NORGES TEKNISK-NATURVITENSKAPELIGE UNIVERSITET

FAKULTET FOR INFORMASJONSTEKNOLOGI, MATEMATIKK OG ELEKTROTEKNIKK

MASTEROPPGAVE

Kandidatenes navn: Erlend Agøy Engum og Rolf Axel Wathne

Fag: TDT 4900 Masteroppgave, Sivilingeniørstudiet i Datateknikk

Oppgavens tittel (norsk): Databaser for dynamiske arbeidsprosesser

Oppgavens tittel (engelsk): Databases for dynamic task procedures

Oppgavens tekst:

I oppgaven skal man studere beskrivelser av arbeidsprosesser og vurdere hvilke krav lagring av slike setter til databaseløsningen. Med utgangspunkt i dette skal man utarbeide et forslag til datamodell for lagring av arbeidsprosesser og man skal utvikle en prototyp som demonstrerer hvordan en del utvalgte viktige problemer kan løses i praksis.

Oppgaven gitt:	20. januar 2004
Besvarelsen leveres innen:	22. juni 2004
Besvarelsen levert:	18. juni 2004
Utført ved:	Institutt for datateknikk og informasjonsvitenskap
Veiledere:	Folke B. M. B. Sørli og Jostein Sveen

Trondheim, 18. juni 2004

Roger Midtstraum
Faglærer

Databaser for dynamiske arbeidsprosesser

Axel Wathne, IDI - NTNU
Erlend Agøy Engum, IDI - NTNU

17. juni 2004

Forord

Dette dokumentet er en masteroppgave skrevet våren 2004. Den er avslutningen av sivilingeniørstudiet i datateknikk ved Institutt for Datateknikk og Informasjonsvitenskap - IDI, ved Norges Teknisk-Naturvitenskapelige Universitet - NTNU. Oppgaven er skrevet for flere interessenter og vi vil gjerne kartlegge for leseren de ulike synspunktene vi har tatt hensyn til i løpet av skriveprosessen. Følgende er vår forståelse av oppgavens utvikling.

Vår oppdragsgiver og hovedsamarbeidspartner har vært Sintef Teknologiledelse. De samarbeider med Norsk Hydro i forbindelse med prosjekter som omhandler *arbeidsprosessorientert styring*. Hydro ønsker å beskrive arbeidsprosesser ved hjelp av elektroniske hjelpemidler, i motsetning til dagens papirbaserte system. Til denne oppgaven har Hydro kjøpt et system fra SoluDyne AS. Sintef ble engasjert for å se på brukergrensesnitt og generelt kvalitetssikre prosessen. Det er her de har ønsket vår uavhengige, faglige vurdering av datatekniske utfordringer rundt lagring og behandling av arbeidsprosesser.

Da vi begynte arbeidet var det meningen at vi skulle jobbe nært opp mot SoluDyne og/eller Norsk Hydro. Dessverre har omstendighetene rundt dette samarbeidet vært noe vanskelige. Det viste seg at Stavangerbaserte SoluDyne ikke hadde kapasitet til å følge opp vårt arbeid på en effektiv måte. De ville heller ikke gi tilgang til dokumentasjon på systemet deres og vi bestemte oss for å fokusere på andre problemstillinger. Ved siden av dette ble Hydros prosjektleder dessverre sykemeldt i over to måneder og deres prosjekt ble midlertidig stanset. Vi fokuserte derfor på Sintefs daværende problemforståelse. Denne ble vi oppfordret til å jobbe videre med.

I løpet av arbeidet har vi lært mye nytt. For det første har vi hatt muligheten til å sette oss inn i arbeidsprosesser, noe vi er overbevist om å treffe på også i våre profesjonelle karrierer. For det andre kan vi nå mye mer om strukturering og prosessering av data basert på xml. Vi har også lært noe om er arbeid under skiftende forutsetninger med mange interessenter og ulike interesser.

Vi har valgt å skrive en kompakt rapport som er ment å belyse både teoretisk bakgrunn, arkitektur og spesielle aspekter ved elektronisk lagring av arbeidsprosesser. I valg av teori og bakgrunnsstoff har vi til dette fått uvurderlig hjelp av vår faglærer førsteamanuensis Roger Midtstraum og førsteamanuensis Trond Aalberg, begge ved IDI. I tillegg vil vi rette en stor takk til vår alltid positive og entusiastiske veileder ved Sintef, Folke Sørli og koordinator Jostein Sveen.

Trondheim, Juni 2004

Erlend Agøy Engum Rolf Axel Wathne

Ressurspersoner

Hovedfaglærer Roger Midtstraum, IDI - NTNU
`Roger.Midtstraum@idi.ntnu.no`

Koordinator Jostein Sveen, SINTEF Teknologiledelse
`Jostein.Sveen@sintef.no`, Tlf 73596066

Veileder Folke B. M. B. Sørli, SINTEF Teknologiledelse
`Folke.Sorli@sintef.no`, Tlf 73592457 / 95853244

Sammendrag

Bedrifter med sikkerhetskritiske arbeidsoppgaver kan oppnå sikrere og mer effektiv drift gjennom standardisering og formalisering av disse. Slike arbeidsbeskrivelser, eller arbeidsprosessdokumenter, må være tilgjengelige til enhver tid. Presentasjonen av dem bør være tilpasset konteksten de aksesseres i, og må kunne støttes med multimediadata. Ettersom arbeidsprosessene stadig forandres og økes i omfang er det viktig å sikre konsistensen internt og dem imellom.

Vi har sett på rammeverk og lagringsmetoder i et system som skal lagre og tilgjengeliggjøre arbeidsprosessdokumentene. Vi har definert et lagringsformat ved hjelp av et w3c xml-schema, som lar oss spesifisere regler og beskrankninger for dokumentene som vanlige behandlingsverktøy for xml kan verifisere dem opp mot. Et strengt spesifisert format sikrer interoperabilitet mellom de forskjellige systemkomponentene som skal behandle informasjonen.

Vårt dokumentformat beskriver arbeidsprosessene som flytskjemaer med tilhørende informasjon. For at store og komplekse arbeidsprosesser skal bli oversiktlige legger vi opp til at logiske grupperinger av aktiviteter innenfor en arbeidsprosess skilles ut som egne arbeidsprosesser, spesielt dersom en tilsvarende gruppering er å finne som en del av flere arbeidsprosesser. Disse lavere-nivå-arbeidsprosessene inkluderes så i denne og andre arbeidsprosesser som subprosesser. Dersom man gjør forbedringer i en subprosess som benyttes i mange sammenhenger sikrer dette at den nye informasjonen er tilgjengelig for alle.

Vi har implementert et system som demonstrerer konseptet vårt, basert på en sql-database og en applikasjonsserver med webgrensesnitt. Systemet lar brukeren legge arbeidsprosesser inn i databasen, og kan vise disse som websider.

Innhold

Tabeller

Figurer

Kapittel 1

Innledning

”Vår viktigste ressurs er våre ansattes kunnskap, erfaring og kontaktnett” er en påstand vi ofte hører. Ansattes kunnskap og erfaring varierer fra person til person og påvirker måten de utfører sine oppgaver på. Formalisering av denne kunnskapen gjør at den kan spares og overføres til andre.

En stor bedrift har mye å tjene på standardiserte arbeidsmetoder. Arbeidere bør gjøre samme operasjon på samme sikre og effektive måte uansett hvor de jobber, derfor standardiseres metodene i arbeidsprosesser som binder arbeiderens erfaring til bedriftens organisasjonskunnskap. Standardiseringen kombinert med gode muligheter for tilbakemelding og revisjon av prosessene bidrar til at den enkeltes erfaring har en kanal tilbake til organisasjonskunnskapen. Optimal dokumentering av arbeidsmetodene er krevende. Det er nødvendig med en beskrivelse som blant annet dekker motivasjon, ansvarsforhold, roller og utførelse.

Kunnskap på organisasjonsnivå er viktig for mennesker i hele bedriftsstrukturen. Arbeideren bruker prosessbeskrivelsen til å utføre sine daglige gjøremål. Mellomlederne bruker informasjonen til å fordele arbeidsoppgaver. Ledelsen skaffer seg grunnleggende kunnskap om egen bedrift. I tillegg kan opplæring og videreutdanning basere seg på nedskrevne arbeidsprosesser. Samlet kalles konseptet *arbeidsprosessorientert styring.*

Et produkt som har en så stor innvirkning på en bedrifts arbeidsmetoder må være av god kvalitet. Derfor er det viktig å ta i bruk den beste tilgjengelige teknologi. Informasjon som leveres må til enhver tid være oppdatert og korrekt. Systemer må derfor tåle fremtidige endringer og utvidelser. Moderne systemer forventes også å støtte prosessbeskrivelsen ved hjelp av diagrammer, bilder, tekst, lyd, video, animasjon og interaksjon. Dette setter store krav til underliggende struktur og arkitektur.

Vår problemstilling handler om formalisering og lagring av arbeidsprosesser: Hvordan skal disse beskrives, lagres og aksesseres, og hvilke krav stilles til deres funksjonalitet? Hvordan påvirker informasjonens stadige forandring måten disse må beskrives og behandles på? Vi skal besvare disse spørsmålene, utvikle et lagringsformat og implementere et rammeverk som forvalter dokumenter basert på formatet.

Vi begynner med å analysere typen informasjon som skal lagres. Dette vil være strukturert, tekstlig informasjon i kombinasjon med multimediadata. Så kartlegger vi bruken av slik informasjon i et informasjonssystem.

På bakgrunn av dette identifiserer vi i kapittel 3 kravene som stilles til systemet og muligheter som bør tas hensyn til, og spesifiserer en modell for strukturering av informasjonen og formatet som implementerer denne. Vi skisserer så en systemarkitektur før vi implementerer et system som lar oss demonstrere bruken av dokumentformatet. En beskrivelse av implementasjonen finnes i kapittel 4. Kapittel 5 inneholder konklusjon og forslag til videre arbeid.

Kapittel 2

Analyse av arbeidsprosessystemer

Begrepet arbeidsprosess står sentralt i hele denne rapporten. En arbeidsprosess skal beskrive en arbeidsoppgave på en slik måte at den kan utføres effektivt og sikkert. Arbeidsprosessen er en kilde til informasjon som under de riktige omstendighetene kan gjøres til kunnskap og visdom. Kapittelet åpner med en innføring til kunnskap og læring. Dette er basiskunnskap som trengs om en skal forstå systemer for kunnskapsoverføring. Arbeidsprosessen er en organiseringsform for slike systemer og generell informasjon om denne organiseringen kommer i delkapittelet ”arbeidsprosesser”.

Videre følger en innføring i multimediasystemer. Moderne arbeidsprosessystemer vil benytte rike beskrivelsesformer og det er derfor viktig å ha en god forståelse for hva dette innebærer før en analyserer og finner systemkomponenter. Dette skjer i den siste delen av kapittelet, kalt ”bruksanalyse av kodifiserte systemer". Kodifiserte systemer inneholder prosedyrekunnskap og det er slik informasjon vi ønsker å lage et system for. Bruksanalysen identifiserer ulike aktører og funksjonskrav gjennom en usecase-analyse.

2.1 Begreper

Denne første delen av rapporten tar for seg informasjon, kunnskap og læring. Her blir et solid begrepssett rundt kunnskap og informasjon dannet. Dette blir fundament for oppbygging og design senere i rapporten. Vi skal vise at forskjellige bedrifter har forskjellige mål med sin kunnskap, og derfor velger ulike strategier for sin håndtering av informasjon. De ulike strategiene blir skissert før det gjøres en større fordypning i det domenet som er mest relevant for oppgaven, arbeidsprosesser og organisering av disse.

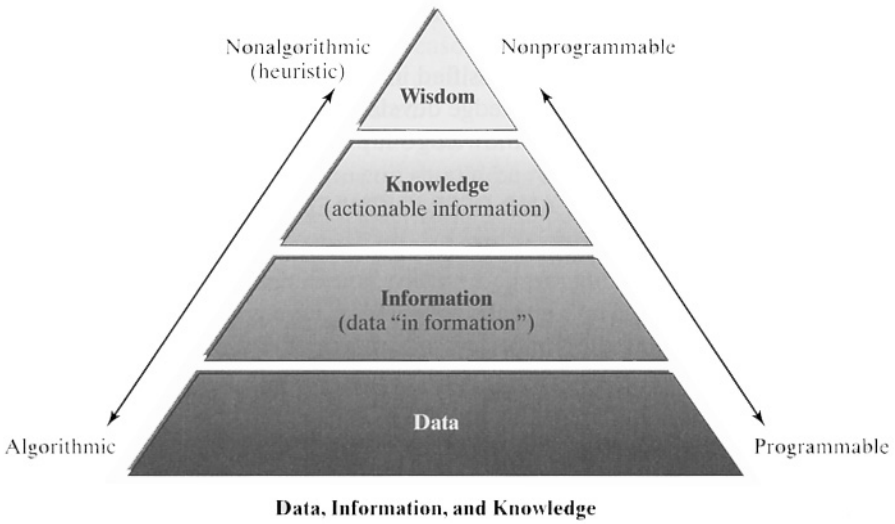
2.1.1 Kunnskap og læring

Når en leser om kunnskap generelt og programvare for kunnskapsdeling spesielt slår det en hvor mange måter man kan dele inn og definere kunnskap på. Dette avsnittet vil ta for seg definisjoner av kunnskap og kunnskapsdeling, og hvorfor disse definisjonene er valgt. Det vil også bli gitt definisjoner av prosessrelaterte begreper som blir sentrale videre i oppgaven.

Kunnskap

I begrepet kunnskap kan en legge mange forskjellige meninger. Her skal det vises noen måter å se på kunnskap på, disse vil danne grunnlag for behandlingen av kunnskapssystemer videre utover i rapporten.

Som en første inndeling av kunnskap tas det utgangspunkt i en kunnskapspyramide vist i figur 2.1 på neste side. Det laveste nivået i pyramiden er data. Data er råmaterialet. All kunnskap baserer seg på data, men data i seg selv har ikke struktur og derfor ikke mening. Over data ligger informasjon. Informasjon kan sies å være data ”i formasjon”. I en database er data lagret på en strukturert måte; en lager tabeller med metadata og de aktuelle dataene. Dette muliggjør tolkningen av dataen som informasjon. Informasjon har en også i grafiske diagrammer og i en tekst. Kunnskap er neste steg i pyramiden. Kunnskap har en når en på grunnlag av informasjon og erfaring med denne kan handle på riktig måte i en situasjon. Visdom danner toppen av kunnskapspyramiden. Visdom er kunnskap på et høyere nivå. Når en greier å løsrive sin kunnskap fra en situasjon og kan applikere den i andre situasjoner, forutse hva som vil skje i framtiden og finne kreative løsninger på tradisjonelle problemer har en oppnått en form for visdom.



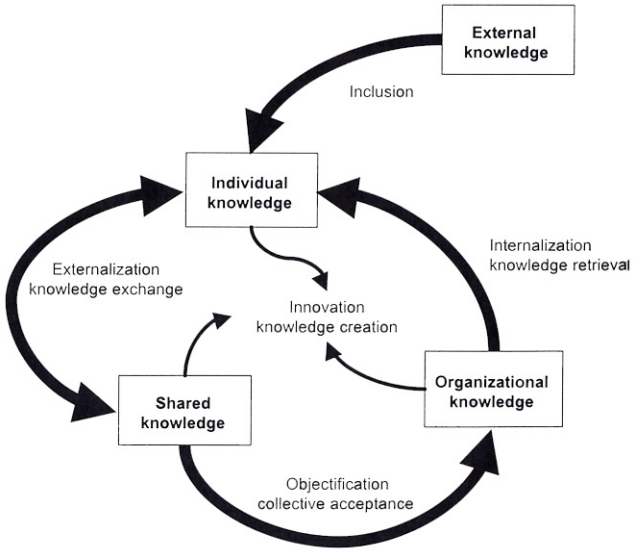
Figur 2.1: Kunnskapspyramiden, figur 2.2 s 41 i [AG04]

Som en andre form for inndeling av kunnskap kan en skille mellom *eksplisitt* og *stilltiende* kunnskap [AG04, HdW02]. Den stilltiende kunnskapen er noe alle har: kunnskap ervervet gjennom erfaring og jobbing over tid. Dette er kunnskapen som gjør oss i stand til å utføre de oppgavene vi er satt til å gjøre. Denne formen for kunnskap overføres ofte best i sosiale rom der andre kan lære gjennom undervisning og eksempler. Den stilltiende kunnskapen brukes også som grunnlag for å danne eksplisitt kunnskap. For at kunnskap skal regnes som eksplisitt må den være dokumentert. Det er mye enklere å distribuere den eksplisitte kunnskapen over distanser i tid og rom. Dessverre er det forskjell på å "vite at" og å "vite hvordan". En må derfor ha tilgang både til eksplisitt og stilltiende kunnskap samt danne egen erfaring før en selv kan si at en vet noe.

Den tredje inndelingen av kunnskap som vises her setter kunnskap inn i et organisasjonsperspektiv. Det kan en se i figur 2.2 på neste side hvor sammenhengene mellom individets, organisasjonens, og den delte kunnskapen vises. Her fokuseres det på de forskjellige kunnskapsprosessene som foregår innen en organisasjon. Disse blir her kort forklart:

Kunnskapsinnhenting er en prosess som individet utfører. En person kan hente informasjon fra organisasjonenes kunnskapsbase eller utenfra for å utvikle seg selv og lære noe nytt som kommer til nytte i organisasjonen.

Kunnskapsutveksling mellom individer har den intensjonen at en skal lære av hverandre for å utvikle en felles større forståelse.



Figur 2.2: Kunnskapssirkel, fra artikkelen [HdW03], figur 2.1 s 30 i [APW03]

Kunnskapsutvikling skjer mellom individene. Når den delte kunnskapen blir inkludert i organisasjonens kunnskap er det gjort innovasjon og organisasjonen har utviklet seg.

Læring

Et viktig mål for kunnskapsbærende systemer er at brukerne skal ta til seg kunnskapen slik at de gjør jobben sin så riktig, trygt og fort som mulig. Dette gjøres gjennom å klatre oppover i kunnskapspyramiden. Læring skal hjelpe til å gjøre informasjon til kunnskap og kunnskap til visdom. Som vist over finnes kunnskap både individuelt og organisasjonelt nivå. Begge disse er avhengige av at noen deler kunnskap.

Mange prinsipper er utviklet for elæring, her presenteres tre prinsipper som en kan ta utgangspunkt i [PC03]:

1. Ansatte lærer best i læringsgrupper. Derfor bør det alltid finnes tilgjengelig personell som kan svare på spørsmål.

2. Læring er mest effektiv når den fokuserer på reelle problemer. Det medfører at systemet på en eller annen måte må vise simulering av den faktiske arbeidssituasjoen som vil oppstå

3. Effektiv kunnskapsledelse innebærer å legge igjen synlige spor fra læreprosessen. Dette betyr at det må finnes en eller annen funksjon som fanger opp det som skjer underveis.

