gebyrer eller andre deler av bankenes komplette informasjonstilbud, ha mulighet til å filtrere bort uønsket informasjon. Og innenfor temaet lånerente kan brukeren nå visualisere dataene etter ulike kriterier.

Med websida publisert på XML-format trengs dataene bare å lastes ned én gang. Basert på XML-fila, som inneholder all informasjon om banken, kan man

trekke ut de feltene man er interessert i og manipulere dem som man ønsker. Man kan filtrere bort informasjon, sortere, søke etter bestemte elementer, se grafiske framstillinger av numeriske data og så videre.

Noen praktiske eksempler kan være:

- visualisere de ulike rentetilbudene på sparekontoer, sortert etter høyeste rente
- visualisere en graf over de ulike lånetilbudene, sortert etter laveste rente
- visualisere alle bankene som er med i oversikten, alfabetisk etter navn

Ettersom informasjonen som blir sendt til brukeren inneholder både data og semantisk informasjon, har slutt-brukeren et utall av muligheter til å ordne dataene etter eget ønske. Til syvende og sist er mulighetene først og fremst begrenset av fantasien til utviklerne av XSL-skjemaene.

4.3.1.3 Kalkulasjon og beregning av kostnader

Kalkulatorer på Internett innebærer både eksempel på applikasjoner hvor man foretrekker at web-klienten tar en stor del av prosesserings-byrden (applikasjonstype to i inndelingen ovenfor) og applikasjoner som baserer seg på informasjonsinnhentende agenter som garanterer oppdatert informasjon (applikasjonstype nummer fire i inndelingen ovenfor).

Kalkulasjon innen finanssektoren på Internett tillater at brukeren taster inn informasjon slik at applikasjonen kan utføre operasjoner på dataene og komme fram til personlige resultat, som for eksempel å beregne brukerens finanskostnader. Slik denne tjenesten fungerer i dag er man først og fremst avhengig av en automatisert prosedyre eller agent som henter inn informasjon fra de ulike finansinstitusjonene (med de problemer dette medfører, som jeg har beskrevet tidligere). Videre må de inntastede dataene behandles og den ønskede informasjonen trekkes ut fra en eventuell database. Dette medfører at serveren må involveres i databehandlingen, noe som forlenger prosesseringstida

Kalkulasjon av personlige finanskostnader kan gjøres enklere dersom man tar i bruk XML som underliggende teknologi fordi man har alle elementene i den nedlastede filen, man trenger kun å utføre enkelte operasjoner på dem. Med XML som basisskjema for informasjonen er det også mulig å utføre raske operasjoner som ikke krever at serveren belastes. Dette fordi all informasjon som trengs for å komme fram til et resultat ligger enten i de inntastede dataene, eller i informasjonen lagret i xml-dokumentet. Man er ikke lenger avhengig av å hente

informasjon som ligger lagret på serveren i form av en database eller ei fil av annet slag. Brukeren taster inn sine verdier, og basert på de ulike valgene som gjøres fra brukerens side henter man fram forskjellige presentasjoner av XML-fila.

4.3.2 Presentasjon av eksempler på tjenester med XML

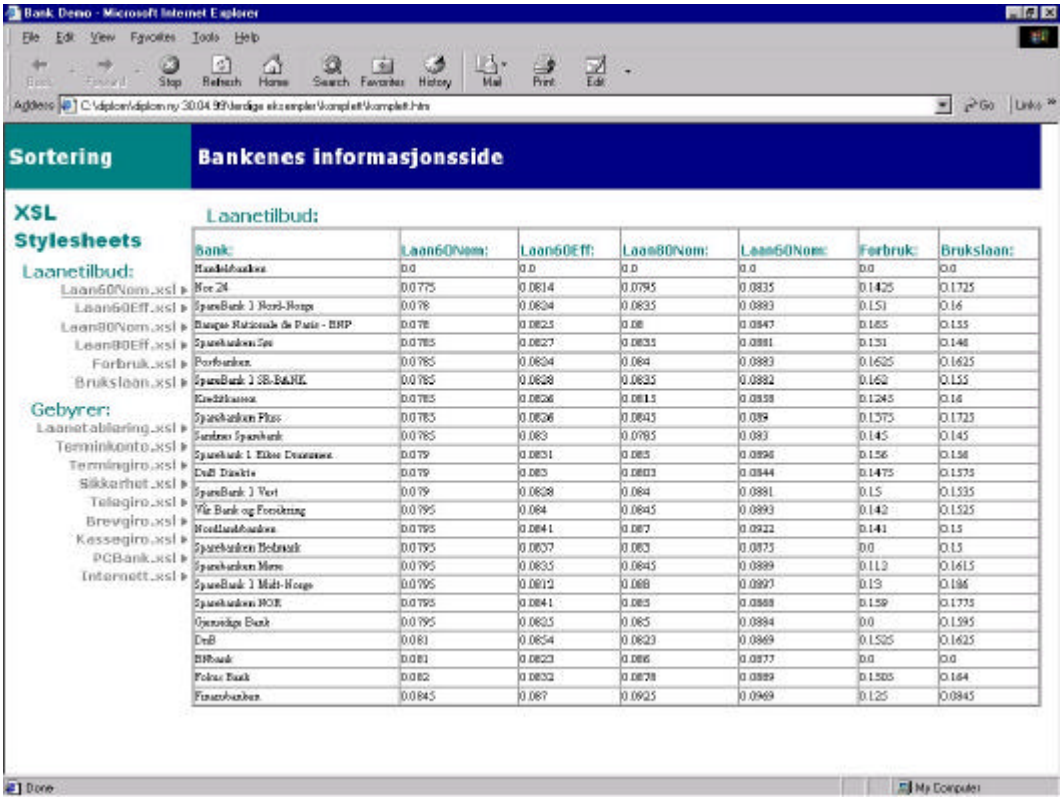
Som forklart under punkt 3.2.4.5 har jeg tatt utgangspunkt i dataene som finnes i databasen Banking.mdb som er en del av prosjektet ”Multimedia Banking”, under utviklingen av eksempler på bruk av XML i finanstjenester på Internett. Kun i det siste eksemplet, hvor jeg viser element-søk, har jeg tatt i bruk den DTD’en jeg har kommet fram til gjennom analyse av datastrukturen, og den medfølgende XML-fila. Den begrensede tida jeg har hatt til rådighet har gjort at jeg ikke har hatt mulighet til å utvikle demonstrasjoner som dekker alle de tre eksemplene på tjeneste-utnyttelse som jeg presenterte i forrige kapittel. Jeg har fortsatt utviklet tre eksempler, men alle hører inn under anvendelsestypen „presentasjon“.

4.3.2.1 Presentasjon av informasjon

Jeg har utviklet to ulike demoer for å vise hvordan man kan *presentere* og *sortere* data som er strukturert i ei XML-fil. Den siste demonstrasjonen viser hvordan man kan *filtrere* ut spesifikke deler av et XML-dokument.

a) eksempel 1 – ulike xsl-filer

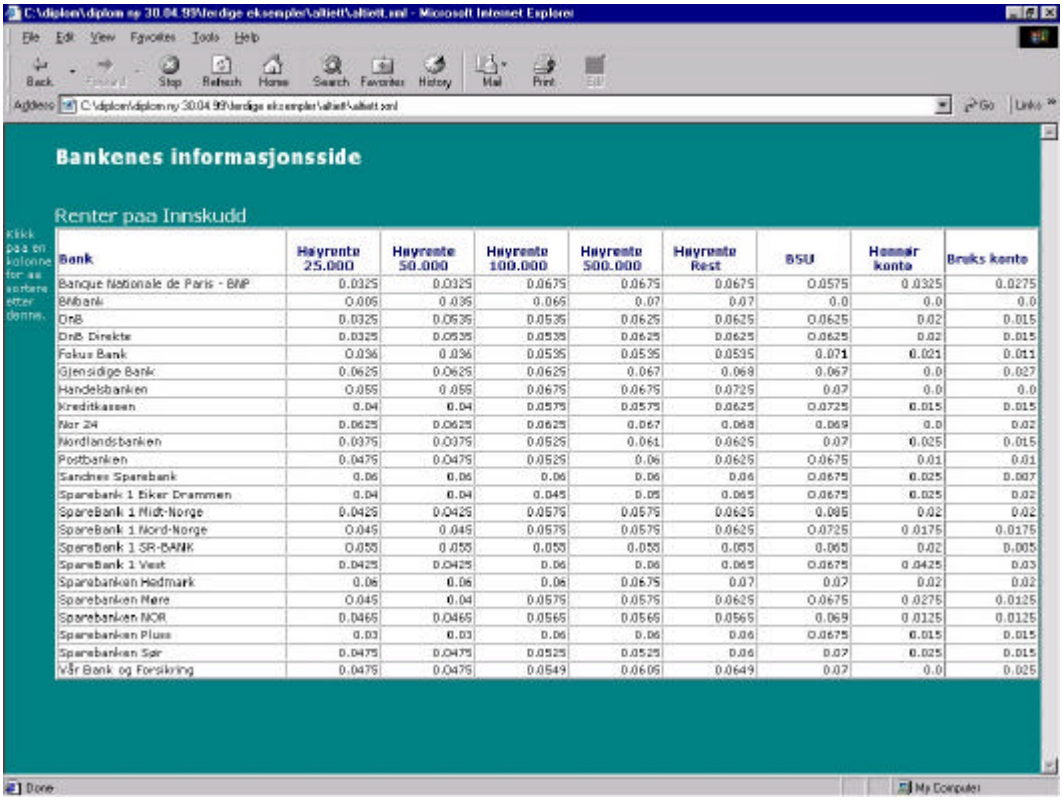
I det første eksemplet har jeg, basert på Microsoft demoen ”Multiple Views” utviklet en htm-side hvor man kan velge ulike måter å sortere lånerenter på. Det hører flere xsl-skjemaer med xml-fila, og ved å klikke på den typen sortering man ønsker, applikeres den tilhørende xsl-fila. Jeg har, for enkelthets skyld, bare tatt med lånetilbud og oversikt over gebyrene i denne demoen. Htm-sida komplett.htm består av to rammer – kontroll.htm og visning.htm. Ved å velge xsl-fil i kontroll-ramma (til venstre) applikeres denne fila som stylesheet for xml-fila som ligger under ramma til høyre (visning.htm). Kildekodene til filene komplett.htm, kontroll.htm, visning.htm og en av xsl.filene (Laan60Nom.xsl) kan sees i vedlegg 7.



Figur 3: XML Sortering med flere xsl-filer

b) Eksempel 2 – Sortering med én xsl-fil

I det andre eksempelet, som viser sortering av innholdet i ei xml-fil, har jeg tatt utgangspunkt i Microsoft demoen ”stock-sorter”.