Det er ikke alltid mulig å danne læregrupper og få hjelp av annet personell. Da er det praktisk å ha et nettbasert alternativ som gjennom bruk av tekst, bilder, film, animasjoner, diskusjonsfora og lignende kan prøve å fylle denne funksjonen. Ved bruk av ulike virkemidler kan en simulere faktiske situasjoner uten å faktisk være der. Etter at situasjonen er gjennomgått vil det være mulig å skrive kommentarer og besvare spørsmål slik at læreprosessen blir fulgt opp.

Å dele kunnskap er en meget kompleks prosess. Det er skrevet mange bøker om dette, både innen pedagogikk og kunnskapsledelse. Det går langt utenfor denne oppgavens grenser å gi en full innføring i disse temaene, isteden fokuseres det på noen av de problemene en støter på under utvikling av kunnskapsbehandlingssystemer, og hvilke avveininger som gjøres i forhold til disse.

2.1.2 Systemer for kunnskapsoverføring

Systemer for kunnskapsoverføring har sitt arbeidsområde i skjæringen mellom informasjon og kunnskap, se figur 2.1 på side 5. Innen datateknikken finnes en gren som heter kunnskapssystemer. Disse har som mål å kunne ta selvstendige avgjørelser på grunnlag av et informasjonssett og på den måten ta over menneskers oppgaver. Likevel er det slik at kunnskap er noe som forbindes med mennesker. Informasjon er noe både mennesker og maskiner kan inneha. Noen typer informasjon lagres bedre hos mennesker, det er den som er kunnskapsnær og abstrakt. Datanær informasjon lagres sikrere i et datasystem enn hos mennesker fordi mennesker som oftest ikke har fotografisk minne, mens en datamaskin lagrer eksakt det den får inn og kan fremkalle det på akkurat samme måten senere. Dette får konsekvenser for hvordan systemdesignet blir seende ut.

Det finnes i hovedsak to strategier for slike systemer, *kodifiserte* og *personaliserte* [HNT99]. De neste avsnittene vil forklare trekk ved disse typene kunnskapssystemer. For denne oppgaven er den kodifiserte delen mest interessant, det vil derfor legges større vekt på denne.

Personaliserte systemer

Personaliserte systemer forkuserer på den kunnskapsnære informasjonen. De legger til rette for nærmere, bedre og mer effektivt samarbeid innad i kunnskaps-

bedriften. Stilltiende kunnskap er vanskeligere å overføre enn eksplisitt. Stilltiende kunnskap er også dypere kunnskap enn den skrevne, eksplisitt uttrykte kunnskapen. Det er derfor ofte nødvendig med førstehånds utveksling av erfaring og kunnskap for at resultatet skal bli bra. Mange bedrifter som jobber med utvikling og dermed er avhengige av hele tiden å finne nye, kreative løsninger velger gjerne et personalisert system. Denne løsningsformen søker å kople individer og grupper tilsvarende pilen ”knowledge exchange” i figur 2.2 på side 6.

Det er implementert mange kunnskapssystemer med den personaliserte innfallsvinkelen. Dermed finnes også mange ulike tekniske løsninger. Både søkbarhet, mengde og type informasjon kan variere fra system til system. De har likevel en del felles faktorer:

Profiler er oftest den sentrale bærebjelken i et personalisert system. En profil inneholder informasjon som forteller om hvem en person er. Profilene brukes ofte aktivt til å finne ut av hvem som er hvem i en større organisasjon. Typisk informasjon som lagres i en profil er navn, kontaktinformasjon, bilder, arbeidshistorie og kunnskapsfelt.

Optimal utnyttelse av personell er det egentlige målet for systemet. Muligheten til å bli ”kjent” med medarbeidere før en møter dem og muligheten til å benytte seg av andres ekspertise er kjernefunksjonalitet. Disse egenskapene gjør systemet attraktivt å bruke og kan øke bedriftens produktivitet.

Oppdatering er en av de store utfordringene. De fleste kunnskapsbedrifter er dynamiske og ressurspersoner flytter rundt i eller forlater bedriften med jevne mellomrom. En kritisk tilstand oppstår om ikke brukerne har full tillit til at systemets kunnskap er identisk med de faktiske realiteter.

Tilgjengelighet av personell listet i systemet er viktig. Når en tar kontakt med en medarbeider gjennom systemet må en kunne forvente å få et svar. Det er også en stor fordel om systemet sier ifra dersom en person er utilgjengelig på grunn av praktiske årsaker som reise og arbeidspress eller organisasjonelle årsaker som lederansvar eller endret kompetansefokus.

For mer informasjon og bakgrunn om personaliserte systemer anbefales artiklene [HNT99, Ehr03, DR03], del II i [AG04] og boka [HdW02].

Kodifiserte systemer

Et kodifisert kunnskapssystem er et system som organiserer og presenterer eksplisitt kunnskap. Det er vanlig at bedrifter med faste arbeidsrutiner og en

prosedyrebasert hverdag velger et kodifisert system når de ønsker å etablere en kunnskapsbase. Et dataverktøy for dynamiske arbeidsprosesser kan kalles et kodifisert system. Vår oppgave heter ”databaser for dynamiske arbeidsprosesser”. For å kunne lage en database for dette verktøyet må en forstå hva et kodifisert system er og hva dette innebærer. Denne seksjonen tar utgangspunkt i livssykelen til systemet og bruker dette som forklaringsmodell. Senere i rapporten blir det benyttet usecase-analyse for å forklare kodifiserte systemer på en annen måte.

Livssykelen til kodifiserte systemer kan beskrives på mange måter, som vist i kapittel 3 i [AG04]. Den livssykelen som legges til grunn her har 5 steg:

1. Innse behov og igangsetting av arbeid
2. Kunnskapsbygging
3. Organisering av kunnskapen
4. Distribusjon og lagring av kunnskapen
5. Vedlikehold og utvidelse av kunnskapsbasen

De neste avsnittene vil ta for seg punktene2 og 3. Da hoveddelen av denne rapporten vil dreie seg om punkt4 og5 vil dette ikke vektlegges her.

Det skal likevel startes med en kort innføring til punkt 1. Igangsetting av arbeid med nye kunnskapssystemer kan skje av flere grunner. En viktig grunn er å ta vare på kunnskapsarbeideres viten slik at den forblir i organisasjonen selv om personen slutter. En annen grunn er å unngå å ”finne opp hjulet på nytt” i forskjellige deler av en større organisasjon. Enklere opplæring, større tilgjengelighet og større sikkerhet er andre faktorer som gjør det interessant å bygge et kodifisert kunnskapssystem.

Kunnskapsbygging Den sentrale prosessen i utviklingen av kunnskapssystemer er kunnksapsinnsamlingsprosessen. Informasjonen som skal inn i systemet kan gjerne allerede være beskrevet i papirutgaven av prosedyrene. Det å ha sin kunnskapsbase lagret i permer medfører en del problemer med tilgjengelighet, distribusjon av nye versjoner, utforming og så videre. I andre tilfeller er kunnskapen kun lagret hos de arbeiderne som til enhver tid gjør jobben. Dette medfører problemer når noen forsvinner. I begge tilfeller er det nødvendig å samle inn kunnskapen før den kan organiseres, forbedres og distribueres elektronisk.

Når kunnskapen som skal presenteres allerede er lagret i prosedyrer er mye av informasjonsinnsamlingen gjort. Det kan likevel gjenstå utfordringer. En typisk utfordring er at forskjellige arkiver har ulike versjoner av prosedyrene og at det er en

viss differanse mellom disse. I slike tilfeller er det viktig å validere informasjonen ved hjelp av kunnskapsbærende arbeidere. I tillegg til den allerede nedskrevne informasjonen er det oftest interessant å benytte informasjonsteknologiens muligheter til å beskrive objekter og oppgaver med multimedia. Dette krever gjerne ekstra informasjonsinnsamling. En kan ta bilder av forskjellige deler, filme eller animere arbeidssekvenser og lage interaksjon med brukeren for å øke dens deltagelse. Dette er oppgaver som må løses i samarbeid med domeneeksperter for å forsikre at alt blir riktig.

I en bedrift som ønsker å kodifisere kunnskap vil en fort oppleve at det finnes et stort kognitivt gap mellom eksperter og nybegynnere/systemutviklere [HP03]. Dette kognitive gapet skyldes i all hovedsak to faktorer. For det første har en ekspert lang erfaring og evner å abstrahere på en måte som en nybegynner ikke kan. For det andre har en ekspert utviklet masse underforstått kunnskap, det vil si kunnskap som for han[1] er så naturlig at han ikke anser det som nødvendig å uttale den. Disse to utfordringene er viktige å komme over når en skal utvikle verktøy. Det er viktig å fange opp all relevant informasjon rundt en prosess og senere kunne framlegge den på en måte som også uinvidde kan forstå. Dette krever altså godt samarbeid med ekspertene og at den som samler informasjonen kan formidle den videre på en måte tilpasset målgruppen.

Organisering av kunnskap I kunnskapsbyggefasen samles store datamengder. Disse lagres gjerne med en viss formasjon ved at data om forskjellige arbeidsoppgaver lagres hver for seg. Dette skaper et grunnlag for å strukturert organisere kunnskapen slik at ulike enheter kan lagres hver for seg. Arbeidsoppgaver kan også kalles prosedyrer og disse er oftest beskrevet steg for steg, noe som også gir form til hver enkelt enhet. Hvert steg i prosedyren kan også beskrives mer detaljert. Alt dette danner en intuitiv oppdeling og organisering av bedriftens viten når det applikeres i det kodifiserte systemet. Det vises hvordan dette kan gjøres i kapittel 3.2.1 på side 45.

Distribusjon og lagring av kunnskap Den organiserte kunnskapen skal tas vare på og leveres til brukere av systemet når de trenger den. Dette innebærer at det må finnes et datalager som effektivt tar vare på ulike dataformer.

Et aspekt ved lagring av ulike dataformater er hensynet til interoperativitet. Kunnskap lagret i et bestemt dataverktøy kan være av interesse også for andre systemer. Det er derfor viktig å ha et velspesifisert grensesnitt for aksessering av data samt en protokoll for overføring som enkelt kan tolkes av ulike systemer.

[1] *Han* bør leses som *hun eller han* gjennom hele dette dokumentet

Som nevnt over bør arbeidsprosesser kunne forklares ved hjelp av et rikt spekter av medietyper. Dette kan dreie seg om ren tekst, grafikk, bilder, animasjon, lyd og film med mere. Innen hver av disse medietypene finnes det ulike standarder og implementasjoner. En utfordring er altså å lagre disse på en slik måte at brukeren får minimal ventetid og maksimal kvalitet og handlefrihet. Dette er problemstillinger som tas opp i kapittelet 3.3.1 på side 58 og blir dermed ikke diskutert i mer detalj her.

Vedlikehold og utvidelse Et prosedyresystem er aldri ferdig. Utstyr som benyttes blir skiftet ut, sikkerhetsprosedyrer endres og en finner mer effektive måter å utføre oppgaver på. Derfor vil spesifikasjon av beste praksis kontinuerlig være åpen for endring. Dataverktøyet må ta hensyn til dette og ha velspesifiserte oppdateringsrutiner. I tillegg til at rutiner oppdateres vil det også være nødvendig å legge til nye rutiner. Awad [AG04] beskriver et kunnskapsledelsessystem som ”start slow and grow” i motsetning til tradisjonelle systemer som er ”specify then build”.

I tillegg til at kunnskapen endrer seg over tid vil også bruksmønster og antall brukere kunne endre seg. Som et resultat kan det være nødvendig å skalere systemet for å forebygge flaskehalser og dermed gjøre systemet bedre tilgjengelig. Dette må tas hensyn til når en utvikler systemet. Hvordan dette kan gjøres blir beskrevet i kapittel 3.3.2 på side 62

Informasjon om kodifiserte systemer er i hovedsak hentet fra del III i [AG04]. I tillegg er det tatt utgangspunkt i [Dåv03] og innspill fra veiledere.

2.1.3 Arbeidsprosesser

Et sentralt begrep som er med å danne bakgrunn for denne rapporten er arbeidsprosessen. I dette kapittelet blir en del aspekter ved arbeidsprosessene kartlagt og forklart.

Terminologi

Dokumentet [Dåv03] beskriver ”arbeidsprosessorientert styring”, og hvordan dette kan settes ut i livet som flytdiagrammer. Dette er en måte å organisere kunnskap om arbeidsoppgaver på. En rekke definisjoner danner grunnlaget for å kunne representere arbeidsprosesser.

Beste Praksis (BP) - Den til en hver tid beste måten å utføre en oppgave på som er kjent i organisasjonen. Avvikende utførelse skal kvalitetssikres i den enkelte organisasjonsenhet. Eies av prosesseier eller fagenhet, men utførende skal tungt involveres i erfaringstilbakeføring. Ansvarliggjøre den utførende.

Krav - Kravdokument beskriver hvilke tiltak som må gjøres og sikkerhetsforanstaltninger som må være til stede når en arbeidsoppgave løses. Et krav kan ikke fravikes med mindre det er gitt avvikstillatelse fra styringselementet som ”eies” av prosesseier eller en fagenhet.

Prosess - Samling av beslektede eller samvirkende aktiviteter som omformer tilført grunnlag (input) til resultater (output).

Subprosess - Delmengde av en prosess som kan skilles ut og stå for seg selv slik at den kan inngå i flere ulike prosesser.

Aktivitet - Er det laveste nivået i en arbeidsprosessbeskrivelse og kjennetegnes av at den er entydig mht. roller, dvs. det er kun en aktør involvert i en aktivitet. En aktivitet ”eies” av prosesseier.

Rolle - Er knyttet til aktivitet og beskriver aktøren som er involvert i aktiviteten. Rolle er ikke det samme som stilling. Det (kan) stilles kompetansekrav til rollen. En rolledefinisjon eies av prosesseier. Kompetansekrav til denne eies av prosesseier eller fagsjef.

Produkt - Resultat av en prosess.

Verdikjede - Samling av avhengige aktiviteter som skaper verdier for hverandre.

Tall og størrelser

Når arbeidsprosesser skal beskrives er det nyttig med et begrep om omfang og datamengder en kan forvente å jobbe med. I tabell 2.1 på neste side ser vi at det finnes flere ulike typer dokumentasjon. I arbeidsprosess-sammenheng arbeider en med den styrende dokumentasjonen. I en database med styrende dokumentasjon vil det være datamengder i størrelsesorden gigabyte[2].

Informasjon om arbeidsprosesser må holdes oppdatert til enhver tid for at det skal være interessant for brukerne å benytte seg av denne informasjonen. I et

[2]Tall oppgitt av Espen A. Eidem i SoluDyne AS.

Type	Innhold	Antall (størrelsesorden)
Styrende dokumenter	Kvalitetssikret multimedia-adata	10^4
Grunnlagsdokumenter	Tekstdokumenter, skisser o.l.	10^5
Rådata	Dataenheter i matriser o.l.	10^6

Tabell 2.1: Dokumenter i arbeidsprosessystemer[2]

fungerende arbeidsprosessystem kan det forekomme hundrevis av endringer hver eneste dag[3].

Presentasjon av arbeidsprosesser

I dag brukes for det meste en av to måter å presentere arbeidsprosesser på. Den ene benytter dokument-tankegangen og fremlegger arbedsflyt punkt for punkt slik at det kan printes på et stykke papir og tas med ut på arbeidsplassen.

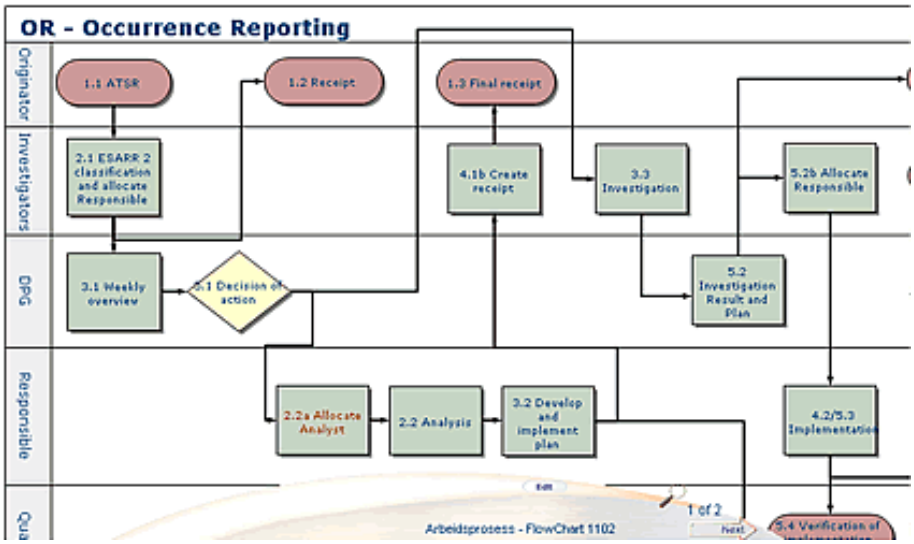
Den andre måten søker å utnytte at den mennesklige hjernen er svært flink til å behandle grafikk og se sammenhengen mellom noen få enheter. Det er derfor vanlig å lage grafiske grensesnitt som forklarer gangen i en arbeidsprosess. Hyperlenker brukes til å finne mer detaljert informasjon.

Et arbeidsflytdiagram er en slik måte å organisere en arbeidsprosess på. Ved hjelp av ulike bokser og en rolleinndeling kan gangen og ansvarsforhold i prosessen enkelt beskrives. Et eksempel på flytdiagram finnes i figur 2.3. Boksologien forklares under.

Ellipse beskriver start- og sluttpunkter i en verdikjede. Sluttpunktene representerer klare mål for arbeidsprosessen. Målet er felles for alle aktivitetene og må derfor være synlig.

Pil representerer en oppgave. Oppgaven genereres av et startpunkt eller en aktivitet og terminerer i en aktivitet eller et sluttpunkt. En aktivitet er avhengig av at foregående aktiviteter er ferdige før denne kan igangsettes.

[3]En av de største utfordringene i praktiske kunnskapsledelsessystem er å holde dem oppdatert[HdW02, Hol04]. Kontaktpersoner vi har i Sintef påpeker også at dette er en hovedutfordring for arbeidsprosessorienterte systemer.



Figur 2.3: Arbeidsflytdiagram hentet fra SoluDyne 2.05 [SOS]

Rektangel beskriver en aktivitet. En aktivitet kan ha en eller flere oppgaver den skal løse (piler inn) og har en pil ut. Det er vanlig at en aktivitet støttes av mer detaljerte beskrivelser. Dette kan være subprosesser, tekst og/eller multimedia. For en aktivitet må det forklares flere aspekter:

- *Hvorfor* aktiviteten gjennomføres. Dette klargjør mål og krav som stilles til utførelsen.
- *Hvordan* aktiviteten skal gjennomføres. Beskrives i henhold til beste praksis.
- *Hva skjer om* noe går galt i aktiviteten. Beskriver mulige avvik og deres konsekvenser. I forbindelse med avvik må det være kartlagt og beskrevet egene prosedyrer for avviksbehandling.

Rombe (diamanten) er en valgboks. Til en valgboks går det en pil inn. Denne pilen bærer en verdi. Valgboksen inneholder et spørsmål og svarene er ja eller nei. Prosessen følger så en av stiene videre.

Rollebokser er avgrenset av de horisontale linjene. Helt til venstre står det en kode som forklarer hvilke roller som er inkludert. En rolle er ansvarlig for alle de aktivitetene som ligger innenfor sitt rektangel.

2.2 Multimediasystemer

Med begrepet *multimediasystemer* menes som oftest datasystemer som har muligheter for å formidle levende bilder og lyd av høy kvalitet. Med innføringen av cd-rom og lydkort på hjemmedatamaskinene på 90-tallet ble multimedia et ord som beskrev nettopp dette.

Med multimedia menes også bilder og grafikk, der man med bilder mener at de er beskrevet som punkter i et rutenett, mens med grafikk menes geometrisk beskrevet informasjon som også kan være tredimensjonal. Grafikk kan også inneholde bevegelse, den vil da kalles *animasjon*.

Lyd og video/animasjon skiller seg fra tekst og bilder/grafikk ved at de har et såkalt temporalt aspekt, det vil si at de har et forhold til tid. I tillegg til den nye dimensjonen dette medfører for hver enkelt mediatype kommer vanligvis synkronisering av flere parallelle datastrømmer; strømmene skal ikke bare spilles av i riktig hastighet, men de må også ha en mekanisme for å sikre at de befinner seg på samme og ønsket plassering på tidslinjen slik brukeren oppfatter det.

Selv om lagring og overføring av data er områder der utviklingen stadig gjør fremskritt kan man si at disse ikke er hindringer for dagens multimediasystemer; på en typisk harddisk kan man lagre 350 timer video med kvalitet som overgår vanlige TV-sendinger, for så å overføre denne i sann tid hjem til folk over eksisterende telefonlinjer[4].

De nye utfordringene innenfor behandling av multimediadata ligger i søking, indeksering og tolking av de forskjellige mediatypene, gjenkjenning av objekter i bevegelse i video, gjenkjenning av tale og stemme, spatiale spørringer i bilder og video. Et eksempel kan være å be en database returnere alle scener fra en film der en navngitt skuespiller står til venstre for en annen navngitt skuespiller, nevner en annen rollefigurs navn og forsvinner ut av bildet til venstre, eventuelt kan brukeren få et lite vindu der han fort kan tegne opp en form som han ønsker at databasen skal lete etter i filmbiblioteket men returnere kun de klippene der en mann med vernehjelm vises i samme bilde. Et mål for slike systemer vil være at de selv klarer å trekke ut metainformasjon av dataen de forvalter, for eksempel å kjenne igjen objekter i en filmsekvens og skape spatial kunnskap om innholdet basert på det. Enkelte systemer baserer seg på at brukere skal bidra med å markere og annotere istedenfor at dette gjøres automatisk [D+03].

[4] Omtrentlige tall, basert på 1,5 Mbps mpeg-4 asp, 250GB harddisk og adsl med datarate 1,5Mbps eller mer.

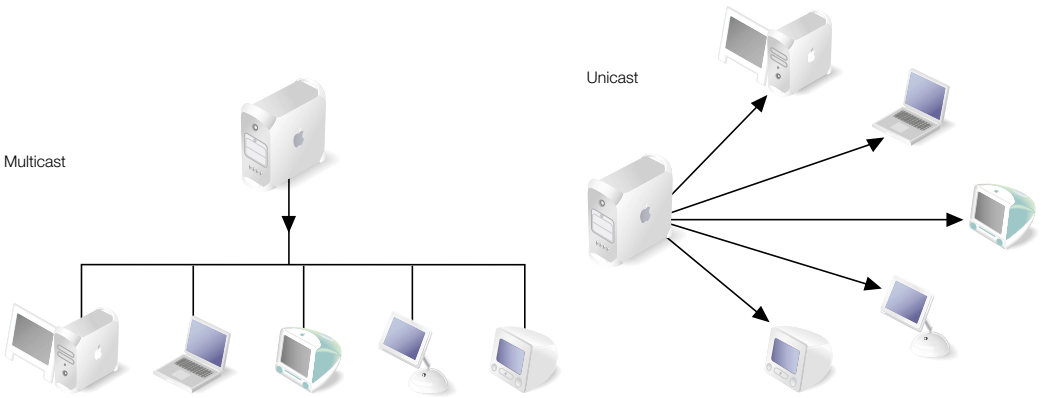
Overføring av temporale mediatyper

I et system som inneholder et mediabibliotek er alle mediatyper lagret som filer i et filsystem eller objekter i en database. I mange tilfeller er brukeren ikke interessert i hele filen dersom denne har et temporalt aspekt, men kun en spesiell del. Det finnes flere strategier for overføring av disse temporale mediatypene, alt etter hva målet for overføringen er:

Progressiv nedlasting vil si at filen overføres i sin helhet til mottakeren over http akkurat som et bilde på en webside. Den lagres på disk hos brukeren og kan spilles av derfra. Dersom avspilleren og filtypen tillater det kan avspilling begynne med en gang den første dataen er overført. Dette kalles progressiv nedlasting. Man forutsetter at resten av filen lastes fortere enn avspillingen skjer. Om den ikke gjør det kan avspilleren beregne hvor stort forsprang nedlastingen trenger å få før avspillingen kan begynne, slik at de begge blir ferdige samtidig. For eksempel kan avspilleren beregne at hvis avspillingsdataraten er 120kB/s og overføringsraten er 60kB/s må den vente til halve filen er overført før sammenhengende avspilling kan begynne. På denne måten blir filen lagret på brukerens maskin, og kan vilkårlig spilles av ved senere anledninger uten at det belaster serveren.

Multicast streaming vil si at serveren sender kun én strøm av data ut på nettet som så routes til klientene som abonnerer på strømmen, se til venstre i figur 2.4 på neste side. Dette skalerer bra med tanke på antall klienter man kan betjene, men krever en spesiell nettverkstilgang [App02]. Klientene har ikke anledning til å velge et vilkårlig avspillingspunkt, men mottar akkurat det samme til samme tid som alle andre som er knyttet til strømmen, på samme måte som tradisjonelt fjernsyn eller radio. Derfor passer også multicast streaming godt til direkte overføring, der man uansett bare er interessert i å følge det som skjer akkurat der og da.

Unicast streaming vil i kontrast til multicast si at hver enkelt klient har sin egen dedikerte strøm gående ut fra serveren og kan spille av forskjellig innhold, se til høyre i figur 2.4 på neste side. Gitt gode nettverksforhold og tilstrekkelige serverressurser vil dette langt på vei være sammenlignbart med avspilling av lokale filer. Avspillingen kan stoppes midlertidig, og avspillingspunkt kan velges vilkårlig. I likhet med multicast streaming og i motsetning til progressiv nedlastning foregår dette over egne protokoller, og ikke over http. Dette kan falle inn under restriksjonene i routere og brannmurer og således være utilgjengelig for tilfeldige brukere.



Figur 2.4: Multicast og Unicast Streaming, tatt fra [App02]

For en grundig innføring på dette feltet se [KA97, App02, KWW00, SGV02, HGL+03]

2.3 Bruksanalyse av kodifisert system

Rapporten har til nå tatt utgangspunkt i teori om kunnskap, læring og verktøy for elektronisk formidling av kunnskapen. I den neste delen tar vi et steg videre ved å avdekke krav som stilles til dataverktøy for elektronisk behandling av dynamiske arbeidsprosesser. Det første steget tar utgangspunkt i usecase-analyse. Usecase er en del av den omfattende UML-standarden [OMG].

”Usecase er en teknikk for å fange opp de funksjonelle kravene til et system. Usecase beskriver den typiske interaksjonen mellom brukeren og systemet og gir en forklaring på hvordan systemet brukes” [Fow04]. I denne delen presenterer vi et usecase-diagram som viser den fulle funksjonaliteten til et generelt arbeidsprosessverktøy, se figur 2.5 på side 20. Bakgrunnen og begrunnelsen til hver rolle og hvert usecase vil forklares.

Denne analysen har vi gjort for et generelt dataverktøy for behandling av arbeidsprosesser. Bakgrunnen for systemet bygger på teorien om kodifiserte systemer i avsnitt 2.1.2. Ved siden av dette er det gitt innsyn i et case benyttet i et internt prosjekt ved Sintef, ”Resertifisering av PZV”. Arbeid med dette har gitt viktig innsikt og forståelse for hvem som interagerer med systemet og hvordan dette foregår. Vi kommer tilbake til dette prosjektet i casestudiet i kapittel 4.4

2.3.1 Aktører

Aktører i et usecase beskrives som ”en rolle som en bruker spiller i forhold til systemet” [Fow04]. Aktøren må ikke være et menneske, men kan også være et eksternt system som ber om tjenester fra systemet. Før en går inn i de konkrete usecasene bør en ta for seg de aktørene som involveres. Når en forstår hvilke roller disse har er det enklere å lese usecasene. Aktørene beskrives her:

Arbeider er involvert i den faktiske utførelsen av arbeidet på en oljeplattform, i en fabrikk eller et annet sted i organisasjonen. For at arbeideren skal kunne utføre sin jobb på en god måte må han følge bedriftens beste praksis som er skrevet ned i arbeidsprosessene.

Arbeidsprosessansvarlig har ansvaret for at arbeidsprosessene lagret i systemet er i henhold til bedriftens for øyeblikket beste praksis.

Databasen er den grunnleggende bærebjelken i informasjonsbehandlingsverktøyet. Den kan bestå av en eller flere servere og må kunne behandle forskjellige typer data raskt og effektivt.

Systemansvarlig har ansvaret for at informasjonssystemet der arbeidsprosessene er lagret er tilgjengelig til enhver tid. Derfor må han jevnlig sjekke status på systemet, behandle feilmeldinger og gjøre eventuelle endringer for at systemet skal virke som det skal.

Tilbakemelder gir rapport etter utført arbeidsprosess (eller etter flere utførte arbeidsprosesser). Rollen innehas av fagarbeidere eller andre som har direkte kontakt med arbeidsprosessen. En tilbakemelder skal si ifra om noe har gått galt under gjennomføringen og om det bør gjøres endringer i beste praksis.

2.3.2 Usecase-analyse

En usecase-analyse består av flere deler. Vi har til nå sett på hva usecaser kan gjøre og gjort oss kjent med de ulike aktørene som inngår i denne analysen. Dette avsnittet vil gi en overordnet oversikt over usecasene og en introduksjon til den videre analysen som skjer i neste avsnitt.

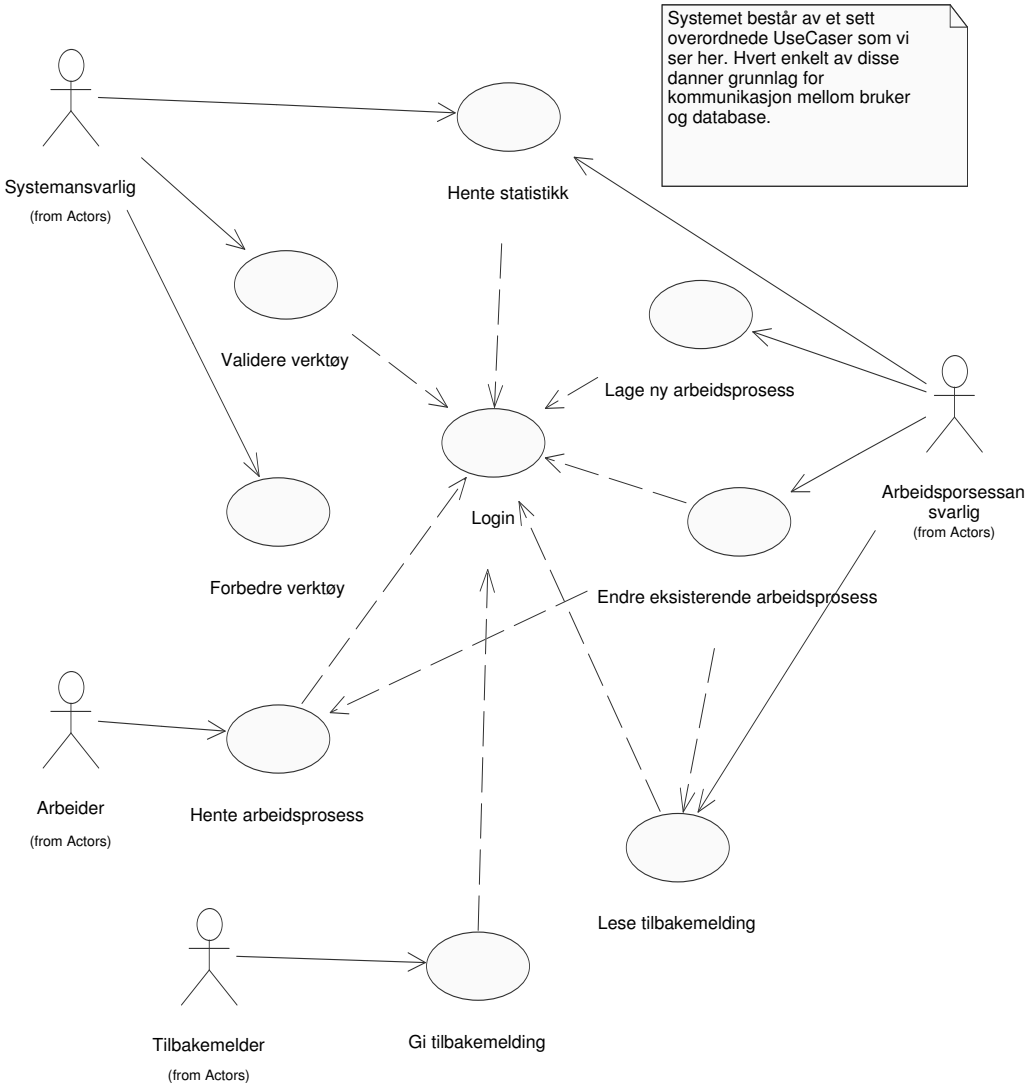
Et usecase-diagram kan sammenlignes med en innholdsfortegnelse for et systems ulike funksjoner. Det viser hvilken aktør som medvirker i hvilke handlinger og hvilke usecaser som inkluderer andre. Det gir en ganske tydelig avgrensning for hva systemet gjør. Det fungerer altså som et fint utgangspunkt for å diskutere systemets innhold. Figur 2.5 på neste side er et usecase-diagram og viser systemet som et sett av flere usecaser og de allerede beskrevne aktørene. Den neste delen vil gi en beskrivelse av hensikten med de ulike usecasene.

Et sekvensdiagram viser interaksjon mellom aktører og systemet. I diagrammene finnes to akser. Den horisontale aksen skiller mellom objekter slik at vi grafisk ser hvilket objekt som utfører en handling. Den vertikale aksen viser tidslinja. En interaksjon starter på toppen av diagrammet og går nedover. Dette støttes også ved at hver aksjon er beskrevet med nummer og forklarende navn. I denne delen av analysen tas det utgangspunkt i [Fow04] og Rational Rose sine muligheter og begrensninger [Cor00].

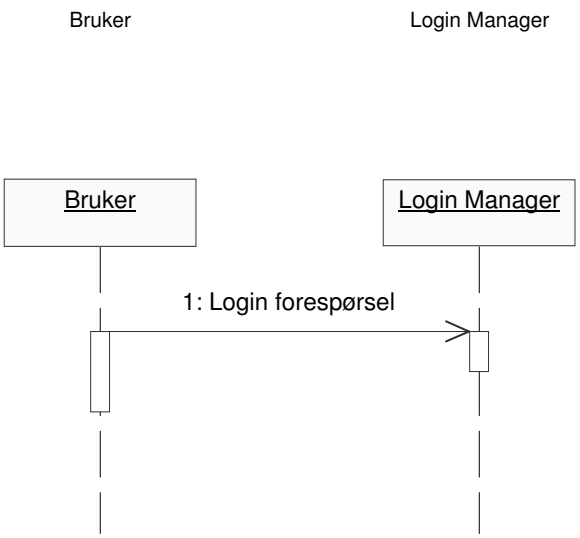
I sekvensdiagrammene er det valgt å se på standardinteraksjonen. De skal virke som et utgangspunkt for videre diskusjon og utvikling av usecasene slik at de kan klargjøres for implementasjon. Feilsituasjoner blir ikke presentert i disse.

Det kan også nevnes at notasjon for hvilke data som sendes med hvert kall og notasjon for svar ikke implementert i diagrammene. Dette betyr ikke at det ikke sendes svar, men at svarene kommer implisitt i kallet.

De tekstlige usecasene beskriver steg for steg interaksjonen mellom brukeren



Figur 2.5: Usecase-diagram med alle aktører og den viktigste interaksjonen med systemet



Figur 2.6: Interaksjon mellom bruker og system i usecaset ”Login”

og systemet. Disse har, i tillegg til sekvensdiagrammenes standardinteraksjon, beskrivelse av mulige feilsituasjoner og hvordan systemet skal reagere i slike situasjoner.

2.3.3 Beskrivelse av usecasene

I denne delen blir alle systemets usecaser beskrevet i detalj. For å gjøre dette benyttes sekvensdiagrammer og tekstlige usecaser som beskrevet over.

Login

Brukeren må logge inn slik at systemet kan registrere hvem som bruker det og vise det grensesnittet brukeren trenger for å utføre sine oppgaver. Samtidig skal dette sikre systemet mot feilbruk. Usecaset er beskrevet i figur 2.6 og tabell 2.2 på neste side.

Login

Hovedscenario

1. Brukeren ber om ny loginmanager
2. Systemet genererer loginmanager
3. Brukeren gjør en loginforespørsel
4. Systemet bekrefter denne og gir videre tilgang til systemet

Utvidelser

- 2a Systemet kan ikke generere loginmanager
  1. Systemet gir feilmelding og avsluttes
- 4a Systemet kan ikke gi tilgang
  1. Systemet viser feilmelding
  2. Brukeren prøver på nytt eller avbryter

Tabell 2.2: Tekstlig beskrivelse av usecaset ”Login”

Lage ny arbeidsprosess

Som beskrevet tidligere er kodifiserte systemer ofte ”start slow and grow”. Det vil altså i hele systemets levetid være interessant å kunne utvide med nye arbeidsprosesser. Dette gjøres av arbeidsprosessansvarlig. Når han går inn i systemet har han allerede en klar formening om hvordan prosessen skal være og alle data (filmer og slikt) er tilgjengelige. Oppgaven blir da å konfigurere dette på en riktig måte og laste opp data. Verktøyet for å bygge og endre arbeidsprosesser brukes til dette. Bruken beskrives i dette usecaset. Det er beskrevet i figur 2.7 på side 25 og tabell 2.3 på neste side.

Hente Arbeidsprosess

Arbeideren vil stadig ha behov for å oppdatere seg på de ulike arbeidsprosessene han er involvert i. Han kan utføre dette usecaset i forbindelse med opplæring eller i forkant av arbeidet. Uansett når han skal hente arbeidsprosessen er målet å lese og forstå denne. En arbeidsposess vil være beskrevet av flytdiagrammer på flere nivåer, men kan presenteres på ulike måter alt etter hvilket medium som brukes for aksess. Arbeideren er ofte interessert i å fordype seg i en eller flere mindre deler av prosessen. Dette usecaset beskriver prosessen fra arbeideren søker etter en arbeidsprosess fram til han har fordypet seg ned til den aktiviteten som er av interesse. Usecaset er beskrevet i figur 2.8 på side 27 og tabell 2.4 på side 26.