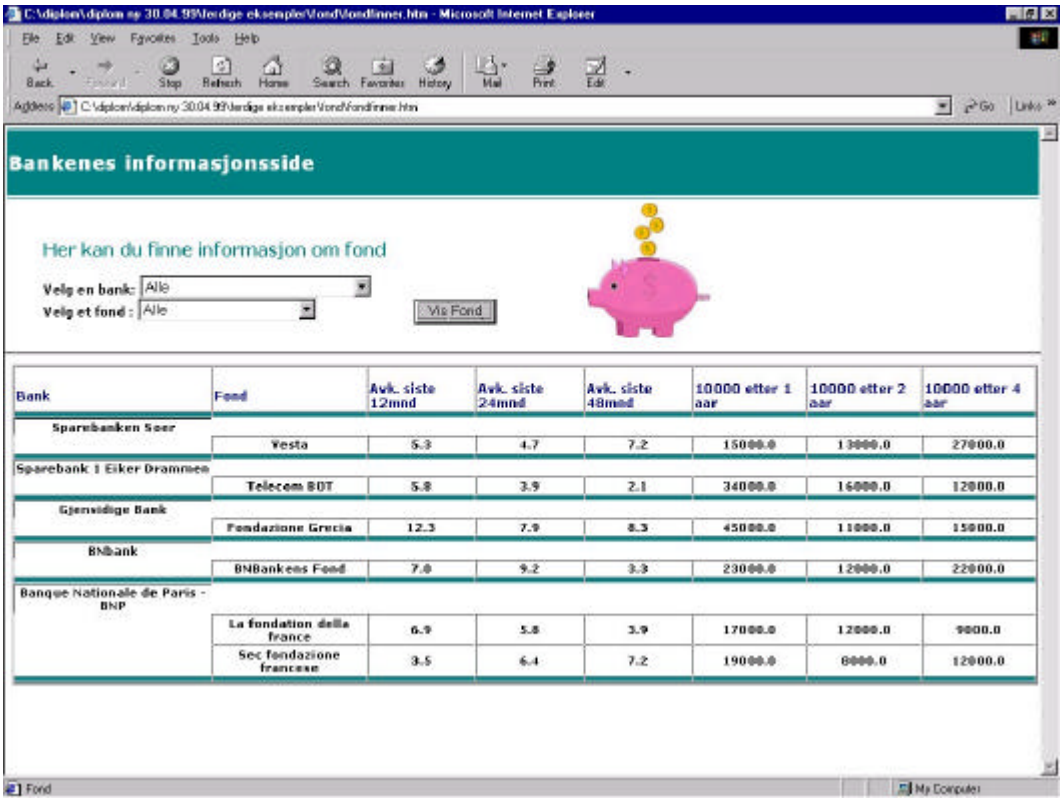
Bank	Høyrente 25.000	Høyrente 50.000	Høyrente 100.000	Høyrente 500.000	Høyrente Rest	BSU	Hjemmekonto	Bruks konto
Banque Nationale de Paris - BNP	0.0325	0.0325	0.0675	0.0675	0.0675	0.0575	0.0325	0.0275
BNbank	0.035	0.035	0.065	0.07	0.07	0.0	0.0	0.0
DnB	0.0325	0.0535	0.0535	0.0625	0.0625	0.0625	0.02	0.015
DnB Direkte	0.0325	0.0535	0.0535	0.0625	0.0625	0.0625	0.02	0.015
Fokus Bank	0.036	0.036	0.0595	0.0595	0.0595	0.071	0.021	0.011
Gjensidige Bank	0.0625	0.0625	0.0625	0.067	0.068	0.067	0.0	0.027
Handelsbanken	0.055	0.055	0.0675	0.0675	0.0725	0.07	0.0	0.0
Kreditkassen	0.04	0.04	0.0575	0.0575	0.0625	0.0725	0.015	0.015
Nor 24	0.0625	0.0625	0.0625	0.067	0.068	0.069	0.0	0.02
Nordlandsbanken	0.0375	0.0375	0.0525	0.061	0.0625	0.07	0.025	0.015
Postbanken	0.0475	0.0475	0.0525	0.06	0.0625	0.0675	0.01	0.01
Sandnes Sparebank	0.06	0.06	0.06	0.06	0.06	0.0675	0.025	0.007
Sparebank 1 Eiker Drammen	0.04	0.04	0.045	0.05	0.065	0.0675	0.025	0.02
SpareBank 1 Midt-Norge	0.0425	0.0425	0.0575	0.0575	0.0625	0.085	0.02	0.02
SpareBank 1 Nord-Norge	0.045	0.045	0.0575	0.0575	0.0625	0.0725	0.0175	0.0175
SpareBank 1 SR-BANK	0.055	0.055	0.055	0.055	0.055	0.065	0.02	0.005
SpareBank 1 Vest	0.0425	0.0425	0.06	0.06	0.065	0.0675	0.0425	0.03
Sparebanken Hedmark	0.06	0.06	0.06	0.0675	0.07	0.07	0.02	0.02
Sparebanken Møre	0.045	0.04	0.0575	0.0575	0.0625	0.0675	0.0275	0.0125
Sparebanken NOR	0.0465	0.0465	0.0565	0.0565	0.0565	0.069	0.0125	0.0125
Sparebanken Pluss	0.03	0.03	0.06	0.06	0.06	0.0675	0.015	0.015
Sparebanken Sør	0.0475	0.0475	0.0525	0.0525	0.06	0.07	0.025	0.015
Vår Bank og Forsikring	0.0475	0.0475	0.0549	0.0605	0.0649	0.07	0.0	0.025

Figur 4: XML Sortering med én xsl-fil

I denne demoen, som jeg har kalt ”altiett”, kan man sortere informasjonen direkte ved å klikke på den kolonnen man vil sortere etter. Siden er en xml-fil med en tilhørende xsl-fil spesifisert i begynnelsen av fila. Figuren over viser hvordan dette eksemplet ser ut. Kildekoden til xsl-fila kan sees under vedlegg 8.

c) Eksempel 3 – Leting etter spesifikk informasjon

I det tredje eksempelet viser jeg hvordan man kan identifisere informasjons-elementer i ei xml-fil. Demoen viser hvordan man kan lete etter spesifikk informasjon, eller benytte seg av filtre som gir som resultat kun den informasjonen man er ute etter. Eksempelet inneholder informasjon om fond som publiseres av bankene. Man kan enten vise fond etter hvilken bank man vil henvende seg til, eller velge fond, og se hvilke banker som har dette fondet.



Bank	Fond	Avk. siste 12mnd	Avk. siste 24mnd	Avk. siste 48mnd	10000 etter 1 aar	10000 etter 2 aar	10000 etter 4 aar
Sparebanken Soer	Yesta	5.3	4.7	7.2	15000.0	13000.0	27000.0
Sparebank 1 Eiker Drammen	Telecom BDT	5.8	3.9	2.1	34000.0	16000.0	12000.0
Gjensidige Bank	Fondazione Grecia	12.3	7.9	8.3	45000.0	11000.0	15000.0
BNbank	BNBankens Fond	7.0	9.2	3.3	23000.0	12000.0	22000.0
Banque Nationale de Paris - BNP	La fondation della france	6.9	5.8	3.9	17000.0	12000.0	9000.0
	Sec fondazione francese	3.5	6.4	7.2	19000.0	8000.0	12000.0

Figur 5: XML Leting etter spesifikk informasjon

Til dette siste eksemplet har jeg tatt utgangspunkt i Microsoft demonstrasjonen ”book_finder”. Fila fondfinner.htm kaller xml-fila med tilhørende xsl-fil. For hver gang man velger nye parametre (bank, fond) og trykker på ”Vis fond”-knappen, velger man utgangspunkt i xml-fila, dvs. man velger hvilken del av xml-treet som skal vises gjennom xsl-filteret som tabell i nedre del av htm-skjermbildet. Kildekodene til Fondfinner.htm og xsl-fila fond.xsl kan sees under vedlegg 9.

5 Konklusjoner, drøfting og konsekvenser

Dette kapitlet gir i første rekke en oversikt over de konklusjoner jeg trekker av arbeidet mitt. Jeg har her valgt å la inndelingen av konklusjonene følge probleminndelingen i resten av oppgaven, det vil si at jeg først omtaler de konklusjonene jeg trekker av studiene rundt det generelle informasjons-integreringsproblemet, og deretter det som gjelder fellesformatet og strukturering av finansinformasjonen. Til slutt i del en ser jeg på hvilke konklusjoner jeg kan trekke for tjenesteutnyttelsen av det strukturerte informasjonsinnholdet i finanssektoren.

I del to av dette kapitlet vurderer jeg konklusjonene mine ved å se på de problemene som vil fortsette å eksistere etter å ha tatt i bruk den foreslåtte løsningen, og i den siste delen av kapitlet konsentrerer jeg meg om de konsekvensene jeg mener arbeidet mitt kan få for den aktuelle sektoren.

5.1 Konklusjoner

I oppgaven har jeg presentert informasjonsintegreringsproblemet på Internett. Under arbeidet med kartleggingen av teori og anvendelser har jeg fått inntrykk av at mange engasjerer seg i å lette tilgangen på informasjon ved å finne stadig mer „intelligente“ systemer; for eksempel blir prosedyrene for ord-gjenkjenning stadig mer avanserte, og søkemotorene raskere og mer fleksible. Likevel kommer man, via disse teoriene og anvendelsene, ikke nærmere en løsning på selve problemet; nemlig det at Internett mangler struktur. Jeg har derfor valgt XML som løsning på problemene for finanssektoren. Dette fellesformatet for presentasjon av data omgår på sett og vis det tradisjonelle, tredelte problemet, og gjør noe direkte med kildene. Tanken bak XML er til syvende og sist å strukturere alle informasjonskilder på Internett. Dette er selvsagt et altfor ambisiøst ønske, men mange bedrifter har allerede sett nytten av å adoptere XML som presentasjonsformat.

Etter å ha kartlagt i oppgaven hvordan tilbydere av integrert informasjon i finanssektoren foretar innsamling av deres data, ser jeg et klart potensiale for XML. Kartleggingen viser at de fleste operatørene henter inn informasjon på relativt tradisjonelle og definitivt tungvindte måter. XML vil, med de egenskaper jeg har beskrevet, gjøre innhenting og integrasjon av finansinformasjon enklere, dersom aktørene opererer med en felles DTD.

Jeg har også vist at flere applikasjonstyper på Internett vil være pådrivere av en overgang til XML-format ettersom de opplever begrensede muligheter med dagens HTML-format. Applikasjonene kan med det nye formatet enkelt samle og sammenligne informasjon fra mange kilder, noe som gjør det mulig å videreutvikle blant annet mellomledds-funksjonene som trer fram på Internett mellom bruker og produsent, for å bistå brukere i informasjonsinnhenting og beslutningstaking. Man får videre mindre prosesseringsbyrde på serveren fordi dataene kun trengs å lastes ned én gang. Dette gjør at brukeren opplever en bedre tjeneste med mindre venting. Man får også muligheten til å presentere de samme dataene på flere forskjellige måter uten å måtte designe nye web-sider. I tillegg kan man definere brukertilgang direkte i datastrukturen, slik at visse data kan visualiseres kun av enkelte brukere.

5.2 Drøfting

XML kan tross alt ikke løse alle problemene forbundet med integrering av informasjon på Internett. Noen av de problemene som fortsatt vil bestå er [17]:

- Det vil alltid finnes applikasjoner og web-områder som ikke ønsker å offentliggjøre den underliggende datastrukturen, og som ikke aktivt vil dele dataene med omverdenen. De vil derfor ikke oppgi noen DTD på sitt web-område, og selv om informasjonen publiseres på XML-format, vil man ikke nødvendigvis kjenne strukturen som dataene tilhører. For finanssektoren betyr det at de institusjonene som ikke ønsker å følge en eventuell DTD under publiseringen, vanskelig vil kunne delta som tilbydere i de nye informasjons-integrerende grensesnittene som er under oppvekst på Internett.
- HTML vil ikke opphøre å eksistere eller plutselig ikke bli brukt mer selv om XML blir mer anerkjent og benyttet. Mange vil fortsette å publisere sidene sine på HTML, mange bedrifter vil ikke oppdage nytten, eller se poenget med XML. Det kan for eksempel være store bedrifter som allerede har investert i dyre EDI-systemer, eller små bedrifter som ikke har råd til ny kompetanse på Internett-feltet.
- XML vil lette tilgangen til kilder innen spesifikke domener, men mellom ulike fagdomener er det vanskelig å tro at man skal kunne komme fram til en entydig struktur for hvordan informasjon skal modelleres. Ved at bedrifter av samme typen blir enige om en felles DTD seg imellom, vil disse kunne

utveksle informasjon, eller gjøre innhenting og integrering av informasjon fra disse bedriftene enkel. Men man kan vanskelig tenke seg at *alle* bedrifter, organisasjoner og enkeltpersoner vil kunne komme fram til en felles notasjon på alle felter. På denne måten vil de semantiske problemene forbundet med utveksling av informasjon bestå, selv om XML tas i bruk.

- Problemet med automatisk å kunne finne fram til, og integrere nye web-kilder vil bestå. XML gjør det lettere å identifisere elementer i kildene, og gjør det også enklere å vite hva en kilde presenterer og hvordan den gjør det. Men det blir ikke nødvendigvis enklere å *oppdage* kilden på Internett. Med andre ord, XML gjør ingenting med identifiseringsproblemet på Internett. Det løser kun deler av ekstraheringsproblemet og konverteringsproblemet.

På tross av disse problemene som ikke vil bli løst selv om man går over til å publisere informasjon på XML-format, mener jeg det vil lønne seg for banker og finansinstitusjoner å konvertere deres informasjonsinnhold på Internett til dette formatet. Behandlingen min av problemene viser at det er et stort behov for *struktur* i informasjonen. Strukturen er først og fremst viktig for at man kan utnytte på en bedre måte den informasjonen som allerede finnes, og legge grunnlag for nye verdiøkende tjenester i sektoren. Men det holder ikke at én bedrift standardiserer sitt informasjonsinnhold. Bedriftene må arbeide sammen for å skape „online networks“ av kunder, tilbydere og verdiøkende tjeneste [27]. Firmaene må lære å utvikle seg sammen med partnere for å skape gunstige vilkår for alle medvirkende parter, og da dreier det seg om å skape et felleskap hvor alle parter kan nyte godt av langvarige relasjoner med leverandører og kunder.

Når det gjelder dybden i det arbeidet jeg har gjennomført, vil det nok skinne igjennom at jeg ikke har en inngående kjennskap til finanssektoren. Muligens vil spesielt konsekvensene jeg ser, være begrenset av en manglende økonomisk bakgrunn. Likevel føler jeg at litteraturstudiene har gitt meg et tilstrekkelig overblikk og den forståelse som skal til for å besvare oppgaven.