Gi Tilbakemelding

Arbeidsprosesser er gjenstand for stadig revisjon og endring. En grunn til dette er at tilbakemelderen har innvendinger til dagens beste praksis for en arbeidsprosess. Systemet har da et eget grensenitt slik at han kan formidle sine kommentarer og forslag videre. Disse tilbakemeldingene blir så formidlet til den eller de som myndighet til å vurdere og endre beste praksis. Usecaset er beskrevet i figur 2.9 på side 29 og tabell 2.5 på side 28.

Lese tilbakemelding

Å lese tilbakemeldingen er en oppgave arbeidsprosessansvarlig utfører når det har kommet nye tilbakemeldinger. Systemet har beskjed om dette og det er da

Lage ny arbeidsprosess

Forutsetning:

- Arbeidsprosessansvarlig har alle nødvendige data tilgjengelig
- Arbeidsprosessansvarlig har logget inn

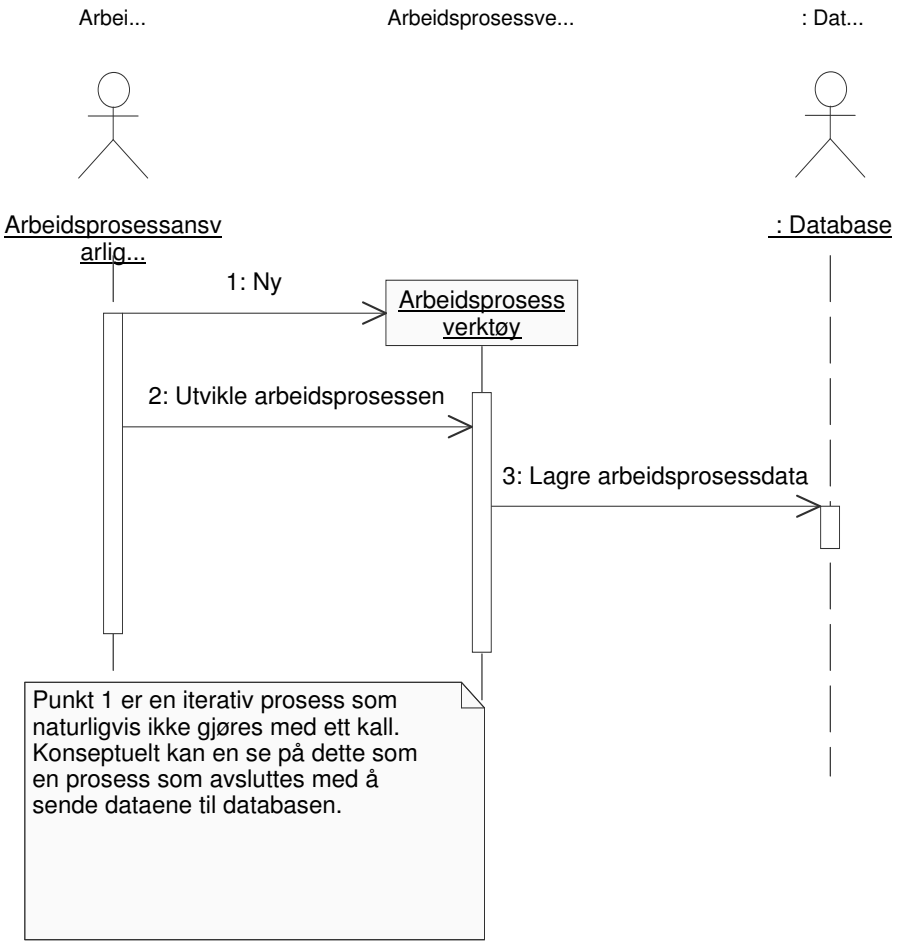
Hovedscenario

1. Brukeren ber om nytt arbeidsprosessverktøy
2. Systemet genererer arbeidsprosessverktøyet
3. Brukeren utvikler arbeidsprosessen
4. Brukeren bekrefter arbeidsprosessen
5. Systemet lagrer arbeidsprosessen
6. Systemet gjør arbeidsprosessen tilgjengelig

Utvidelser

- 2a Systemet kan ikke generere arbeidsprosessverktøy
    1. Systemet gir feilmelding og avslutter
- 5a Systemet kan ikke lagre arbeidsprosessen
    1. Systemet viser feilmelding
    2. Brukeren prøver på nytt eller avbryter

Tabell 2.3: Tekstlig beskrivelse av usecaset ”Lage ny arbeidsprosess”



Figur 2.7: Interaksjon mellom bruker og system i usecaset ”Lage ny arbeidsprosess”

Hente Arbeidsprosess

Forutsetninger: Brukeren er logget inn

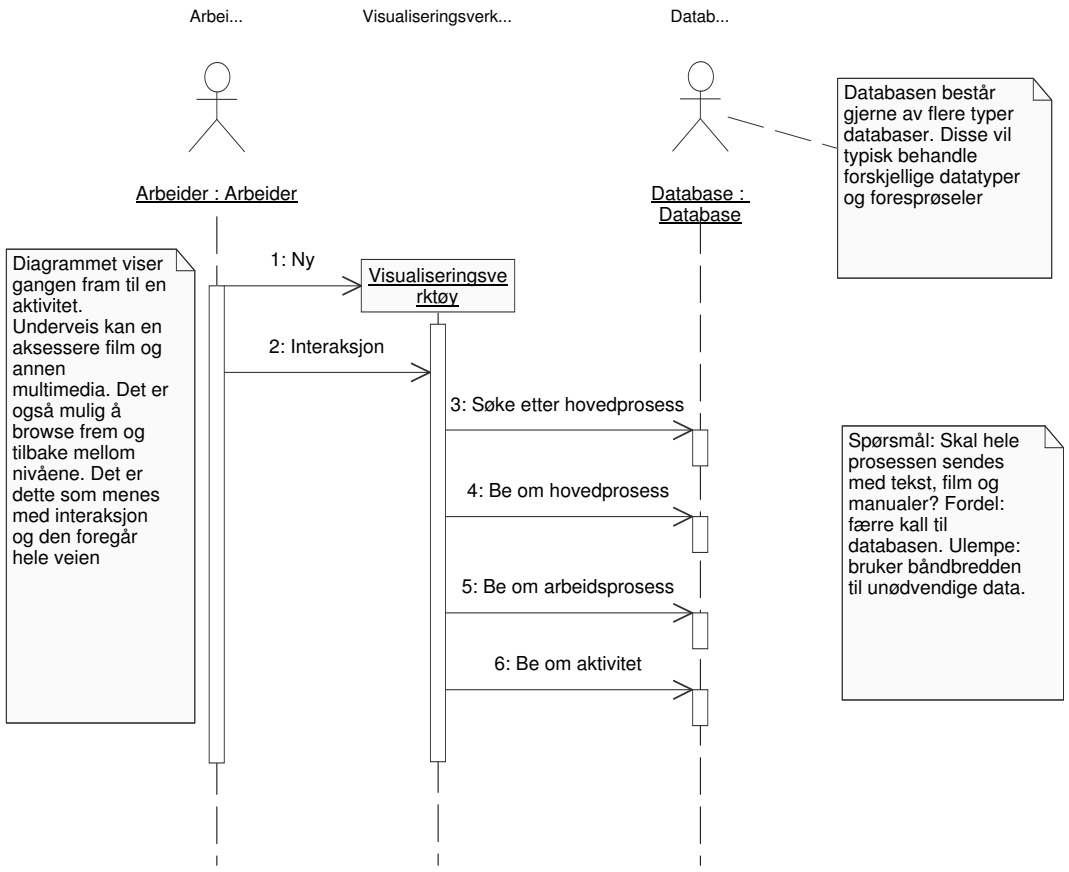
Hovedscenario

1. Brukeren ber om nytt visualiseringsverktøy
2. Systemet genererer visualiseringsverktøy
3. Brukeren søker etter en hovedprosess
4. Systemet gir en liste med hovedprosesser
5. Brukeren velger en hovedprosess
6. Systemet viser prosessen
7. Brukeren fordyper seg i en av hovedprosessens subprosesser
8. Systemet viser subprosessen
9. Brukeren fordyper seg i en av subprosessens aktiviteter
10. Systemet viser aktiviteten

Utvidelser

- Ved hver fordypning kan brukeren velge å gå opp ett nivå i steden.
- Mellom hver fordypning kan brukeren velge å se informasjon ifm. entiteter.

Tabell 2.4: Tekstlig beskrivelse av usecaset ”Hente arbeidsprosess”



Figur 2.8: Interaksjon mellom bruker og system i usecaset ”Hente arbeidsprosess”

Gi tilbakemelding

Forutsetninger:

- Tilbakemelder har lest arbeidsprosessen han gir tilbakemelding på
- Tilbakemelder har logget inn i systemet

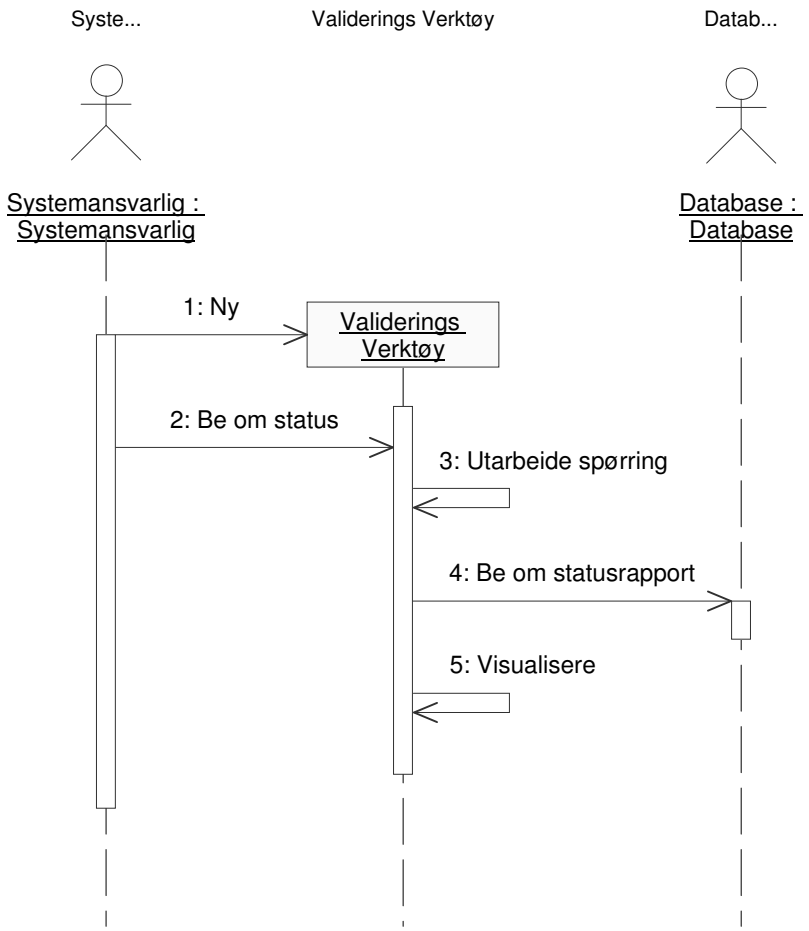
Hovedscenario

1. Brukeren oppretter ny tilbakemeldingsmodul
2. Systemet viser grensesnittet
3. Brukeren fyller inn tilbakemeldingene, tilbakemeldingen inneholder prosessidentifikator.
4. Systemet lagrer tilbakemeldingene i databasen
5. Systemet sender beskjed til arbeidsprosessansvarlig

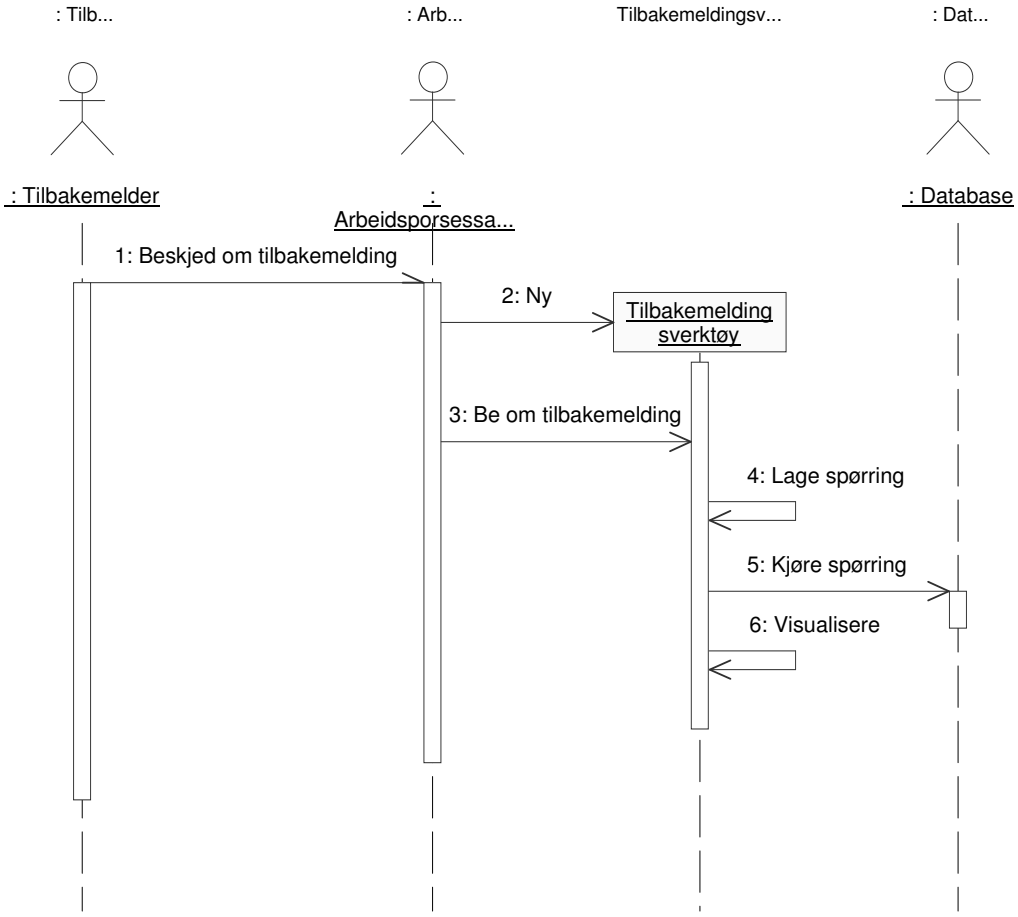
Utvidelser

- 2a Systemet kan ikke generere tilbakemeldingsmodul
  1. Systemet viser feilmelding og avslutter
- 4a Systemet kan ikke lagre tilbakemelding
  1. Systemet sender feilmelding
  2. Brukeren besvarer feilmeldingen ved å prøve igjen eller avbryte

Tabell 2.5: Tekstlig beskrivelse av usecaset ”Gi tilbakemelding”



Figur 2.9: Interaksjon mellom bruker og system i usecaset ”Gi tilbakemelding”



Figur 2.10: Interaksjon mellom bruker og system i usecaset ”Lese tilbakemeldinger”

ansvarliges ansvar at tilbakemeldingen blir behandlet på riktig måte. Behandlingen kan ende opp i endring av eksisterende arbeidsprosess eller at tilbakemeldingen blir refusert og intet blir endret i arbeidsprosessene. Systemet har et grensesnitt som gjør det enkelt å lese og behandle tilbakemeldinger, bruk av dette beskrives her. Usecaset er beskrevet i figur 2.10 og tabell 2.6 på neste side.

Endre eksisterende arbeidsprosess

Det finnes mange ulike årsaker til at en prosessbeskrivelse skal endres. Det kan komme av teknologiutvikling, altså at utstyret eller andre forutsetninger har endret

Lese tilbakemelding

Forutsetninger: Brukeren er logget inn

Hovedscenario

1. Brukeren mottar beskjed om tilbakemelding
2. Brukeren ber om nytt tilbakemeldingsverktøy
3. Systemet genererer tilbakemeldingsverktøy
4. Brukeren ber om en gitt tilbakemelding
5. Systemet utarbeider en SQL-spørring
6. Systemet kjører spørringen
7. Systemet visualiserer resultatet

Utvidelser

- 1a Brukeren bestemmer seg selv for å se på tilbakemeldinger
  1. Gå direkte til punkt 3
- 2a Systemet kan ikke generere tilbakemeldingsverktøy
  1. Systemet gir feilmelding og avsluttes
- 6a Systemet kan ikke kjøre spørringen
  1. Systemet viser feilmelding
  2. Brukeren prøver på nytt eller avbryter

Tabell 2.6: Tekstlig beskrivelse av usecaset ”Lese tilbakemelding”

seg. Det kan også hende at noen har gitt en god tilbakemelding med konstruktive forslag til endringer. Uansett hvorfor må arbeidsprosessansvarlig ha muligheten til å gjøre endringer i en allerede eksisterende arbeidsprosess. Til dette må systemet ha et eget grensesnitt som kan hente opp den gamle prosessen og gjøre endringer på denne. Den nye må så kunne lagres slik at gammel versjon fortsatt er mulig å se, men blir klassifisert som utdatert. Usecaset er vist i figur 2.11 på side 34 og tabell 2.7 på neste side.

Forbedre Verktøy

Dersom det skal gjøres oppgraderinger av programvarestrukturen i systemet må systemansvarlig kunne gjøre det på en effektiv måte. Optimalt sett skal ikke systemet ha nedetid på grunn av slike endringer og i det tilfellet kan det trenges egne moduler for å gjøre slikt vedlikehold. Interaksjonen beskrives i dette usecaset. Operasjonen er beskrevet i figur 2.12 på side 35 og tabell 2.8 på side 35.

Hente Statistikk

Det kan være av interesse for både arbeidsprosessansvarlig og systemansvarlig å vite noe om bruksmønsteret til systemet. For prosessansvarlig er det viktig å vite hvorvidt hans prosess benyttes, når og av hvem. Systemansvarlig kan bruke statistikken til å finne generelle driftsdata han kan bruke i sitt arbeid. Det vil finnes et grenesnitt i systemet som kan brukes til grafisk og med tallmateriale å vise statistikk produsert på bakgrunn av bruk av systemet. Dette usecaset viser interaksjonen som er nødvendig for å få ut statistikken. Det er beskrevet i figur 2.13 på side 37 og tabell 2.9 på side 36.

Validere Verktøy

Er usecaset systemansvarlig utfører for å holde kontroll på at maskinvare i systemet fungerer som det skal. Her vil han kunne finne feilmeldinger og indikatorer på hvilke deler av systemet som fungerer normalt og hvilke som ikke gjør det. Det vil finnes et grensesnitt som kan indikere hvor feil befinner seg, feilmeldinger i den forbindelse og mulige forslag til løsninger. Interaksjonen med systemet beskrives i figur 2.14 på side 39 og tabell 2.10 på side 38.

Endre eksisterende arbeidsprosess

Forutsetning: Brukeren er logget inn

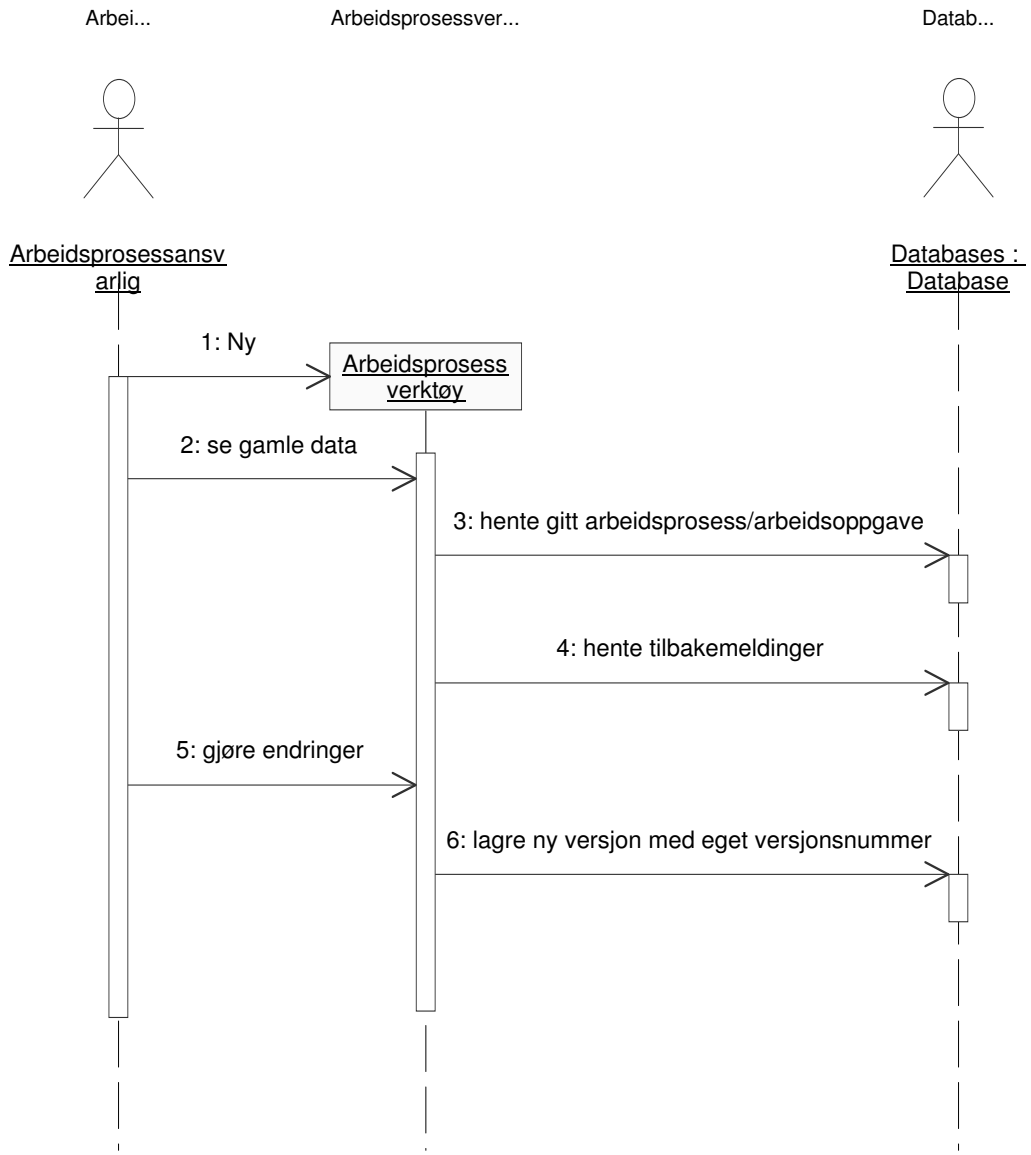
Hovedscenario

1. Brukeren oppretter ny instans av arbeidsprosessverktøy
2. Systemet visualiserer arbeidsprosessverktøy
3. Brukeren søker opp den gamle arbeidsprosessen, se ”hente arbeidsprosess”, figur 2.8 på side 27
4. Systemet leverer alle forespurte data
5. Brukeren leser tilbakemeldinger, se ”lese tilbakemeldinger”, figur 2.10 på side 30
6. Systemet viser tilbakemeldinger
7. Brukeren gjør endringene og bekrefter disse
8. Systemet lagrer endringene med eget versjonsnummer

Utvidelser

- 2a Systemet kan ikke generere arbeidsprosessverktøyet
  1. Systemet viser feilmelding og avslutter
- 5a Brukeren gjør endringer uten å lese tilbakemeldinger
  1. Gå til punkt 7

Tabell 2.7: Tekstlig beskrivelse av usecaset ”Endre eksisterende arbeidsprosess”



Figur 2.11: Interaksjon mellom bruker og system i usecaset ”Endre eksisterende arbeidsprosess”

Forbedre Verktøy

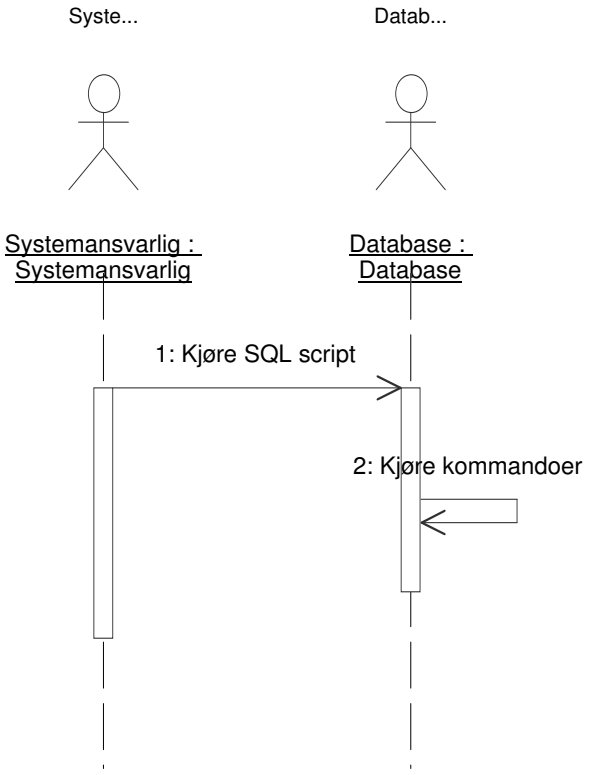
Hovedscenario

1. Brukeren har et klart SQL-script og sender dette til databasen
2. Systemet kjører kommandoene og havner i en ny tilstand

Utvidelser

- 2a Det oppstår en feil under kjøring av scriptet
    1. Systemet sender en feilmelding

Tabell 2.8: Tekstlig beskrivelse av usecaset ”Forbedre Verktøy”



Figur 2.12: Interaksjon mellom bruker og system i usecaset ”Forbedre verktøy”

Hente statistikk

Forutsetninger: Brukeren har logget inn

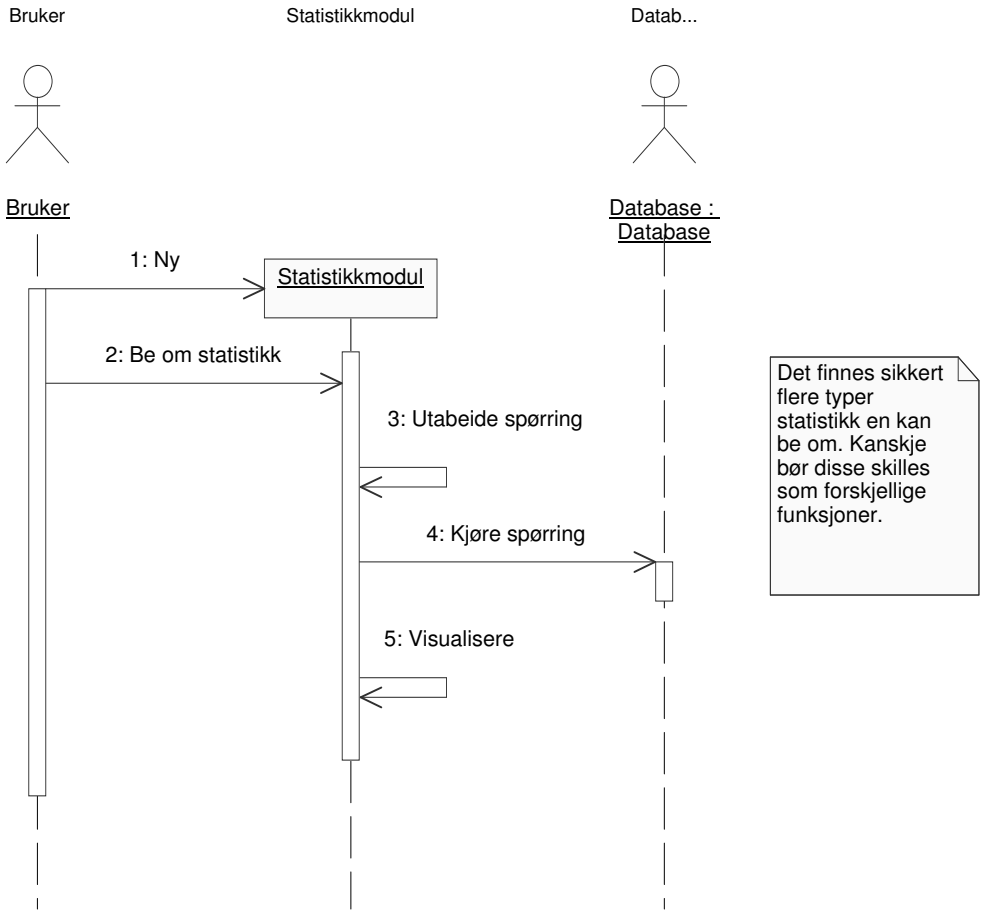
Hovedscenario

1. Brukeren ber om ny statistikkmodul
2. Systemet genererer statistikkverktøy
3. Brukeren ber om en gitt type statistikk
4. Systemet utarbeider en SQL spørring
5. Systemet kjører spørringen
6. Systemet visualiserer resultatet

Utvidelser

- 2a Systemet kan generere statistikkverktøy
    1. Systemet viser feilmelding og avslutter
- 5a Systemet kan ikke kjøre spørringen
    1. Systemet viser feilmelding
    2. Brukeren prøver på nytt eller avbryter

Tabell 2.9: Tekstlig beskrivelse av usecaset ”Hente statistikk”



Figur 2.13: Interaksjon mellom bruker og system i usecaset ”Hente statistikk”

Validere verktøy

Forutsetning: Brukeren er logget inn.

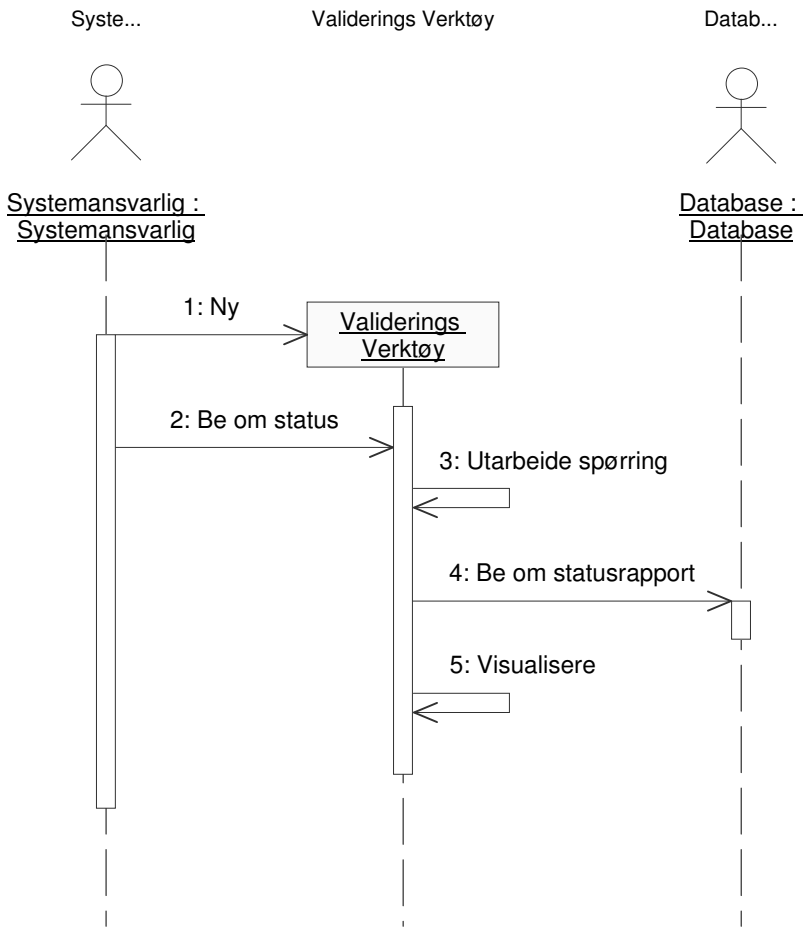
Hovedscenario

1. Brukeren ber om valideringsverktøy
2. Systemet genererer valideringsverktøy
3. Brukeren ber om en systemvalidering
4. Systemet utarbeider en SQL-spørring
5. Systemet kjører spørringen
6. Systemet visualiserer resultatet

Utvidelser

- 2a Systemet kan ikke generere valideringsverktøy
    1. Systemet gir feilmelding og avsluttes
- 5a Systemet kan ikke kjøre spørringen
    1. Systemet viser feilmelding
    2. Brukeren prøver på nytt eller avbryter

Tabell 2.10: Tekstlig beskrivelse av usecaset ”Validere Verktøy”



Figur 2.14: Interaksjon mellom bruker og system i usecaset ”Validere verktøy”

Kapittel 3

Spesifikasjoner

Det foregående kapittelet har gitt en generell innføring i arbeidsprosesser og systemer som kan brukes til å beskrive sådanne. I dette kapittelet blir begrepet tatt et steg videre i det det settes inn i en sammenheng: lagring av industrielle arbeidsprosesser.

Første del tar for seg en kravspesifikasjon med funksjonelle og ikke funksjonelle krav til systemet samt generelle systemkrav. Dette danner grunnlag for den neste delen der det blir beskrevet en detaljert ontologi for et dokumentformat som kan brukes til lagring av arbeidsprosesser.

Lagringsformatet i seg selv er ikke nok til å presentere den informasjonen vi nå kan lagre. Det trengs også et system for gjenfinning, distribuering, utvikling og videreutvikling av arbeidsprosesser. Referansearkitekturen skissert i siste del av kapittelet gir et rammeverk for et slikt system.

3.1 Spesifikasjon av et arbeidsprosessystem

I denne delen spesifiserer vi hensikten og funksjonaliteten ved et arbeidsprosesssystem og kartlegger det vi mener bør være krav som stilles til et slikt system. Det finnes flere typer krav, noen av dem går direkte på systemets funksjonalitet, andre går på mer generelle krav til system og metodikk. Vi har delt kravene inn i funksjonelle og ikke-funksjonelle som skissert i [Som01].

3.1.1 Beskrivelse og bruksområde

Systemet skal være en lagringsplass for bedriftens kunnskap i form av arbeidsprosessdokumenter. Arbeidsprosessene beskriver aktiviteter forbundet med daglig drift og vedlikehold av installasjoner og utstyr. De skal være tilgjengelige for medarbeiderene, slik at disse kan utføre det pålagte arbeidet på tilfredsstillende måte.

3.1.2 Funksjonalitetskrav

- Brukeren skal kunne hente arbeidsprosesser basert på referanser fra arbeidsordre.
- Brukeren skal kunne benytte fritekstsøk og søk på nøkkelord for å hente aktuelle arbeidsprosesser.
- Brukeren skal kunne få en dokumentrepresentasjon av arbeidsprosessen, og kunne skrive denne ut.
- Brukeren skal kunne navigere grafisk i arbeidsprosessen og dens subprosesser, og få fremvist multimediainnhold.
- Brukeren skal kunne laste en arbeidsprosess inn i en applikasjon som hjelper ham med utførelsen av arbeidsprosessen.
- Brukeren skal kunne få arbeidsprosessen presentert i en multimediastøttet gjennomgang.
- Arbeidsprosessansvarlig skal kunne tilføye nye arbeidsprosesser.
- Arbeidsprosessansvarlig skal kunne revidere arbeidsprosessene og tilføye nye versjoner.

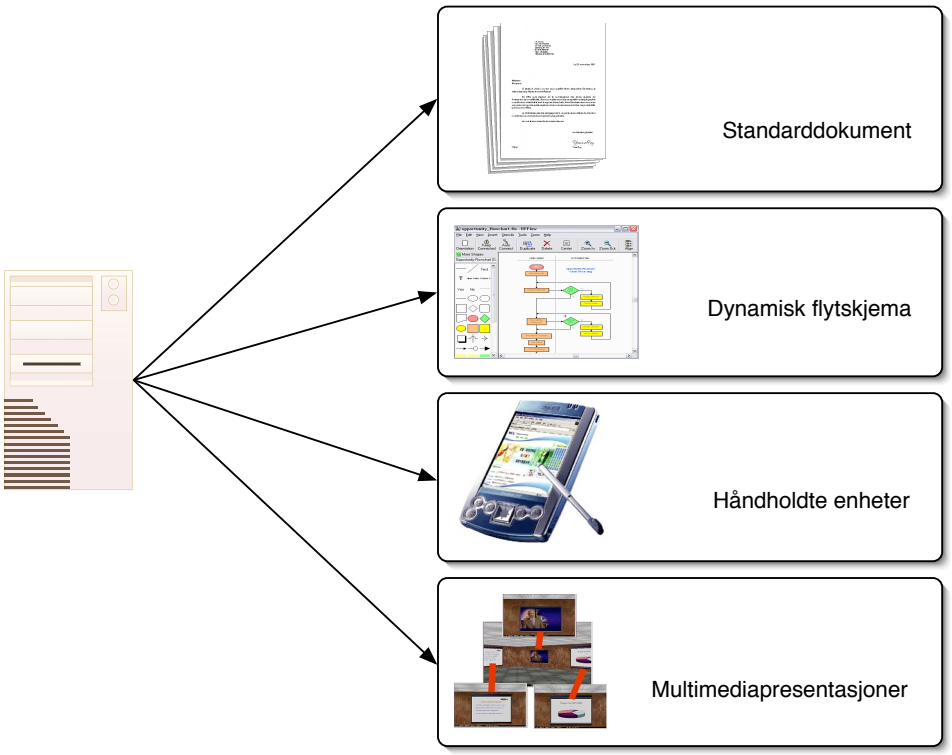
- Brukeren skal kunne gi tilbakemelding på informasjonen som kan tas hensyn til ved revidering.
- Systemansvarlig og arbeidsprosessansvarlige skal kunne hente ut bruksstatistikk.