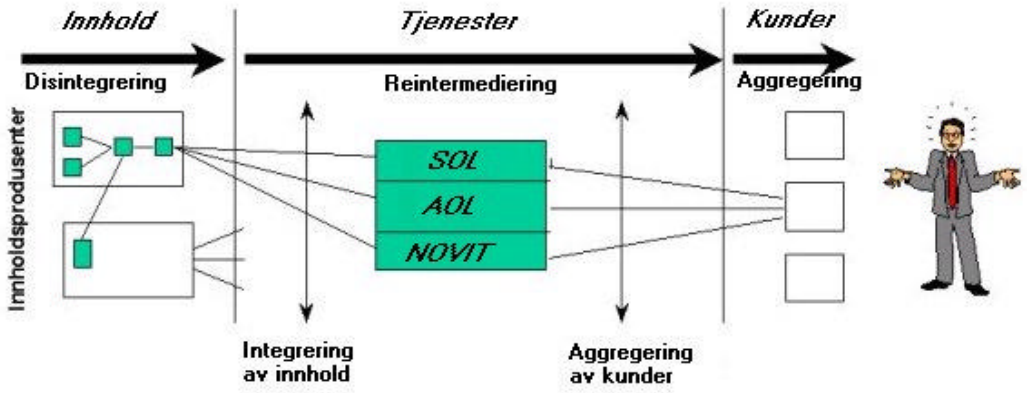
Videre hadde jeg i utgangspunktet kun generell kjennskap til de ulike forsknings-områdene innen informasjonsintegreringsproblemet. Dette har nok gjort at det overblikket jeg har kunnet skaffe meg i løpet av månedene arbeidet med hovedoppgaven har foregått, ikke er så nyansert som det jeg kunne ønske. Men også her tror jeg at en viss forståelse og et relativt oversiktlig bilde av

situasjonen er det viktigste for å kunne belyse problemene og løsningene i oppgaven.

5.3 Konsekvenser

Den tradisjonelle bankvirksomheten, slik vi kjenner den, er i endring. Det er ikke lenger penger som er primærobjektet i handelen, men derimot informasjon. I følge Lloyd Darlington [28] i Bank of Montreal Group of Companies defineres i dag en bank først og fremst på bakgrunn av dens evne til å tilby verdiøkende tjenester i kundeforholdene, noe som igjen kan brytes ned i det å tilegne seg, analysere, integrere og levere informasjon om, fra og til nytte for hver individuelle kunde. Det er blitt klart for bankene at kundesegmentet som handler finansielle produkter på Internett er stort og voksende, og dersom man ikke engasjerer seg i kampen om å vinne marked ved å følge den teknologiske utviklingen og ved å utvikle verdiøkende tjenester, kan man senere ha vanskelig for å hente inn det forspranget de andre har fått.

Alt dette må bygges inn i en ny strategimodell for finansinstitusjonene. I følge den konseptuelle forretningsmodellen for det virtuelle markeds-rommet (se figur 6) kan endringene i finansvirksomheten som følge av den digitale revolusjonen [27] sammenfattes som en *disintegrering* på innholdsleverandør-siden, en *reintermediering* i infrastrukturen og en *aggregering* på kunde-siden. Jeg ønsker her å se på hvilke konsekvenser man kan tenke seg at en omlegging til felles presentasjonsformat får for de ulike segmentene av den nye forretningsmodellen.



Figur 6: Konseptuell forretningsmodell. Etter [26]

Disintegreringen av tjenester på leverandørsiden har tradisjonelt ikke vært mye brukt i banktjenesten [26]. De nye kravene Internett-handelen stiller, forlanger imidlertid at bankene fokuserer mer på kunden, markedssegmenter og kunderelasjoner, og mindre på å beholde en integrert verdikjede. Her er outsourcing et virkemiddel. Et felles presentasjonsformat for finansinformasjonen ville her kunne gjøre det enklere å outsource elementer i produksjonen ved å la underleverandører behandle deler av informasjonsinnholdet. Et eksempel på dette kan være at en bank velger å la en underleverandør ta seg av kredittvurderingen på bankens vegne. Med et felles, underliggende format kan man så enkelt integrere den informasjonen som kommer fra underleverandører med bankens egen produksjon. Et annet eksempel er betydningen fellesformatet får for utviklingen av verdiøkende tjenester på produsent-siden. Man kan nå lettere tenke seg at underleverandører utvikler applikasjoner som for eksempel nettverkskalkulatorer eller rente-varslingssystemer, mens banken selv konsentrerer seg om sine mer tradisjonelle oppgaver. Fellesformatet gjør at bank og underleverandør opererer på det samme strukturgrunnlaget, uten å måtte konvertere informasjon eller foreta tilpasninger i systemene.

Reintermedieringen på Internett vil si at det vokser opp nye mellomledd, mens tradisjonelle roller forsvinner. Eksempler på nye mellomledd er tjenesteintegrerende systemer eller agenter som henter informasjon på vegne av en kunde. Denne typen mellomledd finnes allerede i dag, men de opplever vanskene med informasjonsintegreringsproblemet. Et felles presentasjonsformat ville her gjøre det lettere for disse systemene å samle informasjonen, integrere og presentere den. Med en felles DTD vil både kunder og applikasjoner kunne finne den informasjonen de søker etter, på kortere tid og med et sikrere resultat ved at informasjonen i kildene valideres mot den gyldige DTD‘en. XML gjør det så mulig å visualisere den integrerte informasjonen på ulike måter.

Nye mellomledd beyr også nye aktører, og man ser allerede eksempler på framvekst av firmaer som konsentrerer seg om et enkelt finansprodukt, som for eksempel lån til bil- eller huskjøp eller investering i ulike fond. XML gjør det enklere å opprette tilbud som dette, ved at de nye aktørene kan kjøpe deler av finansprodukter, og tilby dem direkte på nettet uten behov for større tilretteleggingner. Men for å forbli konkurransedyktige i sektoren, må også banker tilgjengeliggjøre deres informasjonsinnhold på best mulig måte, og i tillegg tilby

tjenester som kan gi kunden mer enn bare informasjon om for eksempel rentebetingelser. Dette kan man oppnå eksempelvis ved å tilpasse informasjonen den individuelle kunde, slik at kunden føler at det tas hensyn til personlige preferanser og kriterier. XML gjør det så mulig for kunden å velge, på en enkel måte, den informasjonen som ønskes vist; man kan sortere, filtrere eller foreta beregninger uten å foreta nye nedlastinger.

En *aggregering* på kundesiden vil si at kundene grupperer seg i felleskap hvor konsumentene får kontakt og diskuterer produkterfaringer eller betingelser. Disse gruppene kan for eksempel være en del av et tjenesteintegrerende system, hvor man oppfordrer kundene til å utveksle erfaringer gjennom gruppefelleskap. Kunder på Internett kan også regnes som tilbydere av verdiøkende tjenester i slike sammenhenger. Man kan for eksempel tenke seg diskusjonsgrupper eller fora som et interessant tilbud for å tiltrekke andre kunder. Dersom informasjonen som kundene sitter inne med, og som de ønsker å utveksle, er på samme format, gjør dette også her en utveksling av informasjon enklere.

Det å strukturere finansinformasjon på fellesformat vil altså ha konsekvenser for alle segmentene i den nye konseptuelle forretningsmodellen for markedsrommet. Det går fram av konsekvensene jeg presenterer over, hvordan et felles informasjonsformat forenkler informasjonsflyten både mellom bedrifter, mellom bedrift og kunde, og kunder seg imellom.

I [27] påpekes det at det nå er informasjon som er finansindustriens viktigste virkemiddel; „We still handle money, but information, not money, is now the lifeblood of our industry“. Dette verdibildet bør innarbeides hos aktørene i sektoren for at de skal overleve den digitale revolusjonen. Samtidig heter det videre i [28] at „hoarding knowledge (as opposed to land, goods, or capital) is typically counterproductive and nearly impossible; in the digital economy, knowledge must be shared“. Jeg synes disse to utsagnene, sammen med konsekvensene jeg har beskrevet, på en grei måte begrunner både behovet for - og mulighetene med - et felles representasjonsformat og felles struktur på informasjonsinnholdet til banker og finansinstitusjoner.

Referanser

1. Guttman, R.H., .Moukas, A.G., Maes, P.: *Agent-mediated Electronic Commerce: A Survey*, To appear, Knowledge Engineering Review, June 1998
2. Methlie, L.B. (1997). *Teleøkonomi. Et forretningskonsept for multimedia banking –en forstudie*, SNF-rapport, Juni 1997
3. Troye, S.V., Grønhaug, K. (1993): *Hvordan skrive en utredning til glede for både deg selv og andre*, Tano A/S, 1993
4. Moukas, A., Guttman, R., Maes, P.: *Agent-mediated Electronic Commerce: An MIT Media Laboratory Perspective*, Proceedings of the International Conference on Electronic Commerce (ICEC'98), Seoul, Korea, April 1998
5. Gruser, J.R., Raschid, L., Vidal, M.E., Bright, L.: *Wrapper Generation for Web Accessible Data Sources*, In Proceedings CoopIS'98
6. Soderland, S. (1997): *Learning to Extract Text-based Information from the World Wide Web*, To appear in Proceedings of Third International Conference on Knowledge Discovery and Data Mining (KDD-97)
7. Mecca, G., Merialdo, P., Atzeni, P. (1998): *Do we really need a new query language for XML?*, [Online] http://w3c.bilkent.edu.tr/TandS/QL/QL98/pp/MeMAql98.html
8. Tidwell, D. (1999): *Building an XML application*, [Online] http://www.software.ibm.com/xml/education/buildappl/abstract.html
9. Bosak, J. (1997): *XML, Java and the future of the Web*, [Online] http://metalab.unc.edu/pub/sun-info/standards/xml/why/xmlapps.htm
10. Petrie, C.J. (1996): *Agent-Based Engineering, the Web, and Intelligence*, IEEE Expert, 11:6, pp. 24-29, December, 1996
11. Grosof, B.N. (1997): *Building Commercial Agents: An IBM Research Perspective (Invited Talk)*, Proceedings of the Second International Conference on The Practical Applications of Intelligent Agents and Multi-Agent Technology (PAAM97), UK, April 21-23, 1997
12. Bright, L., Gruser, J.R., Raschid, L., Vidal, M.E.: *Wrapper Generation toolkit to specify and construct Wrappers for Web Accessible Data Sources (WebSources)*, In Proceedings CoopIS'98
13. Ambler, S.W. (1997): *An Object-Oriented Architecture for Buisness-To-Consumer Electronic Commerce On The Internet*, [Online] http://www.ambysoft.com/eCommerceArchitecture.html
14. Ambler, S.W. (1998): *The Unified Modeling Language v1.1 and Beyond: The Techniques of Object-Oriented Modeling* [Online] http://www.ambysoft.com/umlAndBeyond.html
15. Booch, G., Rumbaugh, J., Jacobsen, I. (1998): *The UML User Guide*, An Imprint of Addison Wesley Langman. Inc.
16. Levy, A.Y.: *The Information Manifold approach to data integration*, Marti A. Hearst. Information Integration. IEEE Intelligent Systems, Vol. 13, No. 5, September/October 1998
17. Knoblock, C.A., Minton, S.: *The Ariadne approach to Web-based information integration*, Marti A. Hearst. Information Integration. IEEE Intelligent Systems, Vol. 13, No. 5, September/October 1998
18. Cohen, W.W.: *The Whirl approach to information integration:* Marti A. Hearst. Information Integration. IEEE Intelligent Systems, Vol. 13, No. 5, September/October 1998
19. Huck, G., Frankhauser, P., Aberer, K., Neuhold, E.: *Jedi: Extracting and Synthesizing Information from the Web*, Proceedings of the 3rd IFCIS International Conference on Cooperative Information Systems (CoopIS'98), New York City, August 1998
20. Sahuguet, A., Azavant, F.: *Web Ecology: Recycling HTML pages as XML documents using W4F* [Online] http://db.cis.upenn.edu/W4F/publi.html