3.1.3 Ikke-funksjonelle krav

- Arbeidsprosessene skal være under versjonskontroll, det vil si at dersom det lagres forandringer skal det opprettes en ny versjon av arbeidsprosessen, og den gamle skal bevares.
- Arbeidsprosessene skal være hierarkisk ordnet, slik at logisk sammenhørende elementer og elementer som er felles for flere arbeidsprosesser kan trekkes ut som egne arbeidsprosesser og inkluderes som subprosesser.
- Avhengigheter mellom arbeidsprosesser skal bevares og valideres ved endringer.
- Arbeidsprosessene skal kunne referere til multimediadata.
- Arbeidsprosessene skal kunne presenteres med referert multimediadata.
- Definisjonsspråket for arbeidsprosesser skal være fleksibelt nok til å beskrive arbeidsprosesser med så høy grad av kompleksitet som virkeligheten tilsier.
- Systemet skal kunne gjøre oppslag i andre systemer for å få tilgang til blant annet spesifikasjoner av roller og krav.
- Systemet skal sikres mot urettmessig tilgang og forandring.

3.1.4 Systemkrav

- Oppdatering av systemet skal kunne gjøres på en enkel måte. En skal kunne bygge grensesnittmoduler som forenkler endringen av en arbeidsprosess. Når endringen er gjort vil det med en gang være synlig for alle interessenter. Spesielt interesserte kan informeres om endringer gjennom et valgt medium, for eksempel e-post eller sms.
- Fremvisning av arbeidsprosesser kan foregå på flere ulike måter. For noen ideer til visningsformer se figur 3.1 på neste side.



Figur 3.1: Mulige presentasjonsformer basert på et generelt dokumentformat

– Et dokument skal kunne endres uten at dette forringer resten av systemet . Det skal ikke finnes brudne lenker i strukturen. Samtidig skal den til enhver tid beste praksis avspeiles i systemet.
– Det må finnes nok oppmerking til å spesifisere en arbeidsprosess fullt ut samtidig som formatet ikke blir rigid og vanskelig å benytte.

3.2 Spesifikasjon av arbeidsprosesser

I kapittel 2 ga vi en generell innføring i hva en arbeidsprosess er og hvilke egenskaper som kjennetegner den generelle arbeidsprosessen. I dette kapittelet har vi til nå analysert kravene som stilles til arbeidsprosessystemer. Vi ønsker nå å benytte dette for å kartlegge behov for lagring av informasjon. Vi beskriver informasjonen i en ontologi. Det viser seg at det vil trenges en del metainformasjon som kan brukes ved lagring og gjenfinning og kategorisering av arbeidsprosesser. For å behandle multimediavedlegg og andre eksterne ressurser trengs det også en definert vedleggstruktur.

Under utviklingen av arbeidsprosessdokumentet er det gjort en del nødvendige avveininger i forhold til hva som skal inn i lagringsformatet og hvordan denne er sammensatt. Disse blir beskrevet etter ontologien.

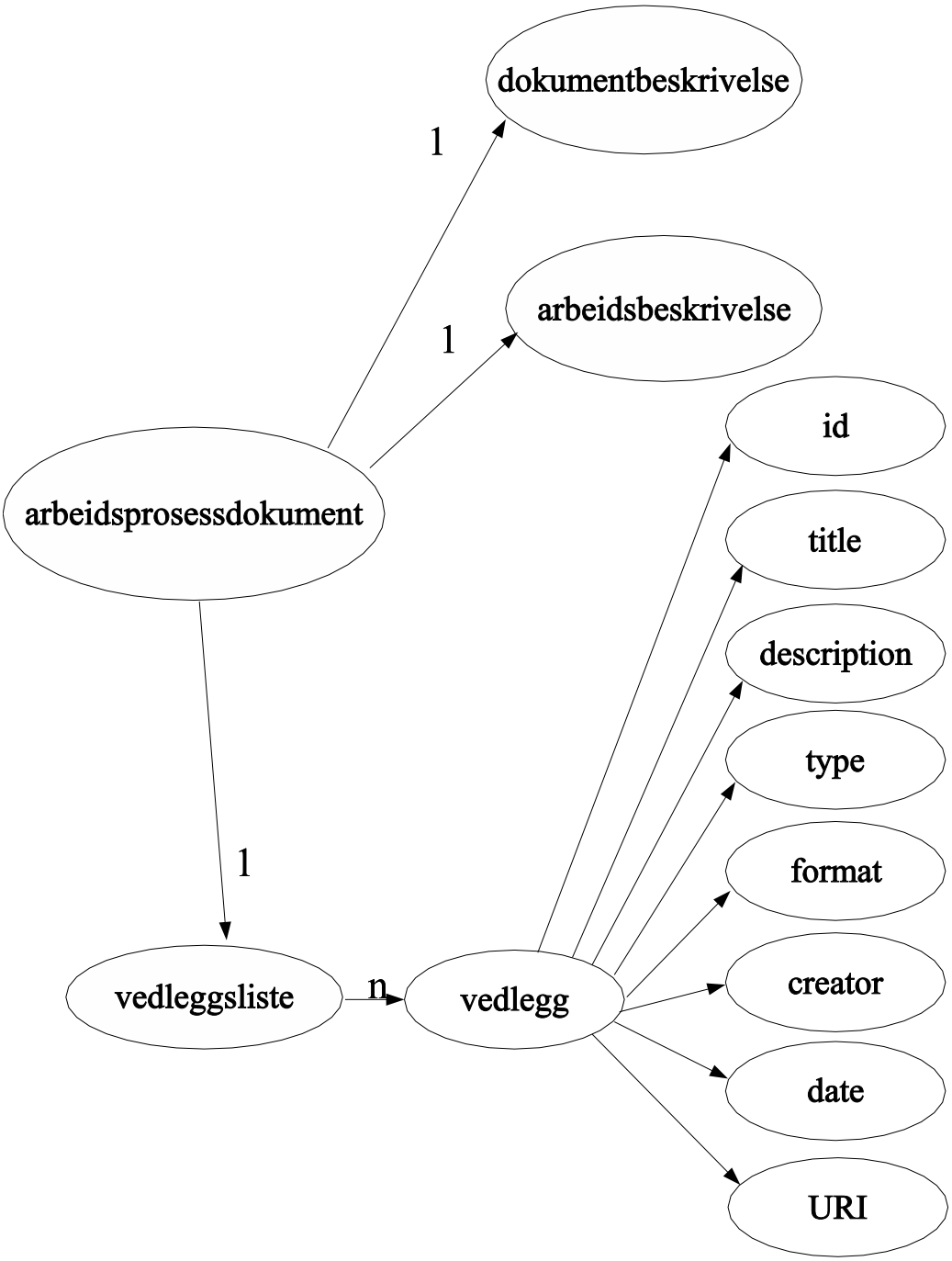
3.2.1 Ontologi

Elektronisk lagring av eksplisitt prosesskunnskap krever et lagringsformat. Dette avsnittet vil vise hvordan vi ønsker å lage strukturen i et dokumentformat tilpasset industrielle arbeidsprosesser. Inspirasjon til strukturen og innholdet har vi fått fra Workflow Management Coalition [WMC02], The Dublin Core Metadata Initiative [DCMI] sammen med interne dokumenter fra Sintef og Norsk Hydro.

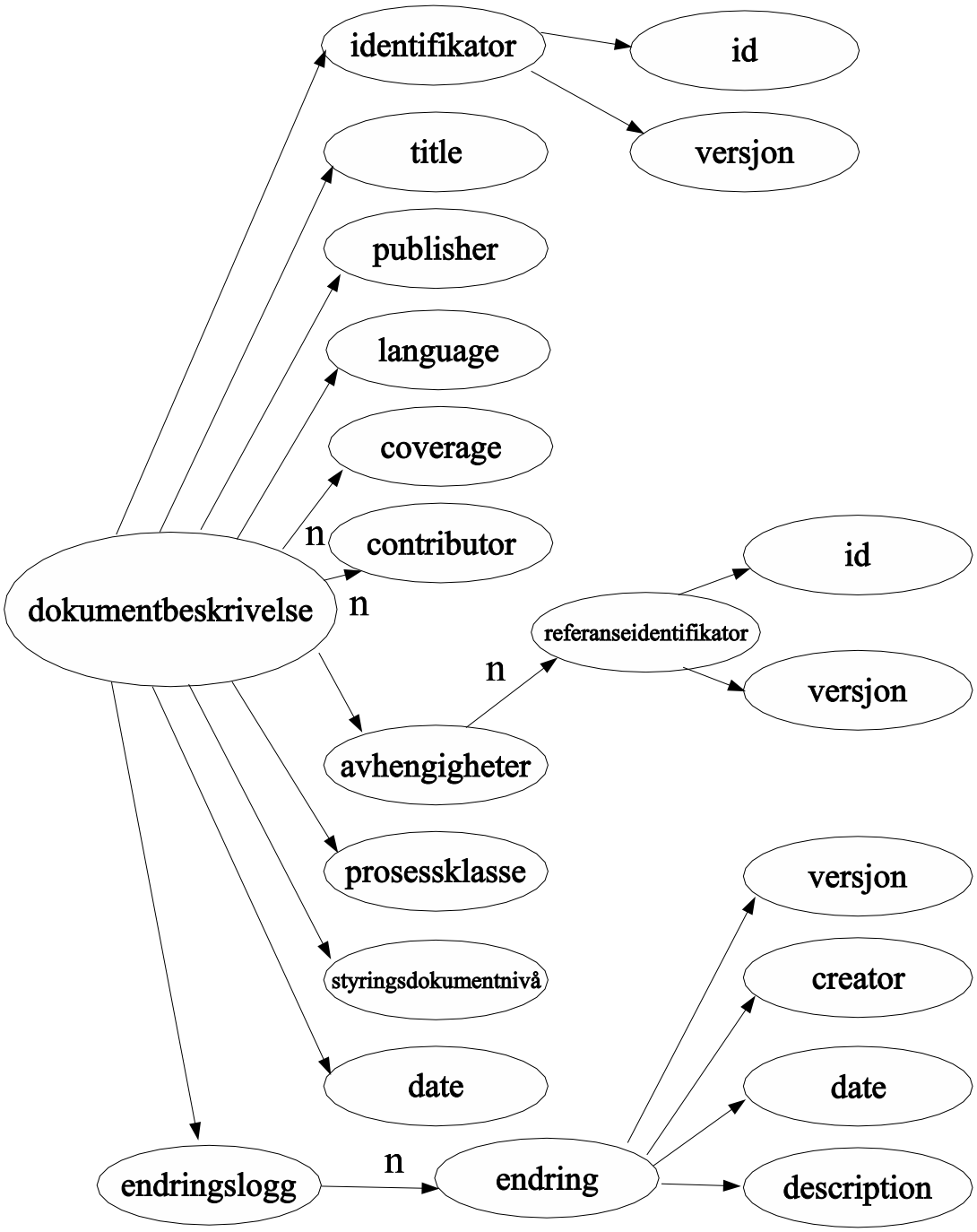
Informasjonen som kan lagres ved hjelp av dokumentformatet velger vi å lagre i ulike elementer. Hvordan disse henger sammen og hvilde data de inneholder har vi skissert i figurene 3.2, 3.3 og 3.4. En slik beksrivelse kan kalles en ontologi. Denne ontologien vil beskrives i de følgende avsnittene.

Overskriftene under vil være hovedelementer som under seg igjen inneholder flere elementer. Antallet elementer av hver type blir beskrevet i parenteser etter elementnavnet på følgende måte:

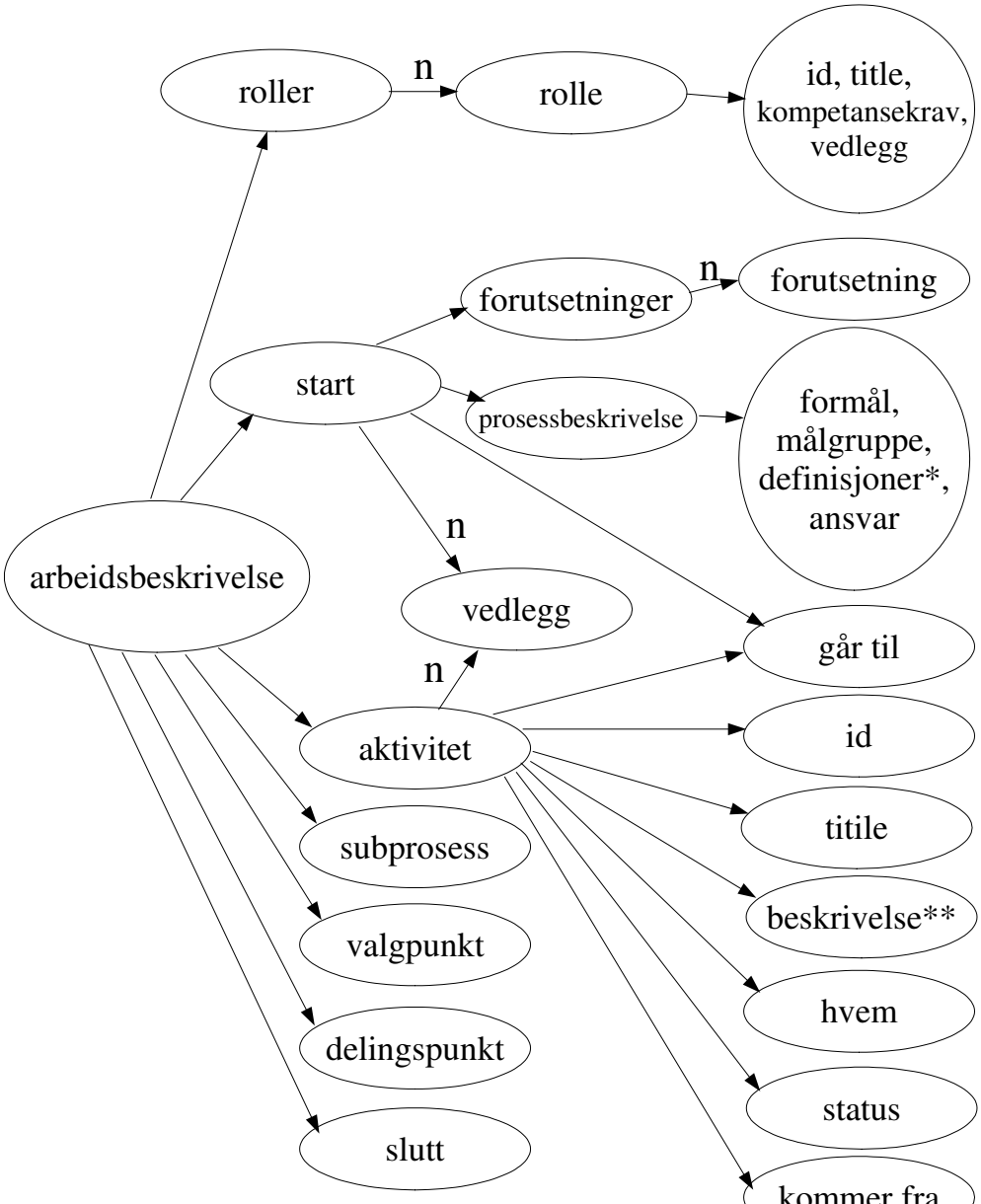
$$
\begin{aligned}
0 \ldots 1 &= \text{Null eller én} \\
0 \ldots n &= \text{Null til mange} \\
1 &= \text{Akkurat én} \\
1 \ldots n &= \text{En eller flere} \\
2 \ldots n &= \text{To eller flere}
\end{aligned}
$$



Figur 3.2: Grunnstrukturen i et arbeidsprosessdokument samt trestrukturen til vedleggsreferansene



Figur 3.3: Strukturen i et arbeidsprosessdokument. Utsnittet som viser er dokumentbeskrivelsen



* Definisjoner består av flere definisjon-elementer
** Beskrivelse består av mål, krav, hva, hvordan, hvorfor, årsak, risiko, konsekvens og hva om elementer.

Figur 3.4: Utsnitt som viser arbeidsbeskrivelsen i et arbeidsprosessdokument

Arbeidsprosessdokument

Elementet *arbeidsprosessdokument(1)* defineres som dokumentets rotelement. Under dette nivået finnes tre andre elementer som hver for seg inneholder elementer. En dokumentbeskrivelse brukes til å lagre generelle beskrivelser av det enkelte dokumentet. Arbeidsbeskrivelsen tar steg for steg for seg gangen i en arbeidsprosess. Vedleggslisten holder oversikt over vedleggene. Alle disse elementene blir beskrevet i detalj i de kommende avsnittene.

Dokumentbeskrivelse

En *dokumentbeskrivelse(1)* inneholder all metainformasjonen for arbeidsprosessen. Dette er informasjon som ikke er forbundet med utføringen av arbeidsprosessen. En trestruktur som viser hvordan dokumentbeskrivelsen er bygget opp vises i figur 3.3 på side 47. De ulike elementene blir forklart under.

- *Identifikator(1)* er det elementet som gir det spesifikke dokumentet en unik identifikasjon, primærnøkkelen. Den består av et *identifikasjonsnummer(1)* som bestemmer arbeidsprosessen som beskrives, og et *versjonsnummer(1)* for denne.
- *Tittel(1)* på arbeidsprosessen. Den er knyttet opp mot identifikasjonsnummeret i identifikatorbolken slik at samme identifikasjonsnummer alltid har samme tittel.
- *Prosessansvarlig(1)* beskriver en person/stilling/rolle som har ansvaret for å behandle forespørsler og forslag om forandringer i arbeidsprosessen. Han har også ansvaret for å revidere arbeidsprosessen dersom andre arbeidsprosesser som denne er avhengig av forandres.
- *Språk(1)* som brukes gjennom dokumentet. Dcmi anbefaler referanser i henholdt til standarden ISO639 [JWG] som beskrevet i RFC3066 [Alv01].
- *Prosessklasse(1)* er en horisontal klassifisering av dokumentet.
- *Styringsdokumentnivå(1)* definerer et nivåsystem og klassifiserer arbeidsprosessene i detaljnivåer.
- *Dato(1)* viser når dokumentet sist ble forandret. Dcmi anbefaler standarden ISO8601[WW97].
- *Avhengigheter(1)* En liste over andre arbeidsprosesser i representert med en *avhengighet(0 . . . n)*. Hver av disse består av en intern *id(1)* og en *refer-*

anseidentifikator(1) som den gitte arbeidsprosessen er avhengig av. Dette vil si at de kan inkluderes i arbeidsbeskrivelsen som underprosesser. Gjennom referanseidentifikatoren kobles avhengigheten opp mot spesifikke versjoner av de refererte arbeidsprosessene. Dette sikrer at oppdateringer ikke gjør arbeidsprosessene inkompatible. Mer om avhengighetssystemet finnes i 3.2.2 på side 56.

- *Endringslogg(1)* benyttes hver gang dokumentet oppdateres. Da legges det til et nytt *endring (0 . . . n)*. Denne beskriver den nye *versjon(1)* og *dato(1)* for endring sammen med en *beskrivelse(1)* av endringene som er blitt gjort. Personen som gjør endringen skriver sitt navn ved endringen.
- *Bidragsyter(0 . . . n)* er en ressursperson som har bidratt med informasjon om arbeidsprosessen.
- *Virkeområde (0 . . . n)* definerer sfæren der dokumentet gjelder. Kan for eksempel være forskjellige installasjoner, anlegg eller skift.

Arbeidsbeskrivelse

En *arbeidsbeskrivelse(1)* skal fungere som en modell for hvordan en arbeidsprosess skal beskrives. Som kjent er det viktig tydelig å avklare krav, ansvar og arbeidsgangen i prosessen. I denne seksjonen skapes det derfor et rammeverk for å definere roller og de ulike aktivitetstypene slik at en enkelt kan lagre de data som trengs for senere visning. En trestruktur som beskriver dette elementet finnes i figur 3.4 på side 48.

Roller *Roller(1)* i en arbeidsprosess kan sammenlignes med en ressurs som trengs for å få arbeidet utført. Det kan trenges flere ressurser og hver av disse beskrives med en *rolle(1 . . . n)* For å beskrive disse velger vi å bruke:

- *Id(1)* er et identifikasjonsnummer som holder de ulike rollene fra hverandre.
- *Tittel(1)* er en tekstlig beskrivelse av rollen, for eksempel linjeleder eller mekaniker
- *Kompetansekrav(1)* som forteller hvilke kvalifikasjoner arbeideren som skal fylle rollen må inneha. Kompetansekrav er et flertallsord og vi ser for oss at det under kompetansekrav kan ligge flere ulike *krav(1 . . . n)*. Det kan her være interessant og nødvendig å knytte egenskapene opp mot bedriftens nåværende kompetansedatabase og få dataene til å være i henhold til denne.

– *Vedlegg(0 . . . n)* slik at eksterne ressurser kan benyttes til å forklare rollen.

Start Arbeidsprosessen har et definert punkt, *start(1)*, som markerer begynnelsen. Start nås når visse forutsetninger er oppfylt. Ved oppstart av en prosess er det viktig at disse forutsetningene er klart definert samt at andre variable er satt. Disse beskriver vi med:

– *Forutsetninger(1)* for at denne arbeidsprosessen iverksettes. Dette elementet består av et sett *forutsetning(1 . . . n)* elementer. Forutsetningene vil være global og synlig for alle aktiviteter relatert til denne. Forutsetningene vil inneholde:

– *Prosessbeskrivelse(1)* beskriver en del generell informasjon om verdikjeden. Dette er

  • *Formål(1 . . . n)*, den overordnede hensikten med verdikjeden, hva man ønsker å oppnå, og hvorfor.

  • *Målgruppe(1 . . . n)* forteller hvilket personell som må forholde seg til og forstå dokumentet.

  • *Definisjoner (1)* beskriver ord og uttrykk brukt i arbeidsprosessen. Det kan finnes flere definisjoner og hver av disse lagres i *definisjon(0 . . . n)* elementer.

  • *Ansvar(1)* forklarer hvem som har hovedansvar for oppfølging av instruksene i arbeidsprosessen.

– *Går til(1)* er en referanse til første aktivitet, valgpunkt eller subprosess.

– *Vedlegg(0 . . . n)* er en referanse til et vedlegg i vedleggslisten slik at eksterne ressurser kan benyttes til forklaring av arbeidsprosessen

Aktivitet En *aktivitet(0 . . . n)* skal beskrive en serie sammenhengende arbeidsoppgaver på en slik måte at de kan utføres i henhold til bedriftens beste praksis.

– *Id(1)* er et identifikasjonsnummer som holder de ulike aktivitetene fra hverandre

– *Tittel(1)* er en kort tekstlig beskrivelse av aktiviteten

– *Beskrivelse(1)* av aktiviteten er delt inn i:

  • *Mål(0 . . . n)* for aktiviteten, det vil si hva en ønsker å få ut etter at aktiviteten er utført

  - *Krav(0 . . . n)* forbundet med aktiviteten. Dette kan i likhet med bruken av eksterne rollebeskrivelser være referanser til krav definert i andre systemer
  - *Hva(0 . . . n)* som skal gjøres forklart i noe mer detalj en det som er gitt i navnet
  - *Hvordan(0 . . . n)* aktiviteten skal utføres. En steg for steg forklaring av utførelsen
  - *Hvorfor(0 . . . n)* aktiviteten skal utføres
  - *Årsak(0 . . . n)*, det vil si vanlige årsaker til at ulykker oppstår
  - *Risiko(0 . . . n)* for at skade oppstår ved ulykker
  - *Konsekvens(0 . . . n)* for involverte ressurser om uhellet er ute
  - *Hva om(0 . . . n)* en utfører aktivitetene på feil måte? Hva skal en gjøre for å minimere skadeomfanget

- *Hvem(1)* beskriver hvem som utfører denne aktiviteten. Det vil i praksis være en referanse til et sett med roller
- *Status(1)* skal beskrive hvilken viktighet aktiviteten har (kritisk, normal, moderat etc.)
- *Kommer fra(1 . . . n)* er en referanse til foregående aktivitet. Dette kan være et sett med flere aktiviteter
- *Går til(1)* er en referanse til neste aktivitet i arbeidsprosessen
- *Vedlegg(0 . . . n)* er en referanse til et vedlegg slik at eksterne ressurser kan benyttes til forklaring av arbeidsprosessen

Subprosess En arbeidsbeskrivelse kan i tillegg til de enkelte aktivitetene bestå av en større prosess som beskriver en aktivitet i større detaljnivå. Dette muliggjør gjenbruk av basisprosesser og øker dynamikken i dokumentet. En *subprosess(0 . . . n)* blir beskrevet på følgende måte

- *Id(1)* er aktivitetens referansenummer innad i dette dokumentet
- *Identifikator(1)* brukes for å referere til en avhengighet beskrevet i dokumentbeskrivelsen
- *Kommer fra(1 . . . n)* er en referanse til foregående aktivitet. Dette kan være et sett med flere aktiviteter

- *Går til(1)* er en referanse til neste aktivitet i arbeidsprosessen
- *Vedlegg(0 . . . n)* er en referanse til et vedlegg slik at eksterne ressurser kan benyttes til forklaring av subprosessen

Valgpunkt Et *valgpunkt(0 . . . n)* legges inn i en arbeidsbeskrivelse dersom en på et punkt kan velge mellom flere ulike veier. En valgboks tar en verdi inn og velger en av to mulige utganger alt etter hvilke spørsmål som stilles og verdi som kommer inn. Valgpunktet beskrives under.

- *Id(1)* er valgpunktets referansenummer i dokumentet
- *Spørsmål(1)* av typen ja/nei
- *Beskrivelse(1)* er en forklaring til spørsmålet
- *Ja-går-til(1)* er referansen til neste aktivitet ved positivt svar
- *Nei-går-til(1)* er referansen til neste aktivitet ved negativt svar
- *Vedlegg(0 . . . n)* er en referanse til et vedlegg slik at eksterne ressurser kan benyttes til forklaring av valgpunktet

Delingspunkt Et *delingspunkt(0 . . . n)* legges inn i en arbeidsbeskrivelse der arbeidsflyten deler seg i ulike parallelle aktiviteter eller subprosesser. Delingspunktet beskrives som følger.

- *Id(1)* er delingspunktets referansenummer i dokumentet
- *Beskrivelse(1)* til hvorfor delingen skjer her.
- *Går til(2 . . . n)* er referansen til de aktivitetene som skal foregå i paralell
- *Vedlegg(0 . . . n)* er en referanse til et vedlegg slik at eksterne ressurser kan benyttes til forklaring av delingspunktet

Slutt Når er arbeidsprosess er fullført skal det være mulig beskrive hvilken tilstand systemet er i og hvilke forutsetninger en jobber videre fra.

- *Id(1)*
- *Kommer fra(1 . . . n)* er en referanse til en aktivitet som er det siste fullført før prosessen er sluttført
- *Post-condition(1. . . n)* forklarer hva som er resultatet av prosessen

– *Vedlegg(0 . . . n)* er en referanse til et vedlegg slik at eksterne ressurser kan benyttes til forklaring av sluttpunktet

Det vil være ett eller flere sluttpunkt i en arbeidsbeskrivelse. Disse beskrives med en eller flere ”kommer fra” referanser og et sett postcondition-uttrykk.

Vedleggsliste

Et arbeidsprosessdokument må kunne støttes av vedlegg. De ligger referert i en *vedleggsliste(1)* og kan være av ulike typer. Det kan for eksempel brukes film, animasjon, ekstene websider, xml-dokumenter, lydklipp. Disse samles her slik at de enkelt kan refereres fra andre plasser i dokumentet samtidig som de beskrives på en plass og på en strukturert måte. En figur som viser trestrukturen til en vedleggsliste finnes i figur 3.2 på side 46

Vedlegg Hvert vedlegg blir beskrevet med :

– *Id(1)* er vedleggets referansenummer i dokumentet

– *Tittel(1)* er et beskrivende navn på vedlegget

– *Beskrivelse(1)* er en mer utfyllende beskrivelse av vedlegget

– *Type(1)* beskriver hva slags vedlegg dette er, for eksempel tekst, film, lyd eller lignende

– *Format(1)* beskriver hvilket format vedlegget er lagret på. For eksempel mpeg2, mpeg4, pdf, vorbis, svg, jpeg, swf, osv.

– *Laget av(1)* forteller hvem som har opphavsretten til vedlegget

– *Dato(1)* forteller hvilken dag vedlegget ble opprettet

– *URI(1)* er linken til vedlegget

3.2.2 Begrensninger og avveininger

I løpet av arbeidsperioden var det nødvendig å gjøre en del avveininger med hensyn på arkitektur, metoder og språk. Avveiningene gjort med hensyn på systemarkitektur er beskrevet i referansearkitekturen i kapittel 3.3 på side 58. Andre avveininger som er gjort blir beskrevet i de følgende avsnittene.

Datastruktur

I et lagringsformat må det finnes en struktur. Hvordan denne skal være i hvert enkelt tilfelle er åpent for en bred diskusjon og noe definitivt svar på hva som er best finnes vel ikke.

Arbeidsprosessdokumentet er organisert i en trestruktur. Det treet som tegnes er noe ubalansert og data ligger på ulike nivå i treet. Dette ser vi ikke på som noe problem da strukturen er fast definert. Søking i treet vil gjøres på grunnlag av en ”document object model” (dom) eller lignende. Det medfører at en hele tiden vet hvor i dokumentet en finner gitt informasjon og søketiden vil være:

$$\text{Søketid i kjent trestruktur} \quad = \quad O(n)$$

Logisk struktur

Vi har valgt å la trestrukturen i vårt dokument gjenspeile en logisk oppdeling av dokumentet, men samtidig skal gi en relativt kort aksessvei til dataen. Figur 3.2 på side 46 viser grunnelementet og de tre elementene under med detaljene rundt vedleggsdelen. Figur 3.3 på side 47 viser detaljene til dokumentbeskrivelsen. Figur 3.4 på side 48 viser arbeidsbeskrivelsen. Rolle-, start- og aktivitetselementet er fullt ut beskrevet. Subprosess-, valgpunkt- og sluttelementene er ikke like godt beskrevet, men har flere likheter med aktivitet i det at de alle har id, vedlegg, kommer-fra- og går-til-elementer.

Den inndelingen vi har gjort baserer seg på at en del informasjon er metainformasjon for arbeidsprosessen og kan skilles fra selve arbeidsbeskrivelsen. Derfor har vi skilt elementene i dokumentbeskrivelsen og arbeidsbeskrivelsen fra hverandre.

Når vi på samme nivå har skilt ut en vedleggsliste kommer det av at vedlegg vanligvis er eksterne lenker og disse bør skilles fra resten av dokumentet. Konseptuelt kan en si at roller (som ligger under arbeidsbeskrivelsen) og vedlegg har mange likheter og derfor bør ligge på samme nivå. Når vi likevel har skilt dem kommer det av mer praktiske årsaker slik som muligheten for faktisk å legge inn hele bilder og lignende i vedleggfeltet. Å ha dette skilt ut kan derfor øke lesbarheten noe. Effektiviteten endres ikke på grunn av dette.

Interne begrensninger

I ontologien finnes enkelte felter som viser til eksterne ressurser og noen som viser til interne ressurser. Eksempel på det førstnevnte er vedlegg og avhengigheter. Som eksempel på interne referanser kan vi bruke roller. En del av dokumentet inneholder den formelle spesifikasjonen av roller. Dermed er rollene skilt ut fra arbeidsflyten. Roller refereres i forbindelse med startpunkter og aktiviteter, henholdsvis som hovedansvarlig for prosessen og hvem som utfører aktiviteten. Når definisjon og bruk er adskilt kan samme rolle benyttes i flere aktiviteter uten redundans.

I lagringsformatet foreslår vi altså at roller, vedlegg og avhengigheter gis unike identifikatorer slik at de kan refereres fra andre steder i dokumentet.

Eksterne avhengigheter

Et arbeidsprosessdokument står ikke alene, men kan selv både være inkludert i og inneholde flere andre arbeidsprosesser. Dette er en utfordring som påpekes av flere, for eksempel [Hol04, AG04]. Særlig trengs en strategi for å beholde konsistens gjennom kontinuerlige forandringer i arbeidsprosessene. Det er derimot vanskelig å finne en løsning skissert for behandling av dette.

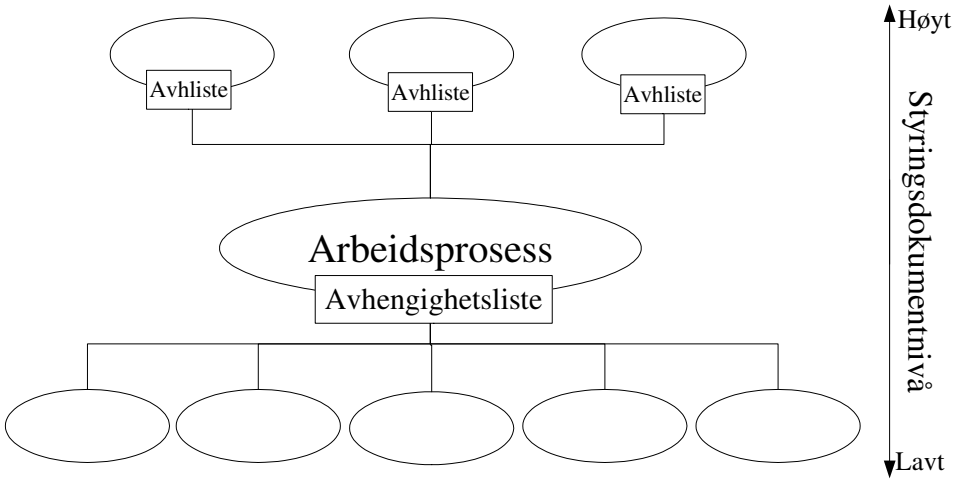
Vi definerer i dokumentet en avhengighetsliste. Denne vil inneholde referanser til andre arbeidsprosessdokumenter. På samme måte vil andre dokumenter inneholde referanser til denne som vist i figur 3.5 på neste side[1]. Subprosesser som inngår i flere verdikjeder kaller vi flerbruksprosesser.

En viktig egenskap ved arbeidsprosesser er at de vil endre seg over tid. Når en flerbruksprosess blir oppdatert kan dette påvirke gangen i alle de overordnede prosessene, i verste fall endre disse til det værre. Det er derfor viktig at ansvarlige for de overordnede prosessene får beskjed om at endringer har skjedd, og kan ta konsekvensene av dette. En løsning på dette skisseres i kapittel 4.2.3 på side 79.

Hensyn til systemkrav

Kravet om stabilitet i systemet må løses på flere nivåer. På dataformatnivået må det bygges inn konstrukter som gjør det mulig å unikt identifisere og referere et

[1] Vi forutsetter at ikke sirkelavhengigheter oppstår. Dette vil være høyst usannsynlig, særlig med organiseringen i styringsdokumentnivåer



Figur 3.5: Forholdet mellom arbeidsprosesser på ulike nivåer

arbeidsprosessdokument. Dette har vi gjort ved å innføre identifikatorer og referanseidentifikatorer. For at disse skal være globalt unike, noe de må være, må også laget over benyttes. Til dette anbefaler vi å bruke en database. Ved å gjøre identifikatoren til primærnøkkel oppnår en at disse er globalt unike.

Det er viktig at formatet er fleksibelt slik at brukeren kan formidle den informasjonen han ønsker. Fleksibilitet oppnår vi ved å ha mange felter hvorav kun de viktigste må være med for å forme et valid dokument.

Et annet konstrukt vi velger for å øke fleksibiliteten er direkte linkede vedlegg. I forbindelse med hvert enkelt steg i en arbeidsprosess kan en tilordne ulike vedlegg samt en beskrivelse av hvordan vedlegget skal benyttes i en gitt sammenheng.

3.3 Referansearkitektur

Dette kapittelet presenterer en systemarkitektur for en arbeidsprosess-server med multimediainnhold. Vi viser en modell og beskrivelser av de enkelte komponentene. Målet for arkitekturen har vært at den skal la seg implementere med en høy andel ferdige komponenter, og være skalerbar opp til mange samtidige brukere. Til slutt følger en strategi for utvidelse av modellen med det mål å øke systemets kapasitet, eller skalering, og en beskrivelse av metoder for å lagre arbeidsprosessdokumentene i databasekomponenten.

3.3.1 Modell

Vi beskriver i figur 3.6 på neste side en arkitektur for en server for dynamiske arbeidsprosesser med multimediainnhold. Den er basert hovedsaklig på standardkomponenter, noe som gjør implementasjon langt raskere og enklere.

Applikasjonsserver

Applikasjonsserveren er basis for logikk og integrasjon av tjenestene i systemet. Den består av flere funksjonsmoduler:

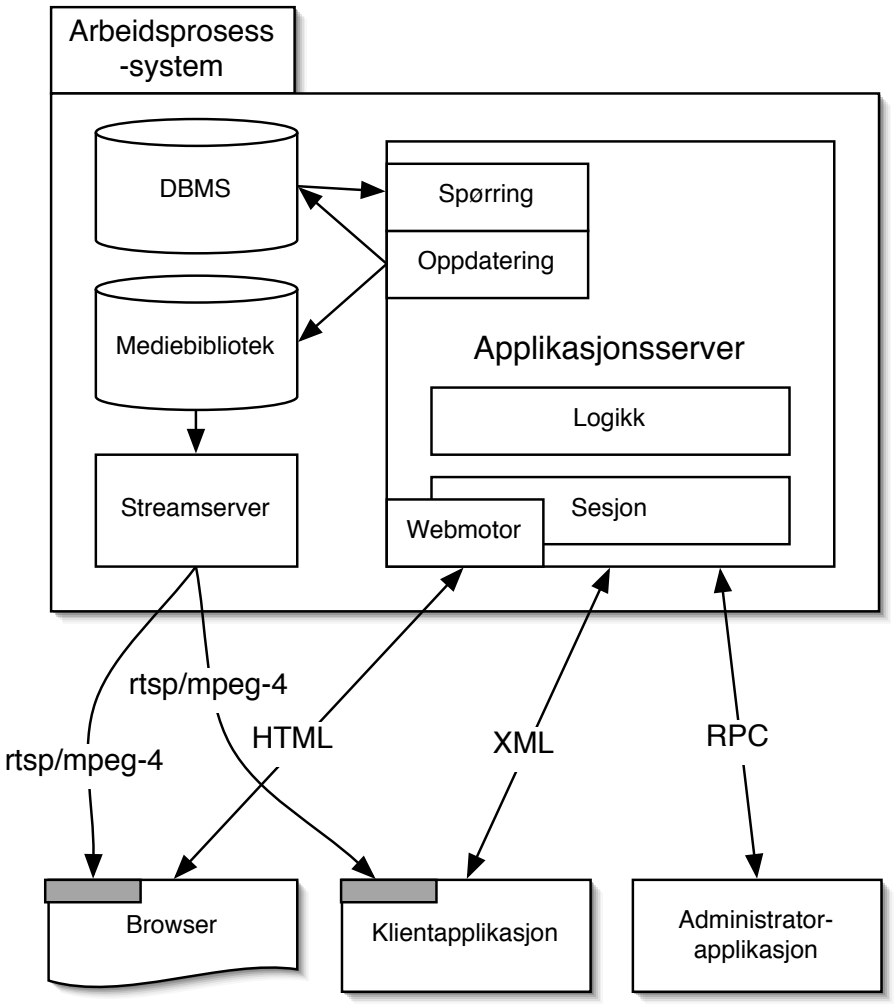
Innloggings- og sesjonsbehandleren tar imot innloggingsforespørsler og holder orden på innloggede brukere og deres status. Sesjonskontroll er viktig for å kunne tilby en god arbeidsflyt for brukere.