21. Ashish, N., Knoblock, C.A.: *Semi-automatic Wrapper Generation for Internet Information Sources*, Second IFCIS Conference on Cooperative Information Systems (CoopIS), Charleston, South Carolina, 1997
22. Pedersen, P.E., (1999): *En agentbasert tjeneste for produkt- og leverandørsammenligninger av finanstjenester*, SNF Arbeidsnotat, Bergen: Stiftelsen for Handel og Næringslivsforskning
23. Maes, P., Guttman, R.H., Moukas, A.G. (1999): *Agents that Buy and Sell: Transforming Commerce as we Know It*, To appear, Communications of the ACM, special issue on agents and electronic commerce
24. Hammer, J., Garcia-Molina, H., Cho, J., Aranha, R., Crespo, A. (199x): *Extracting Semistructured Information from the Web*, Proceedings of the Workshop on Management of Semistructured Data. Tucson, Arizona, Mai 1997
25. Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J., Widom, J.: *Integrating and Accessing Heterogenous Information Sources in TSIMMIS*, Proceedings of the AAAI Symposium on Information Gathering, pp. 61-64, Stanford, California, Mars 1995
26. Methlie, L.B. (1998): *A buisness model for electronic commerce*. Working paper. Bergen, Norway: Norwegian School of Economics and Buisness Administration
27. Ticoll, D., Lowy, A. (1998). *Joined at the Bit. The Emergence of the E-buisness Community*, I Tapscott, D., Lowy, A. & Ticoll, D. (Eds.). Blueprint to the digital economy, s. 21-33. New York: McGraw-Hill.
28. Darlington, L. (1998). *Banking without boundaries. How the banking industry is transforming itself for the digital age*. I Tapscott, D., Lowy, A. & Ticoll, D. (Eds.). Blueprint to the digital economy, s. 113-138. New York: McGraw-Hill.
29. Soderland, S., Fisher, D., Aseltin, J., Lehnert, W.(1995): *CRYSTAL: Inducing a Conceptual Dictionary* [Online] http://ciir.cs.umass.edu/info/psfiles/tepubs/tepubs.html
30. Methlie, L.B., Pedersen, P.E. (1998): *Aktører og agenter i elektroniske markeder*, Presentasjon ved Nasjonalt seminar om samarbeids- og agentteknologi i Tromsø 3. og 4. des. 1998
31. Pedersen P.E. (1998). *Agentprosjektet*, Presentasjon til prosjektmøte ved SNF 7. des. 1998
32. Wiley, L.D. (1998): *Beyond Information Retrieval: Ways to Provide Content in Context*, [Online] http://www.onlineinc.com/database/DB1998/wiley8.html
33. Bray, T.(1999): *Beyond HTML: XML and Automated Web Processing*, [Online] http://developer.netscape.com/viewsource/bray_xml.html
34. Ambler, S.W. (1998): *How the UML Models Fit Together*, [Online] http://www.sdmagazine.com/uml/focus.ambler.htm
35. Andersson, M., Bergstrand, J.(1995): *Formalizing Use Cases with Message Sequence Charts*, [Online] http://www.efd.lth.se/~d87man/EXJOBB/UseCases_in_OOA.html

6 Vedlegg

Vedlegg 1: Tabell over eksisterende teknologi i finanssektoren

Web-område	**Underliggende teknikk**
www.loanshop.com / www.mortgage.com	Blant de første som svarte på mine forespørsler var www.mortgage.com eller www.loanshop.com som er det samme firmaet. Mortgage.com benytter Windows NT 4.0 og Internet Information Server 4.0, samt en Microsoft SQL Server 6.5 database. Sidene er utviklet ved hjelp av Microsoft Interdev som er et utviklingsverktøy spesielt egnet til å kombinere scripting og HTML med SQL server konfigurasjon og forvaltning. Mortgage.com baserer seg på fem fundamentale konsepter: 1. Websidene bygges dynamisk basert på data som er lagret i tabeller på SQL serveren. Fordelen ved dette er at de samme dataene kan brukes flere steder og endringer i dataene kan gjøres effektivt. Alle data om brukerne lagres i databasen slik at hver individuelle bruker kan gjenkjennes. 2. Ettersom data om brukeren fanges, blir disse dataene benyttet til å gjøre websidene personlige. 3. Webområdet er selv-administrerende, dvs. at personer med den rette autorisasjonen (eksempelvis låntilbydere) kan gå inn og editere låneprogrammene sine eller rentene de opererer med. 4. Webområdet tilbyr one-click-access slik at brukerne med ett klikk kan få nærliggende opplysninger som lånerenter eller hjelp. 5. Websidene er laget med tanke på handel. Det vil si at de opererer med et Learn-Choose-Act konsept hvor brukeren blir guidet gjennom hele prosessen. Sidene oppdateres med daglig informasjon. Informasjonen blir publisert fra finansinstitusjonene i form av et Excel spreadsheet via elektronisk mail. Ved Mortgage.com ekstraheres ratene fra spreadsheetet og sendes via en ny e-mail til en spesiell mailbox. Derfra går dataene automatisk inn i SQL serveren hvor de kan aksesseres av web-brukerne. I tillegg finnes et brukergrensesnitt hvor data kan legges inn manuelt.
www.microsurf.com	
www.hsh.com	
www.interest.com / www.mmis.com	Dette webområdet er avhengige av at låntilbyderne faxer inn deres tall hvor de deretter tastes inn manuelt i en database.
www.QuickenMortgage.com	
www.bankrate.com	Bankrate ringer hver dag opp bankene for å få med siste endringer i ratene.

www.theloanpage.com	
www.homeshark.com	Sier de ikke har kapasitet til å svare på spørsmålet mitt, men sender en liste over alle kildene deres.
www.mortgagecafe.com	Ville først ikke ut med hvordan de foretar datainnsamlingen på grunn av konkurranse-trusselen. Jeg skrev tilbake og spurte om de kunne indikere om de benyttet agentteknologi eller tradisjonelle metoder. Svaret var at de ikke benytter spiders og crawlers på dette tidspunktet.
www.rate.net.com	Har spurt om mer info om punkt 4. Rate.Net opererer med fem forskjellige prosedyrer for informasjonsinnhentingen: 1) Telefon 2) Fax 3) Email 4) Robots eller Spiders 5) Remote data entry 1. Markedsundersøkere ringer opp forskjellige markeder og samler inn data som manuelt legges inn i en database 2. Automatiske faxer sendes ut hver dag til forskjellige finans-institusjoner med de aktuelle ratene og dataene. Dersom disse dataene trenger oppdatering gjøres dette, før faxene sendes tilbake. Her blir den oppdaterte informasjonen manuelt lagt inn i en database. 3. Automatiske e-mail sendes ut til forskjellige institusjoner som har tilgang til e-mail, hvor informasjonen studeres av de aktuelle personene som oppdaterer dataene og sender så mailen tilbake. På rate.net sida er denne prosessen automatisert. De returnerte e-mailene blir analysert, feilsjekket og automatisk lagt inn i en database. 4. Noen finansinstitusjoner har webområder hvor de viser rater. Rate.net har roboter som drar rundt til disse områdene, samler inn data og rater og oppdaterer databasen. Rate.net oppgir dette som en god ide med god programmering, men veldig få av bankenes webområder er pålitelige nok til å tilby denne typen informasjon fordi de ikke oppdateres på regelmessig basis. 5. Remote data entry er en av favorittmekanismene til rate.net, hvor finansinstitusjonene logger på en av websidene deres og oppdaterer deres informasjon direkte.
www.keystrokenet.com	
www.banx.com	Vil ikke ut med hvordan de foretar datainnsamlingen på grunn av konkurranse-trusselen.
www.amo-mortgage.com	
www.loanpage.com	

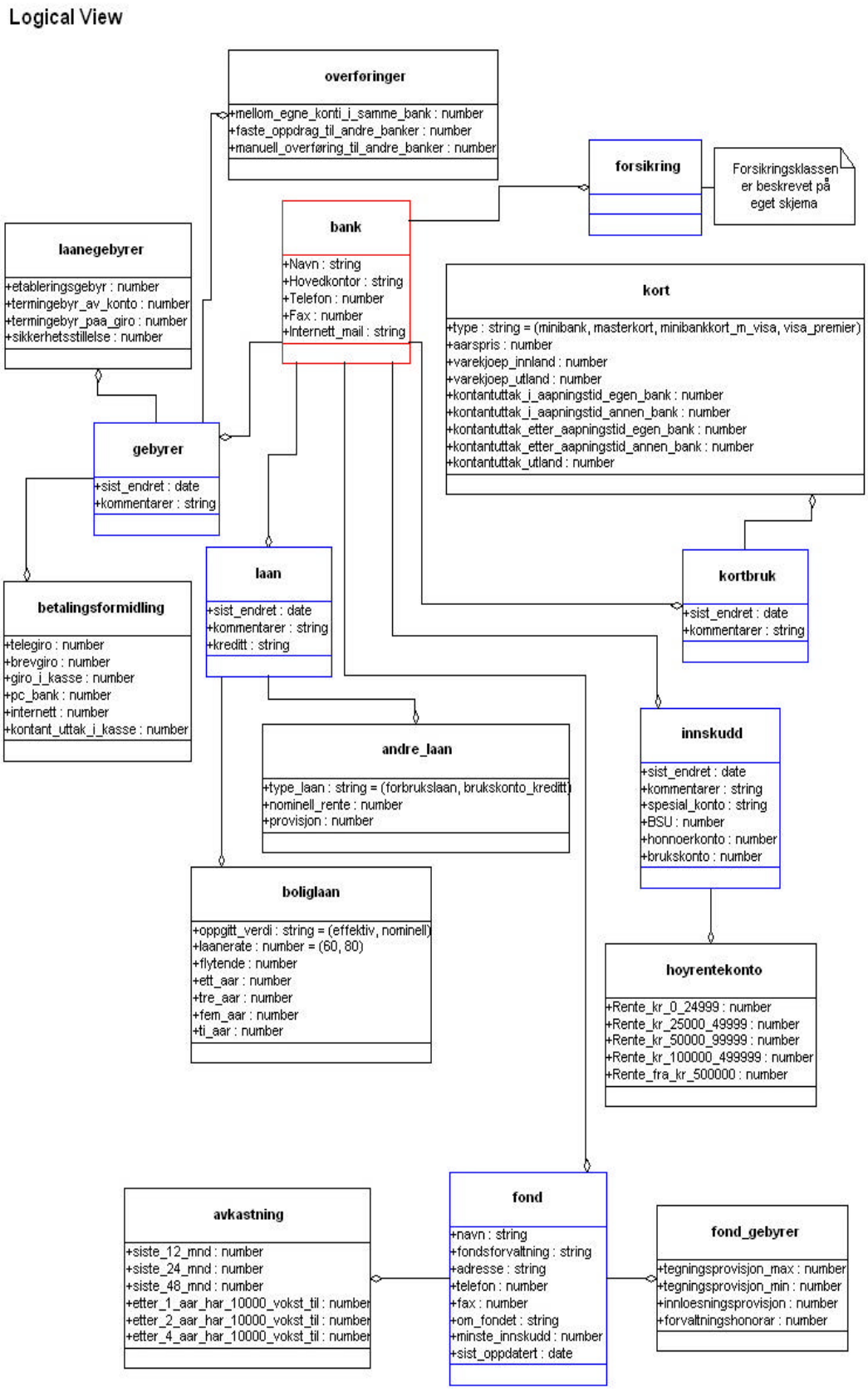
www.BancOneMortgage.com	Oppgir at de ikke har den informasjonen jeg spør etter.
www.mortgage-net.com / Myers Internet Services (www.Myers.com)	
www.FannieMae.com	
www.NFNS.com	NFSN står bak MortgageCafe.com også, og svaret er derfor det samme: Bruker ikke spiders eller crawlers.
www.mortgage-x.com	
www.mortgage101.com / IMARK Design Group (www.i-mark-eting.com)	
www.FreddieMac.com	
www.homefair.com	Låntilbyderne publiserer sine rater på webområdene deres på et gitt format. Et program på Homefairs' server finner disse sidene, ekstraherer ratene og lagrer dem i en database. Formatet som benyttes for bankene er en komma-separert fil med et innholdsformat som Homefair spesifiserer.
www.LendingTree.com	
www.themortgage.com	
www.countrywide.com	
www.E-Loan.com	Vil ikke ut med hvordan de foretar datainnsamlingen på grunn av konkurranse-trusselen. Har skrevet tilbake og spurt om de kan indikere om de benytter agenter eller tradisjonelle metoder. Svarer tilbake at de mottar informasjon fra låntilbyderne på elektronisk vis (antar e-mail), og at de har en automatisert prosedyre som daglig laster disse ratene inn i deres database slik at de kan finnes på websiden deres.
www.InterestRates.com	
www.MortgageStore.com	
www.yourmortgage.com	Informasjonen samles og oppdateres på flere forskjellige måter. Dette inkluderer telefonoppringninger til institusjonene og faxer med informasjonen tilbake til institusjonene slik at de kan bekrefte riktigheten av ratene. I tillegg kontrollerer de låntilbydernes websider. Samtidig ber de låntilbyderne om å kontrollere yourmortgage's sine websider og maile, faxe eller ringe om eventuelle endringer. De sier at de alltid har planlagt et online system som kan aksesseres via nettet som tillater et online finansielt system (antar de mener at de finansielle institusjonene selv kan oppdatere egen informasjon direkte på websiden), dette har imidlertid blitt utsatt.
www.themortgagenetwork.com	The Mortgage Network opererer med manualer fra alle lånetilbyderne som er tilknyttet dem som beskriver produktene og rettningslinjene deres. Hver morgen sender disse låntilbyderne faxer med lister over dagens rater. Ved The Mortgage Network går de så igjennom alle ratelistene og finner de beste ratene for et bestemt

	låneprodukt. Denne informasjonen blir så publisert på websidene.
www.loanz.com	Opererer gjennom et underliggende firma Laser Rates, som mottar faxer med rater fra alle låntilbyderne som Loanz.com benytter. Disse ratene legges manuelt inn i Laser Rates' sitt system. Deretter sendes ratene til Loanz.com sitt web-område. Oppgir at teknologien er mer imponerende etter at informasjonsinnhentingen er foretatt; den tillater kunden å lete etter det beste tilbudet fra mer enn 200 låntilbydere.
www.CostalMortgage.com	
www.CanadaMortgage.com	Ringer opp bankene og spør dem om endringer i ratene.
Www.getsmart.com	Trenger mere tid før de kan svare på spørsmålene mine.
www.canadamortgage.com	
www.mdbanks.com	
www.CrossroadsUSA.com	
www.accordmortgage.com	
www.mortgagebase.com	
www.loansurfer.com	

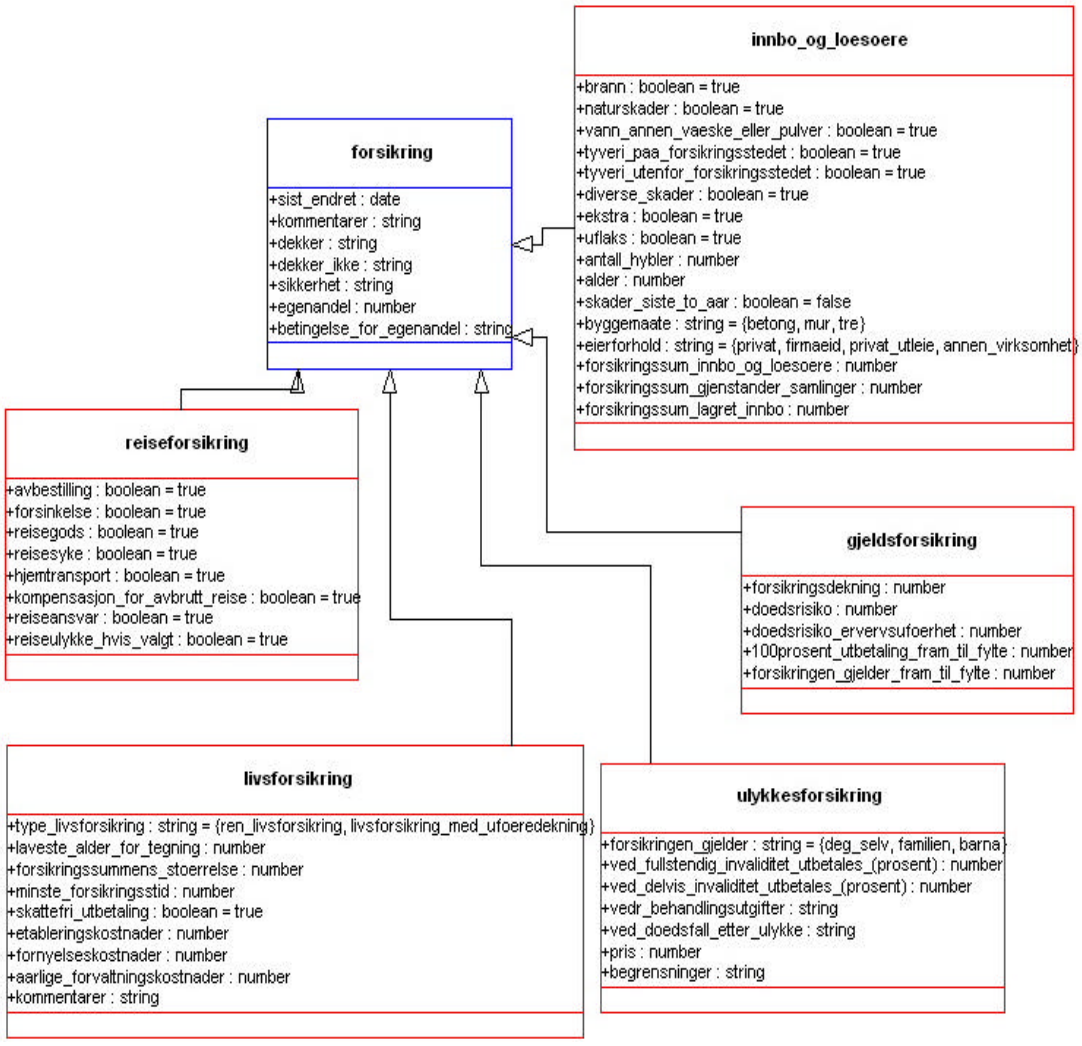
Vedlegg 2: Avhengighetsforholdene blant informasjons-leverandørene

Web-område	Benyttes av
www.loanshop.com / www.mortgage.com	▪ www.Credit.com
www.microsurf.com	▪ www.Lycos.com
www.hsh.com	▪ www.bankrates.com
www.interest.com / www.mmis.com	▪ www.CenturyOak.com ▪ www.moneypage.com
www.QuickenMortgage.com	▪ www.CNNfn.com, (CNN Financial Network (CNNfn), ▪ www.AOL.com, ▪ Excite Money & Investing
www.bankrate.com	▪ www.QuickenMortgage.com ▪ www.Infoseek.com benytter www.Realtor.com som er linket til Bank Rate Monitor (www.bankrate.com) ▪ Yahoo!Finance ▪ www.InfoSpace.com
www.homeshark.com	▪ www.hotbot.com ▪ www.owners.com
www.banx.com	▪ www.Bloomberg.com
www.mortgage101.com / IMARK Design Group (www.i-mark-eting.com)	▪ www.AustinMortgage.com, ▪ www.amo-Mortgage.com, ▪ www.BCMortgage.com, ▪ www.CherylGlober.com
www.LendingTree.com	▪ www.HomeNet.com ▪ www.ourfamilyplace.com ▪ money.go.com
www.E-Loan.com	▪ www.dljdirect.com ▪ www.etrade.com ▪ www.owners.com
www.getsmart.com	▪ www.angelfire.com

Vedlegg 3: UML skjema m/forsikringsdelen



Logical View



Vedlegg 4: Document Type Definition (DTD)

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

  <!ELEMENT Informasjon (Bank)*>
  <!ELEMENT Bank (innskudd, laan, fond?, kortbruk, gebyrer)>
  <!ATTLIST Bank      Navn CDATA #REQUIRED
                      Hovedkontor CDATA #REQUIRED
                      Telefon CDATA #IMPLIED
                      Fax CDATA #IMPLIED
                      Internett_mail CDATA #IMPLIED>
  <!ELEMENT innskudd (brukskonto, honnoerkonto, BSU, hoeyrentekonto)>
  <!ATTLIST innskudd  sist_endret CDATA #IMPLIED
                      kommentarer CDATA #IMPLIED
                      spesial_konto CDATA #IMPLIED>
  <!ELEMENT brukskonto (#PCDATA)>
  <!ELEMENT honnoerkonto (#PCDATA)>
  <!ELEMENT BSU (#PCDATA)>
  <!ELEMENT hoeyrentekonto (Rente_kr_0_24999?, Rente_kr_25000_49999?, Rente_kr_25000_49999?,
Rente_kr_100000_499999?, Rente_fra_kr_500000?)>
  <!ELEMENT Rente_kr_0_24999 (#PCDATA)>
  <!ELEMENT Rente_kr_25000_49999 (#PCDATA)>
  <!ELEMENT Rente_kr_50000_99999 (#PCDATA)>
  <!ELEMENT Rente_kr_100000_499999 (#PCDATA)>
  <!ELEMENT Rente_fra_kr_500000 (#PCDATA)>
  <!ELEMENT laan (boliglaan, andre_laan)>
  <!ATTLIST laan      sist_endret CDATA #IMPLIED
                      kommentarer CDATA #IMPLIED
                      kreditt CDATA #IMPLIED>
  <!ELEMENT boliglaan (effektiv, nominell)>
  <!ELEMENT effektiv (flytende, ett_aar, tre_aar, fem_aar, ti_aar)*>
  <!ATTLIST effektiv  laanerate (60 | 80) "60">
  <!ELEMENT nominell (flytende, ett_aar, tre_aar, fem_aar, ti_aar)*>
  <!ATTLIST nominell  laanerate (60 | 80) "60">
  <!ELEMENT flytende (#PCDATA)>
  <!ELEMENT ett_aar (#PCDATA)>
  <!ELEMENT tre_aar (#PCDATA)>
  <!ELEMENT fem_aar (#PCDATA)>
  <!ELEMENT ti_aar (#PCDATA)>
  <!ELEMENT andre_laan (forbrukslaan, brukskonto_kreditt)>
  <!ELEMENT forbrukslaan (nominell_rente?, provisjon?)>
  <!ELEMENT brukskonto_kreditt (nominell_rente?, provisjon?)>
  <!ELEMENT nominell_rente (#PCDATA)>
  <!ELEMENT provisjon (#PCDATA)>
  <!ELEMENT fond (avkastning, fond_gebyrer)>
  <!ATTLIST fond      navn CDATA #REQUIRED
                      fondsforvaltning CDATA #REQUIRED
                      addresse CDATA #REQUIRED
                      telefon CDATA #IMPLIED
                      sist_oppdatert CDATA #IMPLIED>
  <!ELEMENT avkastning (siste_12_mnd?, siste_24_mnd?, siste_48_mnd?,
etter_1_aar_har_10000_vokst_til?, etter_2_aar_har_10000_vokst_til?,
etter_4_aar_har_10000_vokst_til?)>
  <!ELEMENT siste_12_mnd (#PCDATA)>
  <!ELEMENT siste_24_mnd (#PCDATA)>
  <!ELEMENT siste_48_mnd (#PCDATA)>
  <!ELEMENT etter_1_aar_har_10000_vokst_til (#PCDATA)>
  <!ELEMENT etter_2_aar_har_10000_vokst_til (#PCDATA)>
  <!ELEMENT etter_4_aar_har_10000_vokst_til (#PCDATA)>
  <!ELEMENT fond_gebyrer (tegningsprovisjon_max?, tegningsprovisjon_min?,
innloesningsprovisjon?, forvaltningshonorar?)>
  <!ELEMENT tegningsprovisjon_max (#PCDATA)>
  <!ELEMENT tegningsprovisjon_min (#PCDATA)>
  <!ELEMENT innloesningsprovisjon (#PCDATA)>
  <!ELEMENT forvaltningshonorar (#PCDATA)>
<!ELEMENT kortbruk (kort?)>
  <!ELEMENT kort (aarspris?, varekjoep_innland?, varekjoep_utland?,
kontantuttak_i_aapningstid_egen_bank?, kontantuttak_i_aapningstid_annen_bank?,
kontantuttak_etter_aapningstid_egen_bank?, kontantuttak_etter_aapningstid_annen_bank?,
```

```
kontantuttak_utland?)*>
  <!ATTLIST kort      type (minibank | masterkort | minibankkort_m_visa | visa_premier)
"minibank"
                      sist_endret CDATA #IMPLIED
                      kommentarer CDATA #IMPLIED>
  <!ELEMENT aarspris (#PCDATA)>
  <!ELEMENT varekjoep_innland (#PCDATA)>
  <!ELEMENT varekjoep_utland (#PCDATA)>
  <!ELEMENT kontantuttak_i_aapningstid_egen_bank (#PCDATA)>
  <!ELEMENT kontantuttak_i_aapningstid_annen_bank (#PCDATA)>
  <!ELEMENT kontantuttak_etter_aapningstid_egen_bank (#PCDATA)>
  <!ELEMENT kontantuttak_etter_aapningstid_annen_bank (#PCDATA)>
  <!ELEMENT kontantuttak_utland (#PCDATA)>
  <!ELEMENT gebyrer (laanegebyrer, betalingsformidling, overfoeringer)>
  <!ATTLIST gebyrer   sist_endret CDATA #IMPLIED
                      kommentarer CDATA #IMPLIED>
  <!ELEMENT laanegebyrer (etableringsgebyr?, termingebyr_av_konto?, termingebyr_paa_giro?,
sikkerhetsstillelse?)>
  <!ELEMENT etableringsgebyr (#PCDATA)>
  <!ELEMENT termingebyr_av_konto (#PCDATA)>
  <!ELEMENT termingebyr_paa_giro (#PCDATA)>
  <!ELEMENT sikkerhetsstillelse (#PCDATA)>
  <!ELEMENT betalingsformidling (telegiro?, brevgiro?, giro_i_kasse?, pc_bank?, internett?,
kontant_uttak_i_kasse?)>
  <!ELEMENT telegiro (#PCDATA)>
  <!ELEMENT brevgiro (#PCDATA)>
  <!ELEMENT giro_i_kasse (#PCDATA)>
  <!ELEMENT pc_bank (#PCDATA)>
  <!ELEMENT internett (#PCDATA)>
  <!ELEMENT kontant_uttak_i_kasse (#PCDATA)>
  <!ELEMENT overfoeringer (mellom_egne_konti_i_samme_bank?, faste_oppdrag_til_andre_banker?,
manuell_overfoering_til_andre_banker?)>
  <!ELEMENT mellom_egne_konti_i_samme_bank (#PCDATA)>
  <!ELEMENT faste_oppdrag_til_andre_banker (#PCDATA)>
  <!ELEMENT manuell_overfoering_til_andre_banker (#PCDATA)>
```

Vedlegg 5: XML-dokumentet

```
<?xml version="1.0"?>
<!DOCTYPE Banker SYSTEM "dtdmatt.dtd">

<Informasjon>
 <Bank>
  <Navn>Gjensidige Bank</Navn>
  <Hovedkontor>Oslo</Hovedkontor>
  <Telefon>55555</Telefon>
  <Fax>55555</Fax>
  <Internett_mail>www.gjensidige.no</Internett_mail>
  <innskudd>
   <brukskonto>0.027</brukskonto>
   <honnoerkonto>0.0</honnoerkonto>
   <BSU>0.067</BSU>
   <hoeyrentekonto>
    <Rente_kr_0_24999>0.0625</Rente_kr_0_24999>
    <Rente_kr_25000_49999>0.0625</Rente_kr_25000_49999>
    <Rente_kr_50000_99999>0.0625</Rente_kr_50000_99999>
    <Rente_kr_100000_499999>0.067</Rente_kr_100000_499999>
    <Rente_fra_kr_500000>0.068</Rente_fra_kr_500000>
   </hoeyrentekonto>
   <sist_endret>01.04.99</sist_endret>
   <kommentarer>Hva som helst</kommentarer>
   <spesial_konto>nytt</spesial_konto>
  </innskudd>
  <laan>
   <boliglaan>
    <effektiv laanerate="60">
     <flytende>0.0825</flytende>
     <ett_aar>0.0825</ett_aar>
     <tre_aar>0.0825</tre_aar>
     <fem_aar>0.0825</fem_aar>
     <ti_aar>0.0825</ti_aar>
    </effektiv>
    <effektiv laanerate="80">
     <flytende>0.0884</flytende>
     <ett_aar>0.0884</ett_aar>
     <tre_aar>0.0884</tre_aar>
     <fem_aar>0.0884</fem_aar>
     <ti_aar>0.0884</ti_aar>
    </effektiv>
    <nominell laanerate="60">
     <flytende>0.0795</flytende>
     <ett_aar>0.0795</ett_aar>
     <tre_aar>0.0795</tre_aar>
     <fem_aar>0.0795</fem_aar>
     <ti_aar>0.0795</ti_aar>
    </nominell>
    <nominell laanerate="80">
     <flytende>0.085</flytende>
     <ett_aar>0.085</ett_aar>
     <tre_aar>0.085</tre_aar>
     <fem_aar>0.085</fem_aar>
     <ti_aar>0.085</ti_aar>
    </nominell>
   </boliglaan>
   <andre_laan>
    <forbrukslaan>
     <nominell_rente>0.0</nominell_rente>
     <provisjon>0.0</provisjon>
    </forbrukslaan>
    <brukskonto_kreditt>
     <nominell_rente>0.1595</nominell_rente>
     <provisjon>0.0</provisjon>
    </brukskonto_kreditt>
   </andre_laan>
   <sist_endret>09.03.99</sist_endret>
   <kommentarer>Ingen</kommentarer>
   <kreditt>nytt</kreditt>
  </laan>
```

```
<fond>
 <navn>Vesta</navn>
 <fondsforvaltning>Vesta</fondsforvaltning>
 <addresse>Ukjent</addresse>
 <telefon>222222</telefon>
 <fax>22222</fax>
 <om_fondet>Spesielle kommentarer</om_fondet>
 <minste_innskudd>2.000.000</minste_innskudd>
 <avkastning>
  <siste_12_mnd>1.0</siste_12_mnd>
  <siste_24_mnd>1.0</siste_24_mnd>
  <siste_48_mnd>1.0</siste_48_mnd>
  <etter_1_aar_har_10000_vokst_til>1.0</etter_1_aar_har_10000_vokst_til>
  <etter_2_aar_har_10000_vokst_til>1.0</etter_2_aar_har_10000_vokst_til>
  <etter_4_aar_har_10000_vokst_til>1.0</etter_4_aar_har_10000_vokst_til>
 </avkastning>
 <fond_gebyrer>
  <tegningsprovisjon_max>1.0</tegningsprovisjon_max>
  <tegningsprovisjon_min>1.0</tegningsprovisjon_min>
  <innloesningsprovisjon>1.0</innloesningsprovisjon>
  <forvaltningshonorar>1.0</forvaltningshonorar>
 </fond_gebyrer>
 <sist_oppdatert>12.03.98</sist_oppdatert>
</fond>
<kortbruk>
 <kort type="minibank">
  <aarspris>200.0</aarspris>
  <varekjoep_innland>1.0</varekjoep_innland>
  <varekjoep_utland>0.0</varekjoep_utland>
   <kortbruk>
 <kort type="minibank">
  <aarspris>200.0</aarspris>
  <varekjoep_innland>1.0</varekjoep_innland>
  <varekjoep_utland>0.0</varekjoep_utland>
  <kontantuttak_i_aapningstid_egen_bank>2.0</kontantuttak_i_aapningstid_egen_bank>
  <kontantuttak_i_aapningstid_annen_bank>2.0</kontantuttak_i_aapningstid_annen_bank>
  <kontantuttak_e_aapningstid_egen_bank>5.0</kontantuttak_e_aapningstid_egen_bank>
  <kontantuttak_e_aapningstid_annen_bank>5.0</kontantuttak_e_aapningstid_annen_bank>
  <kontantuttak_utland>0.0</kontantuttak_utland>
 </kort>
 <kort type="masterkort">
  <aarspris>175.0</aarspris>
  <varekjoep_innland>2.0</varekjoep_innland>
  <varekjoep_utland>0.0</varekjoep_utland>
  <kontantuttak_i_aapningstid_egen_bank>0.0</kontantuttak_i_aapningstid_egen_bank>
  <kontantuttak_i_aapningstid_annen_bank>3.0</kontantuttak_i_aapningstid_annen_bank>
  <kontantuttak_e_aapningstid_egen_bank>0.0</kontantuttak_e_aapningstid_egen_bank>
  <kontantuttak_e_aapningstid_annen_bank>3.0</kontantuttak_e_aapningstid_annen_bank>
  <kontantuttak_utland>0.0</kontantuttak_utland>
 </kort>
 <kort type="minibankkort_m_visa">
  <aarspris>0.0</aarspris>
  <varekjoep_innland>0.0</varekjoep_innland>
  <varekjoep_utland>0.0</varekjoep_utland>
  <kontantuttak_i_aapningstid_egen_bank>0.0</kontantuttak_i_aapningstid_egen_bank>
  <kontantuttak_i_aapningstid_annen_bank>0.0</kontantuttak_i_aapningstid_annen_bank>
  <kontantuttak_e_aapningstid_egen_bank>0.0</kontantuttak_e_aapningstid_egen_bank>
  <kontantuttak_e_aapningstid_annen_bank>0.0</kontantuttak_e_aapningstid_annen_bank>
  <kontantuttak_utland>0.0</kontantuttak_utland>
 </kort>
 <kort type="visa_premier">
  <aarspris>200.0</aarspris>
  <varekjoep_innland>2.0</varekjoep_innland>
  <varekjoep_utland>3.0</varekjoep_utland>
  <kontantuttak_i_aapningstid_egen_bank>0.0</kontantuttak_i_aapningstid_egen_bank>
  <kontantuttak_i_aapningstid_annen_bank>3.0</kontantuttak_i_aapningstid_annen_bank>
  <kontantuttak_e_aapningstid_egen_bank>0.0</kontantuttak_e_aapningstid_egen_bank>
  <kontantuttak_e_aapningstid_annen_bank>3.0</kontantuttak_e_aapningstid_annen_bank>
  <kontantuttak_utland>27.0</kontantuttak_utland>
 </kort>
```

```
   <sist_endret>12.03.98</sist_endret>
   <kommentarer>Ingen</kommentarer>
  </kortbruk>
  <gebyrer>
   <laanegebyrer>
    <etableringsgebyr>0.0</etableringsgebyr>
    <termingebyr_av_konto>0.0</termingebyr_av_konto>
    <termingebyr_paa_giro>50.0</termingebyr_paa_giro>
    <sikkerhetsstillelse>0.0</sikkerhetsstillelse>
   </laanegebyrer>
   <betalingsformidling>
    <telegiro>2.0</telegiro>
    <brevgiro>4.0</brevgiro>
    <giro_i_kasse>25.0</giro_i_kasse>
    <pc_bank>0.0</pc_bank>
    <internett>2.0</internett>
    <kontant_uttak_i_kasse>0.0</kontant_uttak_i_kasse>
   </betalingsformidling>
   <overfoeringer>
    <mellom_egne_konti_i_samme_bank>0.0</mellom_egne_konti_i_samme_bank>
    <faste_oppdrag_til_andre_banker>15.0</faste_oppdrag_til_andre_banker>
    <manuell_overfoering_til_andre_banker>10.0</manuell_overfoering_til_andre_banker>
   </overfoeringer>
   <sist_endret>10.03.99</sist_endret>
   <kommentarer>Ingen</kommentarer>
  </gebyrer>
 </Bank>
</Informasjon>
```

Vedlegg 6: DB2XML-dokumentet

```
 <?xml version="1.0" encoding="UTF-8"?>
<Banker>
 <samling>
  <informasjon>
   <BankID>1.0</BankID>
   <Bank>Sparebank 1 Eiker Drammen</Bank>
   <Web>http://nettbank.fellesdata.no/hb-doc/2220/</Web>
   <LÃ¥nEndret>09.03.99</LÃ¥nEndret>
   <LÃ¥n60Nom>0.079</LÃ¥n60Nom>
   <LÃ¥n60Eff>0.0831</LÃ¥n60Eff>
   <LÃ¥n80Nom>0.085</LÃ¥n80Nom>
   <LÃ¥n80Eff>0.0896</LÃ¥n80Eff>
   <Forbruk>0.156</Forbruk>
   <BrukslÃ¥n>0.156</BrukslÃ¥n>
   <BrukslÃ¥nProvisjon>0.0025</BrukslÃ¥nProvisjon>
   <InnskuddEndret>22.03.99</InnskuddEndret>
   <HÃ¸yrente25>0.04</HÃ¸yrente25>
   <HÃ¸yrente50>0.04</HÃ¸yrente50>
   <HÃ¸yrente100>0.045</HÃ¸yrente100>
   <HÃ¸yrente500>0.05</HÃ¸yrente500>
   <HÃ¸yrenteRest>0.065</HÃ¸yrenteRest>
   <BSU>0.0675</BSU>
   <HonnÃ¸r>0.025</HonnÃ¸r>
   <Bruks>0.02</Bruks>
   <LÃ¥nEtablering>950.0</LÃ¥nEtablering>
   <TerminKonto>20.0</TerminKonto>
   <TerminGiro>30.0</TerminGiro>
   <Sikkerhet>350.0</Sikkerhet>
   <Telegiro>2.0</Telegiro>
   <Brevgiro>4.0</Brevgiro>
   <Kassegiro>15.0</Kassegiro>
   <PCbank>0.0</PCbank>
   <Internett>2.0</Internett>
   <KontantKasse>0.0</KontantKasse>
   <EgenOverfÃ¸ring>0.0</EgenOverfÃ¸ring>
   <FastBank>10.0</FastBank>
   <ManuellBank>20.0</ManuellBank>
   <MiniÃ…r>200.0</MiniÃ…r>
   <MiniVarekjÃ¸p>2.0</MiniVarekjÃ¸p>
   <MiniKontantIE>0.0</MiniKontantIE>
   <MiniKontantIA>5.0</MiniKontantIA>
   <MiniKontantUE>0.0</MiniKontantUE>
   <MiniKontantUA>5.0</MiniKontantUA>
   <MiniKontantUU>0.0</MiniKontantUU>
   <VisaÃ…r>250.0</VisaÃ…r>
   <VisaVarekjÃ¸pI>2.0</VisaVarekjÃ¸pI>
   <VisaVarekjÃ¸pU>4.0</VisaVarekjÃ¸pU>
   <VisaKontantIE>0.0</VisaKontantIE>
   <VisaKontantIA>5.0</VisaKontantIA>
   <VisaKontantUE>0.0</VisaKontantUE>
   <VisaKontantUA>5.0</VisaKontantUA>
   <VisaKontantUU>20.0</VisaKontantUU>
  </informasjon>
  <informasjon>
   <BankID>2.0</BankID>
   <Bank>VÃ¥r Bank og Forsikring</Bank>
   <Web>http://www.var.no/</Web>
   <LÃ¥nEndret>08.03.99</LÃ¥nEndret>
   <LÃ¥n60Nom>0.0795</LÃ¥n60Nom>
   <LÃ¥n60Eff>0.084</LÃ¥n60Eff>
   <LÃ¥n80Nom>0.0845</LÃ¥n80Nom>
   <LÃ¥n80Eff>0.0893</LÃ¥n80Eff>
   <Forbruk>0.142</Forbruk>
   <BrukslÃ¥n>0.1525</BrukslÃ¥n>
   <BrukslÃ¥nProvisjon>0.0</BrukslÃ¥nProvisjon>
   <InnskuddEndret>28.02.99</InnskuddEndret>
   <HÃ¸yrente25>0.0475</HÃ¸yrente25>
   <HÃ¸yrente50>0.0475</HÃ¸yrente50>
   <HÃ¸yrente100>0.0549</HÃ¸yrente100>
```

```
<HÃ¸yrente500>0.0605</HÃ¸yrente500>
   <HÃ¸yrenteRest>0.0649</HÃ¸yrenteRest>
<BSU>0.07</BSU>
   <HonnÃ¸r>0.0</HonnÃ¸r>
   <Bruks>0.025</Bruks>
   <LÃ¥nEtablering>500.0</LÃ¥nEtablering>
   <TerminKonto>25.0</TerminKonto>
   <TerminGiro>50.0</TerminGiro>
   <Sikkerhet>1000.0</Sikkerhet>
   <Telegiro>2.0</Telegiro>
   <Brevgiro>5.0</Brevgiro>
   <Kassegiro>15.0</Kassegiro>
   <PCbank>0.0</PCbank>
   <Internett>2.0</Internett>
   <KontantKasse>0.0</KontantKasse>
   <EgenOverfÃ¸ring>0.0</EgenOverfÃ¸ring>
   <FastBank>15.0</FastBank>
   <ManuellBank>20.0</ManuellBank>
   <MiniÃ…r>100.0</MiniÃ…r>
   <MiniVarekjÃ¸p>1.5</MiniVarekjÃ¸p>
   <MiniKontantIE>0.0</MiniKontantIE>
   <MiniKontantIA>0.0</MiniKontantIA>
   <MiniKontantUE>4.0</MiniKontantUE>
   <MiniKontantUA>4.0</MiniKontantUA>
   <MiniKontantUU>0.0</MiniKontantUU>
   <VisaÃ…r>300.0</VisaÃ…r>
   <VisaVarekjÃ¸pI>1.5</VisaVarekjÃ¸pI>
   <VisaVarekjÃ¸pU>4.0</VisaVarekjÃ¸pU>
   <VisaKontantIE>0.0</VisaKontantIE>
   <VisaKontantIA>0.0</VisaKontantIA>
   <VisaKontantUE>4.0</VisaKontantUE>
   <VisaKontantUA>4.0</VisaKontantUA>
   <VisaKontantUU>35.0</VisaKontantUU>
  </informasjon>
```

Vedlegg 7: Eksempelet Komplett

XSL-fila "Laan60Nom"

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
    <HTML>
      <BODY>
        <STYLE>
          BODY {margin:0}
          .bgtitle {font:8pt Verdana; background-color:navy; color:white}
          .bgbody {font:8pt Verdana; background-color:white; color:navy}
          H1 {font:bold 14pt Verdana; width:100%; margin-top:1em}
         H2 {font:bold 12pt Verdana; width:100%; margin-top:1em; margin-bottom:0em;
color:teal;}
         H3 {font:bold 9pt Verdana; width:100%; margin-top:1em; margin-bottom:0em;
color:teal;}
          .row {font:8pt Verdana font-color:#000080; border-bottom:1px solid #CC88CC}
          .header {font:bold 9pt Verdana; cursor:hand; padding:2px; border:2px outset gray}
          .up {background-color:#DDFFDD;}
          .down {background-color:#FFDDDD;}
        </STYLE>

       <!-- Tabell for overskriften -->
        <TABLE width="100%" cellspacing="0">
          <TR><TD class="bgtitle"/><TD class="bgtitle"><H1>Bankenes
informasjonsside</H1></TD></TR>
       </TABLE>
        <TABLE width="100%" cellspacing="0">
          <TR><TD class="bgbody"/><TD class="bgbody"><H2>Laanetilbud: </H2></TD></TR>
       </TABLE>
        <TABLE BORDER="2" width="100%" cellspacing="0">
          <TR>
           <TD classe="bgbody"><H3>Bank: </H3></TD>
           <TD class="bgbody"><H3>Laan60Nom: </H3></TD>
           <TD class="bgbody"><H3>Laan60Eff: </H3></TD>
           <TD class="bgbody"><H3>Laan80Nom: </H3></TD>
           <TD class="bgbody"><H3>Laan60Nom: </H3></TD>
           <TD class="bgbody"><H3>Forbruk: </H3></TD>
           <TD class="bgbody"><H3>Brukslaan: </H3></TD>
          </TR>
         <xsl:for-each select="Banker/samling/informasjon" order-by="LÃ¥n60Nom">
          <TR STYLE="font-size:8pt; font-familiy:Verdana">
           <TD><xsl:value-of select="Bank"/></TD>
           <TD><xsl:value-of select="LÃ¥n60Nom"/></TD>
            <TD><xsl:value-of select="LÃ¥n60Eff"/></TD>
            <TD><xsl:value-of select="LÃ¥n80Nom"/></TD>
            <TD><xsl:value-of select="LÃ¥n80Eff"/></TD>
            <TD><xsl:value-of select="Forbruk"/></TD>
            <TD><xsl:value-of select="BrukslÃ¥n"/></TD>
          </TR>
         </xsl:for-each>
        </TABLE>
      </BODY>
    </HTML>
  </xsl:template>
</xsl:stylesheet>
```

Fila komplett.htm

```
<HTML>
  <HEAD>
    <TITLE>Bank Demo</TITLE>
  </HEAD>

  <SCRIPT FOR="window" EVENT="onload">
    kontroll.init();
  </SCRIPT>

  <FRAMESET COLS="180,*" FRAMEBORDER="0" BORDER="no" FRAMESPACING="0">
    <FRAME ID="kontroll" SRC="kontroll.htm" SCROLLING="no">
    <FRAME ID="visning" SRC="visning.htm">
  </FRAMESET>
</HTML>
```

Fila visning.htm

```
<HTML>
  <SCRIPT>
    function display(xslHTML)
    {
      document.all.item("xslresult").innerHTML = xslHTML;

      // Include the following line to view the HTML source as well
      // document.all.item("xsldebug").innerHTML = "<XMP>" + xslHTML + "</XMP>";
    }
  </SCRIPT>

  <BODY>
    <DIV id="xslresult">
      <!-- resulting HTML will be inserted here -->
    </DIV>
    <DIV id="xsldebug">
      <!-- resulting HTML source will be inserted here -->
    </DIV>
  </BODY>
</HTML>
```

Fila kontroll.htm

```
<HTML>
  <HEAD>
    <STYLE>
      BODY { font-family:Verdana; font-size:9pt; margin:0px; color:teal; }
      .instructions { font-style:italic; text-align:right; color:gray; margin-left:.5em; }
      .bgtitle {font:8pt Verdana; background-color:teal; color:white}
      H1 {font:bold 14pt Verdana; width:100%; margin-top:1em}
      .group { font-family:Arial Black; font-size:14pt; margin-top:.5em; margin-left:.5em;
               margin-bottom:.5em; }
      .group2 { font-family:Verdana; font-size:12pt; font-weight:bold; margin-top:.5em;
margin-left:.5em;
               margin-bottom:.5em; }
      .button { font-family:Verdana; font-size:10pt; font-weight:bold; text-align:right;
                color:gray; }
      .arrow { font-family:Webdings; }
    </STYLE>
  </HEAD>

  <SCRIPT>
    var source;
    var style;
    var sourceURL;
    var styleURL;
    var viewingSrc;

    // ----- Scripts to control XSL Processing ------
    function update()
    {
      if (style.documentElement && source.documentElement)
      {
        parent.visning.display(source.transformNode(style));
      }
    }

    function changeXML(xmldoc)
    {
      if (viewingSrc)
      {
        styleURL = sourceURL;
      }
      sourceURL = xmldoc;
      source.load(sourceURL);
      if (viewingSrc)
      {
        viewingSrc = false;
        style.load(styleURL);
      }
      update();
    }
    function changeXSL(xsldoc)
    {
      if (!viewingSrc)
      {
        styleURL = xsldoc;
        style.load(styleURL);
      }
      else
      {
        sourceURL = xsldoc;
        source.load(sourceURL);
      }

      update();
    }

  </SCRIPT>
```

```
<SCRIPT>
  // ----- Scripts to activate buttons ------
  var oldXMLitem;
  var oldXSLitem;

  function over(item)
  {
    item.style.color = "black";
  }

  function out(item) {
    item.style.color = "gray";
  }

  function select(group, item) {
      oldXSLitem.style.textDecoration = "";
      oldXSLitem = item;
      item.style.textDecoration = "underline";
  }

  // called by parent frame when the whole frameset is ready
  function init()
  {
    oldXSLitem = document.all.item("first-XSL-item");
    select("xsl", oldXSLitem);

    source = new ActiveXObject("Microsoft.XMLDOM");
    source.async = false;
    style = new ActiveXObject("Microsoft.XMLDOM");
    style.async = false;

    changeXML("db2xml.xml");
    changeXSL("Laan60Nom.xsl");
  }
</SCRIPT>

<BODY>
   <TABLE width="100%" cellspacing="0">
     <TR><TD class="bgtitle"/><TD class="bgtitle"><H1>Sortering</H1></TD></TR>
   </TABLE>

   <DIV CLASS="group">XSL Stylesheets
   <DIV CLASS="group2">Laanetilbud:
    <DIV CLASS="button" ID="first-XSL-item"
         onMouseOver="over(this)"
         onMouseOut="out(this)"
         onClick='changeXSL("Laan60Nom.xsl"); select("xsl",this)'>
      Laan60Nom.xsl<SPAN CLASS="arrow">4</SPAN>
    </DIV>
    <DIV CLASS="button"
         onMouseOver="over(this)"
         onMouseOut="out(this)"
         onClick='changeXSL("Laan60Eff.xsl"); select("xsl",this)'>
      Laan60Eff.xsl<SPAN CLASS="arrow">4</SPAN>
    </DIV>
    <DIV CLASS="button"
         onMouseOver="over(this)"
         onMouseOut="out(this)"
         onClick='changeXSL("Laan80Nom.xsl"); select("xsl",this)'>
      Laan80Nom.xsl<SPAN CLASS="arrow">4</SPAN>
    </DIV>
    <DIV CLASS="button"
         onMouseOver="over(this)"
         onMouseOut="out(this)"
         onClick='changeXSL("Laan80Eff.xsl"); select("xsl",this)'>
      Laan80Eff.xsl<SPAN CLASS="arrow">4</SPAN>
    </DIV>
```

```
       <DIV CLASS="button"
           onMouseOver="over(this)"
           onMouseOut="out(this)"
           onClick='changeXSL("Forbruk.xsl"); select("xsl",this)'>
         Forbruk.xsl<SPAN CLASS="arrow">4</SPAN>
       </DIV>
       <DIV CLASS="button"
            onMouseOver="over(this)"
            onMouseOut="out(this)"
            onClick='changeXSL("Brukslaan.xsl"); select("xsl",this)'>
         Brukslaan.xsl<SPAN CLASS="arrow">4</SPAN>
       </DIV>
      <DIV CLASS="group2">Gebyrer:
       <DIV CLASS="button"
            onMouseOver="over(this)"
            onMouseOut="out(this)"
            onClick='changeXSL("Laanetablering.xsl"); select("xsl",this)'>
         Laanetablering.xsl<SPAN CLASS="arrow">4</SPAN>
       </DIV>
       <DIV CLASS="button"
            onMouseOver="over(this)"
            onMouseOut="out(this)"
            onClick='changeXSL("Terminkonto.xsl"); select("xsl",this)'>
         Terminkonto.xsl<SPAN CLASS="arrow">4</SPAN>
       </DIV>
       <DIV CLASS="button"
            onMouseOver="over(this)"
            onMouseOut="out(this)"
            onClick='changeXSL("Termingiro.xsl"); select("xsl",this)'>
         Termingiro.xsl<SPAN CLASS="arrow">4</SPAN>
       </DIV>
       <DIV CLASS="button"
            onMouseOver="over(this)"
            onMouseOut="out(this)"
            onClick='changeXSL("Sikkerhet.xsl"); select("xsl",this)'>
         Sikkerhet.xsl<SPAN CLASS="arrow">4</SPAN>
       </DIV>
       <DIV CLASS="button"
            onMouseOver="over(this)"
            onMouseOut="out(this)"
            onClick='changeXSL("Telegiro.xsl"); select("xsl",this)'>
         Telegiro.xsl<SPAN CLASS="arrow">4</SPAN>
       </DIV>
       <DIV CLASS="button"
            onMouseOver="over(this)"
            onMouseOut="out(this)"
            onClick='changeXSL("Brevgiro.xsl"); select("xsl",this)'>
         Brevgiro.xsl<SPAN CLASS="arrow">4</SPAN>
       </DIV>
       <DIV CLASS="button"
            onMouseOver="over(this)"
            onMouseOut="out(this)"
            onClick='changeXSL("Kassegiro.xsl"); select("xsl",this)'>
         Kassegiro.xsl<SPAN CLASS="arrow">4</SPAN>
       </DIV>
       <DIV CLASS="button"
            onMouseOver="over(this)"
            onMouseOut="out(this)"
            onClick='changeXSL("PCBank.xsl"); select("xsl",this)'>
         PCBank.xsl<SPAN CLASS="arrow">4</SPAN>
       </DIV>
       <DIV CLASS="button"
            onMouseOver="over(this)"
            onMouseOut="out(this)"
            onClick='changeXSL("Internett.xsl"); select("xsl",this)'>
         Internett.xsl<SPAN CLASS="arrow">4</SPAN>
       </DIV>
     </DIV>
   </BODY>
</HTML>
```

Vedlegg 8: Eksempelet Altiett

XSL-fila "Laan60Nom"

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
    <HTML>
      <HEAD>
        <STYLE>
          BODY {margin:0}
          .bgtitle {font:8pt Verdana; background-color:teal; color:white}
          H1 {font:bold 14pt Verdana; width:100%; margin-top:1em}
         H2 {font:bold 12pt Verdana; width:100%; margin-top:1em; margin-bottom:0em;
color:white;}
          H3 {font:bold 9pt Verdana; width:100%; margin-top:1em; margin-bottom:0em;
color:navy}
        </STYLE>
      </HEAD>

      <SCRIPT>
        function sort(field)
        {
          sortField.value = field;
          listing.innerHTML = source.documentElement.transformNode(stylesheet);
        }
       function hand(item)
       {
         item.style.cursor = "hand";
       }
      </SCRIPT>

      <SCRIPT for="window" event="onload"><xsl:comment><![CDATA[
        stylesheet = document.XSLDocument;
        source = document.XMLDocument;
        sortField = document.XSLDocument.selectSingleNode("//@order-by");
      ]]></xsl:comment></SCRIPT>

      <BODY STYLE="background-color:teal">
        <TABLE width="100%" cellspacing="0">
          <TR>
            <TD class="bgtitle"/>
            <TD class="bgtitle">
              <H1>Bankenes informasjonsside</H1>
            </TD>
          </TR>
          <TR>
            <TD class="bgtitle"/>
           <TD class="bgtitle" width="100%" valign="top">
              <H2>Renter paa Innskudd</H2>
            </TD>
         </TR>
         <TR>
            <TD class="bgtitle" width="170" valign="top">
              <P>Klikk paa en kolonne for aa sortere etter denne.</P>
            </TD>
            <TD class="bgtitle" valign="top">
              <DIV id="listing"><xsl:apply-templates match="Banker"/></DIV>
            </TD>
          </TR>
        </TABLE>
      </BODY>
    </HTML>
  </xsl:template>
  <xsl:template match="Banker">
```

```
<TABLE BORDER="2" width="100%" cellspacing="0"  STYLE="font-size:8pt; background-
color:white; font-familiy:Verdana">
      <THEAD>
        <TD width="200"><DIV onClick="sort('Bank')"><H3
onMouseOver="hand(this)">Bank</H3></DIV></TD>
        <TD STYLE="text-align=center" width="80"><DIV onClick="sort('HÃ¸yrente25')"><H3
onMouseOver="hand(this)">HÃ¸yrente 25.000</H3></DIV></TD>
        <TD STYLE="text-align=center" width="80"><DIV onClick="sort('HÃ¸yrente50')"><H3
onMouseOver="hand(this)">HÃ¸yrente 50.000</H3></DIV></TD>
        <TD STYLE="text-align=center" width="80"><DIV onClick="sort('HÃ¸yrente100')"><H3
onMouseOver="hand(this)">HÃ¸yrente 100.000</H3></DIV></TD>
        <TD STYLE="text-align=center" width="80"><DIV onClick="sort('HÃ¸yrente500')"><H3
onMouseOver="hand(this)">HÃ¸yrente 500.000</H3></DIV></TD>
        <TD STYLE="text-align=center" width="80"><DIV onClick="sort('HÃ¸yrenteRest')"><H3
onMouseOver="hand(this)">HÃ¸yrente Rest</H3></DIV></TD>
        <TD STYLE="text-align=center" width="80"><DIV onClick="sort('BSU')"><H3
onMouseOver="hand(this)">BSU</H3></DIV></TD>
        <TD STYLE="text-align=center" width="80"><DIV onClick="sort('HonnÃ¸r')"><H3
onMouseOver="hand(this)">HonnÃ¸r konto</H3></DIV></TD>
        <TD STYLE="text-align=center" width="80"><DIV onClick="sort('Bruks')"><H3
onMouseOver="hand(this)">Bruks konto</H3></DIV></TD>
      </THEAD>
      <xsl:for-each select="samling/informasjon" order-by="Bank">
        <TR>
          <TD><xsl:value-of select="Bank"/></TD>
          <TD STYLE="text-align=right"><xsl:value-of select="HÃ¸yrente25"/></TD>
          <TD STYLE="text-align=right"><xsl:value-of select="HÃ¸yrente50"/></TD>
          <TD STYLE="text-align=right"><xsl:value-of select="HÃ¸yrente100"/></TD>
          <TD STYLE="text-align=right"><xsl:value-of select="HÃ¸yrente500"/></TD>
          <TD STYLE="text-align=right"><xsl:value-of select="HÃ¸yrenteRest"/></TD>
          <TD STYLE="text-align=right"><xsl:value-of select="BSU"/></TD>
          <TD STYLE="text-align=right"><xsl:value-of select="HonnÃ¸r"/></TD>
          <TD STYLE="text-align=right"><xsl:value-of select="Bruks"/></TD>
        </TR>
      </xsl:for-each>
    </TABLE>
  </xsl:template>
</xsl:stylesheet>
```

Vedlegg 9: Eksempelet FondFinner

Fila Fondfinner.htm

```
<HTML>
<HEAD>
  <STYLE>
    BODY {margin:0}
    .bgtitle {font:8pt Verdana; background-color:teal; color:white; width:100%}
    H1 {font:bold 14pt Verdana; width:100%; margin-top:1em}
    H2 {font:bold 12pt Verdana; width:100%; margin-top:1em; margin-bottom:1em; color:teal;}
    H3 {font:bold 9pt Verdana; width:100%; border-color:white;  margin-top:1em; margin-
bottom:0em; color:navy}
   .row {font:bold 8pt Verdana; border-size:0; border-color:white; text-align:center;}
   .tomrow {font-size:1pt; border-size:0; border-color:white; text-align:center; background-
color:teal; font-color:teal}
  </STYLE>
 <TABLE style="width=100%">
    <TR><TD class="bgtitle"/></TR>
    <TR><TD class="bgtitle"><H1>Bankenes informasjonsside</H1></TD></TR>
    <TR><TD class="bgtitle"/></TR>
  </TABLE>
</HEAD>
<BODY onload="WaitToFillBoxes()">
<TABLE>
  <TD STYLE="width:30"></TD>
  <TD><H2>Her kan du finne informasjon om fond</H2>

  <!-- XML Data Island for the selected banks -->
  <XML id=selBanks></XML>

  <!-- XML Data Island for the XSL stylesheet -->
  <XML id=bankSS SRC="fond.xsl"></XML>

  <DIV ID=selectDIVs>
  <SPAN style="FONT-WEIGHT: bold; FONT-SIZE: 12; FONT-FAMILY: Verdana">Velg en bank:
</SPAN><SPAN ID=choices2></SPAN>
  <BR>
  <SPAN style="FONT-WEIGHT: bold; FONT-SIZE: 12; FONT-FAMILY: Verdana">Velg et fond :
</SPAN><SPAN ID=choices3></SPAN>
  </DIV>
  </TD>
  <TD STYLE="width:20"></TD>
  <TD><BR><BR><BR><BR><INPUT TYPE=button NAME="Vis Informasjon" VALUE="Vis Fond"
onclick="getSamples()"><BR></TD>
  <TD STYLE="width:80"></TD>
  <TD>
    <img src="spgris.JPG" width="120" height="130">
  </TD>
</TABLE>

<SCRIPT>
var retNodes;
var patternString;

function fillBoxes(){
  makeBox("Navn","BankNavn",1);
  makeBox("navn","fondnavn",4);
  }
```

```
function makeBox(elementName,boxName,divIndex){
  var pattern = "//" + elementName;
  var selNodes = bankliste.documentElement.selectNodes(pattern);
  var holderNode = bankliste.createNode(1,"holder","");
  holderNode.insertBefore(selNodes.item(0).cloneNode(true), null);
  for (var j=1;j< selNodes.length;j++){
    counter = 0;
    for (var k=0; k< holderNode.childNodes.length;k++){
      if (selNodes.item(j).text == holderNode.childNodes.item(k).text ||
selNodes.item(j).text == ""){
        counter=1;
        break;
        }
      else counter = 0;
      }
    if (counter == 0){
      holderNode.insertBefore(selNodes.item(j).cloneNode(true), null);
      }
    }
  buildSelect(boxName,holderNode.childNodes,divIndex);
  }

function buildSelect(boxName, selNodes, divIndex){
  var str = "<SELECT NAME='" + boxName + "' SIZE=1><OPTION VALUE='' SELECTED>Alle";
       for (var i=0; i < selNodes.length; i++){
        str += "<OPTION VALUE='" + selNodes.item(i).text + "'>" + selNodes.item(i).text;
        }
      str += "</SELECT>";
      selectDIVs.children.item(divIndex).innerHTML = str;
  }

function getSamples(){
  var patternString = "//Bank[";
  if (BankNavn.value != "")
    patternString += "Navn='" + BankNavn.value + "'";
  if (fondnavn.value != ""){
    if (BankNavn.value != "")
      patternString += " $and$ fond/navn='" + fondnavn.value + "'";
    else patternString += "fond/navn='" + fondnavn.value + "'";
    }
  patternString += "]"
  if (patternString == "//Bank[]")
    patternString = "//Bank";
  var selectedNodes = bankliste.selectNodes(patternString);
  var bankHolder = bankliste.createNode(1,"Informasjon","");
  for (var n=0; n< selectedNodes.length; n++)
    bankHolder.insertBefore(selectedNodes.item(n).cloneNode(true),null);
  insertTable.innerHTML = bankHolder.transformNode(bankSS.documentElement);
  }

function getState(){
  if (bankliste.readyState == "complete")
    fillBoxes();
  }

function WaitToFillBoxes(){
  window.status = "waiting...";
  if (bankliste.readyState != "complete")
    window.setTimeout("WaitToFillBoxes()",100);
  if (bankSS.readyState != "complete")
    window.setTimeout("WaitToFillBoxes()",100);

  window.status = "Fond";
  fillBoxes();
}
```

```
</SCRIPT>

<!-- XML Data Island for the banklist data -->
<XML ID="bankliste" SRC = "fond.xml" ></XML>


<HR>
<!-- DIV that will hold the HTML string returned by the transformNode -->
<DIV style="background-color:white"/>
<TABLE>
  <TR><TD></TD><TD id="insertTable"></TD></TR>
</TABLE>


</BODY>
</HTML>
```

Fila fond.xsl

```
<DIV xmlns:xsl="http://www.w3.org/TR/WD-xsl">

  <TABLE BORDER="2" width="100%" cellspacing="0"  STYLE="font-size:8pt; background-
color:white; font-familiy:Verdana">
  <col width="190"/>
  <col width="170"/>
  <col width="60"/>
  <col width="60"/>
  <col width="60"/>
  <col width="60"/>
  <col width="60"/>
  <col width="60"/>
  <THEAD>
    <TD><H3>Bank</H3></TD>
    <TD><H3>Fond</H3></TD>
    <TD><H3>Avk. siste 12mnd</H3></TD>
    <TD><H3>Avk. siste 24mnd</H3></TD>
    <TD><H3>Avk. siste 48mnd</H3></TD>
    <TD><H3>10000 etter 1 aar</H3></TD>
    <TD><H3>10000 etter 2 aar</H3></TD>
    <TD><H3>10000 etter 4 aar</H3></TD></THEAD>
      <TR class="tomrow">
        <TD>---</TD><TD>---</TD><TD>---</TD>
        <TD>---</TD><TD>---</TD><TD>---</TD>
        <TD>---</TD><TD>---</TD>
      </TR>
    <TR>
    <xsl:for-each select="Bank">
      <TD class="row"><xsl:value-of select="Navn"/></TD>
      <xsl:for-each select="fond">
      <TR>
        <TD></TD><TD class="row"><xsl:value-of select="navn"/></TD>
        <xsl:for-each select="avkastning">
          <TD class="row"><xsl:value-of select="siste_12_mnd"/></TD>
          <TD class="row"><xsl:value-of select="siste_24_mnd"/></TD>
          <TD class="row"><xsl:value-of select="siste_48_mnd"/></TD>
          <TD class="row"><xsl:value-of select="etter_1_aar_har_10000_vokst_til"/></TD>
          <TD class="row"><xsl:value-of select="etter_2_aar_har_10000_vokst_til"/></TD>
          <TD class="row"><xsl:value-of select="etter_4_aar_har_10000_vokst_til"/></TD>
        </xsl:for-each>
      </TR>
      </xsl:for-each>
      <TR STYLE="font-size:0.5; font-color:teal; background-color:teal; text-align=center">
        <TD>-</TD><TD>-</TD><TD>-</TD>
        <TD>-</TD><TD>-</TD><TD>-</TD>
        <TD>-</TD><TD>-</TD>
      </TR>
    </xsl:for-each>
  </TR>
</TABLE>
</DIV>
```