Databasespørringsbehandleren foretar søk og spørringer i databasen

Databaseoppdateringsbehandleren foretar alle oppdateringer i databasen, og synkroniserer disse med mediebiblioteket i streameserveren.

Webapplikasjonsmotoren genererer skjermbilder i form av html-dokumenter som er intermanøvrerbare og beskriver forskjellige deler av en arbeidsprosess i på forskjellige nivåer.

Logikkmodulene foretar behandling av data og handlinger basert på dataen.



Figur 3.6: Foreslått referansemodell for en ferdigkomponentbasert multimedia arbeidsprosess-server

Database

Databasen er en standardkomponent som gir oss mye ferdig funksjonalitet i form av stabil og sikker persistens, transaksjonshåndtering og mulighet for avanserte spørringer. De fleste datasystemer behøver en funksjonalitet for å sikkert kunne lagre informasjon. Siden dette behovet er så vanlig finnes det generelle databasesystemer man kan integrere med systemet sitt og konfigurere slik at de passer inn.

Sql-databaser kan brukes til å lagre arbeidsprosessene som strukturerte dokumenter. Disse databasene er svært modne og kjente applikasjoner, og ved å holde seg innenfor standardisert sql kan man velge blant veldig mange av disse også etter at applikasjonen er skrevet. Forskjellige databaseleverandører har forskjellige utvidelser og spesialkomponenter som kan gjøre det enklere å lage løsninger som dette. I vår modell ønsker vi likevel å holde databasen som en enkel komponent for å holde arkiktekturen så ren som mulig. I virkeligheten kunne en del av komponentene vi beskriver være en del av databasen.

Klientapplikasjon

Klientapplikasjonen er installert på brukerens datamaskin, og henter søkeresultater og arbeidsprosessdokumenter fra applikasjonsserveren. Disse vises og kan manøvreres i på en effektiv og responsiv måte. Alternativt til en standard applikasjon kan man basere seg på nedlastbare programmoduler av typen java/flash/shockwave. Disse vil ha mye av de samme fordelene, men vil kunne være plattformuavhengige og ikke behøve oppdateringer hos brukeren etterhvert som de utvikler seg.

Webklient

En nettleser, som er en vanlig applikasjon som finnes på de fleste datamaskiner, brukes mot webgrensesnittet, eller webmotoren, som er implementert i applikasjonsserveren. Til forskjell fra situasjonen der man bruker en klientapplikasjon blir skjermbildene og manøvreringen innad i arbeidsprosessen tatt hånd om på applikasjonsserveren. Hensikten med med dette er at systemet kan aksesseres fra en hvilken som helst datamaskin uten videre installasjon av programvare. Avspilling av media skjer ved hjelp av en tilleggsmodul i nettleseren eller ved at nettleseren kaller operativsystemets foretrukne medieavspiller.

Administratorapplikasjon

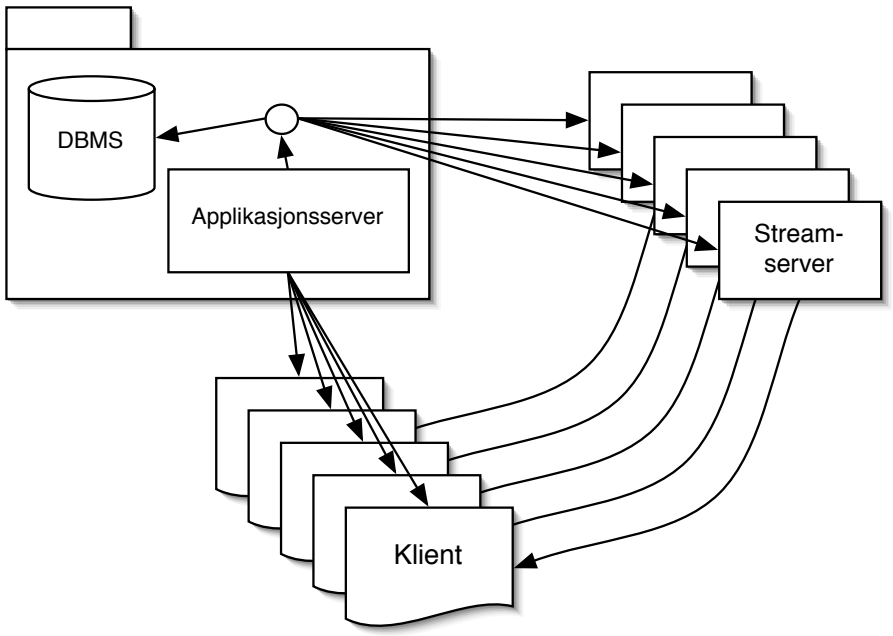
Administrasjonsapplikasjonen inneholder funksjonalitet for å gjøre oppdateringer i databasen og mediebiblioteket samt hente statistikk og gjøre forskjellige endringer i systemet. Det vil kanskje være vanlig å inkludere denne typen funksjonalitet i webapplikasjonen, men dersom enkelte mennesker skal jobbe mye mot dette grensesnittet vil det lønne seg å lage en applikasjon som gjør arbeidet lettere. Grafiske brukergrensesnitt som kjøres hos brukeren har mye bedre responsivitet og fleksibilitet enn dagens nettleserapplikasjoner. For brukere som skal oppdatere systemet er det heller ingen hindring at en slik applikasjon må installeres på maskinen, ettersom dette vil være arbeid man utfører på egen arbeidsplass.

Streamserver

Streamserveren gir tilgang til direkte avspilling av innholdet. Dataen aksesseres over en egen streaming-protokoll av en komponent hos klienten som dekoder den og spiller den av. Streaming begrenser venting og unødvendig bruk av båndbredde og lagringsplass ved å la brukeren spille av det aktuelle innholdet fra serveren og hoppe vilkårlig frem og tilbake fremfor å overføre hele filen.

Mediabiblioteket

Mediabiblioteket er samlingen av presentasjoner, filmer, lydklipp, bilder, modeller og animasjoner som systemet forvalter. Å lagre alle mediafilene i databasen hadde latt seg gjøre. Mediafiler kan lagres som Binary Large Object, eller Blob, sammen med metainformasjon. Dette vil føre til en garantert god konsistens mellom mediabiblioteket og den tekstlige informasjonen i databasen, ettersom mediabiblioteket faktisk hadde vært en del av databasen. Det er likevel en del grunner til at vi velger å ikke gjøre det på denne måten. Neste avsnitt forklarer en av grunnene, nemlig skalering av streamekapasitet. Dette frigjør også databasen fra arbeid, slik at responstidene blir bedre, og systemet kan belastes mye mer før det blir nødvendig å replisere selve databasen. At filene ligger utenfor databasen fører til at det blir enklere for streamserveren å nå dataen, særlig hvis den går på en annen maskin. Det finnes databasesystemer som støtter mye av den påkrevde funksjonaliteten i forhold til et mediabibliotek [Mau03]. Dette innebærer blant annet mulighet for å lagre filer i databasen og la streamservere streame innholdet direkte fra den eller å la databasen automatisk lagre mediafilene i et eksternt filsystem og selv beholde peker og metainformasjon. Man kan så utvide slik at streamserverene selv lagr-



Figur 3.7: Skalering av streamkapasiteten

er filen fra databasen, slik at den ikke må hentes hver gang, men kun behøver å versjonskontrolleres.

3.3.2 Skalering

Når belastningen på systemet øker vil det være en av komponentene som til slutt ikke klarer å betjene forespørslene. Hvilken komponent dette er vil være avhengig av bruksmønsteret, men sett ut fra datamengden er det rimelig å anta at stream-serveren vil bli flaskehalsen. En webside eller et arbeidsprosessdokument i xml-format vil ha en størrelse på noen kbyte, grafikken i en webside noen hundre kbyte, mens noen minutter film av god kvalitet fort kan utgjøre 100Mbyte. Figur 3.7 viser en enkel måte å utvide systemets streamekapasitet.

Applikasjonsserveren kan holde en oversikt over hvilke streamservere som er tilgjengelige, og fordele forespørslene etter belastning og geografisk informasjon, slik at klientene kobler seg til serverene som har ledig kapasitet eller den som er nærmest. En annen strategi er å spre mediebiblioteket utover de forskjellige streamserverne og flytte data mellom dem for å balansere belastningen. Da vil også to klienter som ønsker den samme dataen kunne betjenes lettere enn om

de skulle bruke hver sin server. Det vil naturligvis være transparent for brukeren hvilken server han kobler seg til.

Det er en utfordring å holde et replisert mediebibliotek oppdatert. Å flytte store datamengder over ustabile nettverk med begrenset båndbredde fra en kilde til mange mottakere er en vanskelig oppgave som flere har gitt seg i kast med. En av de mest populære og utbredte løsningene heter BitTorrent [Coh03]. BitTorrent er laget for å unngå at en servers båndbredde blir en flaskehals ved at klienter som alle ønsker å laste ned en større fil sender deler av filen som de allerede har fått lastet ned til hverandre. Initiativet tas fra klienten, så dette er ikke i seg selv en push-teknologi, men et godt fundament for å utvikle noe slikt. Andre systemer eksisterer der klientnoder kan abonnere på en kanal og hjelpe med å videredistribuere dataen.

3.3.3 XML-dokumenter i databasen

Xml, eXtensible Markup Language er et system for å definere dokumentformater og markere opp dokumenter etter de predefinerte formatene. Hvis man skal bruke dette for å beskriv earbeidsprosesser lager man en dokumentdefinisjon for arbeidsprosesser, og skriver arbeidsprosessene etter denne definisjonen. Xml-dokumenter er lesbare for mennesker, og enkle å forstå.

Lagring

Vi ønsker å lagre xml-dokumentene som beskriver arbeidsprosessene i en database. Dette kan gjøres på flere forskjellige måter, alt etter hvor stor jobb man vil gjøre, hva slags fleksibilitet og ytelse man ønsker, og hvilket databaseprodukt man ønsker å binde seg til eller frigjøre seg fra.

Å lagre hele xml-dokumentet i en Blob, eller Binary Large OBject, sammen med en nøkkel og nøkkelord er en enkel fremgangsmåte som kan brukes med de fleste databaser. Denne gir mulighet til å søke på nøkkelen og på nøkkelord, men ikke tilgang til effektive søk i selve teksten. Arbeidsprosessen vil returneres fra databasen akkurat slik den var da den ble lagt inn.

Å lagre dokumentet i en Blob med utvidede søkeord kan være en måte å bedre søkemulighetene på. Det finnes naturligvis grader av dette, men poenget vil være å hente ut verdier og tekstfelter fra xml-dokumentet som man ønsker å kunne søke på, og legge disse i databaseposten. Etterhvert som man

henter ut mer og mer av xml-dokumentet ligger store deler av informasjonen som felter i databasen.

Lagre all informasjonen som felter i databasen, noe som gjør søking ved hjelp av spørringer effektivt. Oppdateringer vil også kunne gjøres mye lettere enn dersom xml-dokumentet er lagret i sin helhet, fordi man bare trenger å oppdatere selve forandringen, og ikke hele Blob-filen. Dette medfører på den annen side at man trenger en forholdsvis avansert komponent som oversetter xml-dokumentet til noe som passer databaseskjemaet, og som også trenger å være tilpasset hver enkelt xml-dokumenttype. Denne komponenten hører kanskje logisk sett til i mellomlaget i modellen, men den er å få ferdig som en del av en database, slik at en relasjonsdatabase blir å anse som en xml-database [DB03]. Dersom dataen i hovedsak skal aksesseres i xml-form vil dette føre til mye ekstra arbeid med å sette dokumentene sammen, dermed krever det mer ressurser [Cha01]. I tillegg kan det være en ganske kompleks oppgave å skrive et databaseskjema for en xml-struktur som inneholder et vilkårlig antall objekter og ubegrenset rekursjon.

Å benytte en ren xml-database kan være en god løsning, selv om den ikke er veldig utbredt ennå. De etablerte, dominerende aktørene i databasemarkedet har valgt å veve inn xml-teknologi i sine eksisterende relasjonelle databasesystemer, fremfor å utvikle nye produkter fra grunnen av.

Søk

Dersom en velger å lagre xml i databasen vil det kreve andre søk- og gjenfinningsstrategier enn de vi kjenner fra tradisjonelle databasesystemer. I relasjonsdatabaser deles dataene opp i tabeller hvor strukturen dem imellom er flat. Lagring og søking i slike databaser er velkjent teknologi og bygger på sql-språket. Et tilsvarende spørringsspråk er ikke ferdigutviklet for xml-databaser, men arbeidet med w3c xml-query (XQuery) språk er kommet inn i den avsluttende fasen. Xquery er allerede implementert i den kommersielle databasen Tamino [SA].

I en arbeidsprosess-sammenheng er det interessant å finne ut hvilke typer søk og gjennfinning som skal gjøres. Utførende må ha muligheten til å søke opp den arbeidsprosessen han er ute etter. I xquery kan en gjøre spørringer etter noder og verdier av noder og benytte funksjoner for å behandle spørsmål og svar. Vi ser derfor for oss at verktøy kan bygges på toppen av xquery og at disse effektivt kan finne igjen arbeidsprosesser og andre verdier en skulle ønske fra xml-databasen.

For mere informasjon om søking ved help av xquery, se [Cha02, W3C, SA].

Kapittel 4

Prototypeimplementasjon

I dette kapittelet ser vi på en prototypeimplementasjon av arbeidsprosessytemet beskrevet over. I den første delen vises hvordan xml kan brukes for å beskrive arbeidsprosessdokumentet på en måte som oppfyller de kravene som stilles til lagringsformatet.

Det er interessant å utforske enkelte aspekter ved det beskrevne systemet, slik som avhengighetssystemet og presentasjon uavhengig av lagringsformatet. Vi har satt opp et enkelt system som implementerer denne funksjonaliteten. I del to av dette kapittelet blir teknologien og funksjonaliteten til denne prototypen beskrevet.

Til slutt blir det presentert et case som er ment å demonstrere den viktigste funksjonaliteten til systemet slik at feil kan oppdages og forbedringer foreslås.

4.1 Xml-schema for arbeidsprosessdokument

4.1.1 eXtensible Markup Language

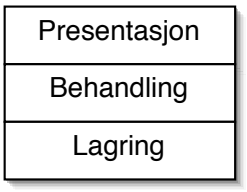
Vi har valgt å benytte xml til implementasjon av arbeidsprosessdokumentet beskrevet i kapittel 3. Xml er et subsett av den omfattende iso-standarden sgml, og gir retningslinjer for hvordan man markerer informasjon i et dokument. Dersom et dataelement skal beskrives med xml skal det benyttes en elementbeskrivelse på denne måten:

```
<elementbeskrivelse>dataelement</elementbeskrivelse>
```

Dette er kjernen i xml. I tillegg kommer muligheten til å legge ved attributter med verdier inne sammen med nøkkelordet som gjelder for dataelementet og en del regler som gjør strukturen i dokumentet mer forutsigbar. Xml definerer ingen regler for nøkkelordene eller dataelementene, noe som gjør det til et abstrakt språk. Tar man i bruk xml definerer man sitt eget dokumentformat, eller bruker en annen eksisterende dokumentdefinisjon. Man står her svært fritt, og det er dette som gir xml sin fleksibilitet og utvidbarhet.

Når man lagrer informasjon i et tekstdokument lagrer man ikke bare informasjonen, men også presentasjonen av denne. Utfra kjennskap til hva en overskrift er kan vi se hva som er overskriften i dokumentet, selv om det ikke står beskrevet noe sted hvilken del av teksten man skal anta er overskriften. Ønsker man å bygge et system der informasjonen skal aksesseres på flere forskjellige måter er det ønskelig å tilpasse presentasjonen til den aktuelle situasjonen. Et eksempel kan være å ha en nettavis som er tilgjengelig for håndholdte enheter i tillegg til konvensjonelle nettlesere. Man ønsker ikke å lagre to versjoner av en nyhet, men å lagre den på en presentasjonsuavhengig måte en gang, og så få den tilpasset til de forskjellige presentasjonsformene etterpå. Xml passer svært godt til denne presentasjonsuavhengige lagringsformen. Figur 4.1 illustrerer den enkle, men svært viktige trelagsmodellen. Informasjon som lagres som xml kan integreres med annen informasjon i behandlingslaget, og presenteres som for eksempel html-webside, pdf-dokument eller flash-grensesnitt i presentasjonslaget.

God informasjon om xml kan finnes i [RHM02, vdV02, GP01] og nettsidene til World Wide Web Consortium [W3C].



Figur 4.1: Trelagsmodellen

4.1.2 Xml-schema

Som sagt over er xml ikke et dokumentformat, men en rekke egenskaper et selvdefinert dokumentformat kan ha. For å definere slike xml-dokumentformater finnes det to metoder, ”Document Type Definition” (dtd) og W3C Xml-Schema. Ved hjelp av en dtd kan man fortelle hvilke elementer som skal inngå i et dokument og hvilke underelementer og attributter disse skal ha. En kan også bestemme typene til attributter og elementer ut ifra et gitt sett med typer. Videre kan en også sette faste eller forventede verdier. Dtd er ikke ekspressivt nok til å definere alle mulige variasjoner av xml-formater, og det har begrensede muligheter til å bestemme regler for innholdet i dataelementene.

Xml-schema er blitt lansert som arvtageren til dtd. Med et xml-schema kan man gjøre praktisk talt alt dtd tillater, og man får mange nye muligheter. Den viktigste egenskapen til xml-schema er muligheten til å definere egne simple og komplekse typer. Dette muliggjør igjen arving og derivasjon av nye typer. I tillegg har xml-schema støtte for navnerom, noe som gjør det mulig å inkludere typer og elementer fra andre xml-schemaer.

Et xml-schema er et dokument skrevet i henhold til W3Cs xml-schema definisjoner. Vårt mål ved å benytte xml-schema er å lage en struktur for innholdet i en arbeidsprosess. Strukturen vil hjelpe brukere av systemet til å lagre riktige data. Den vil også hjelpe programmerere slik at de har et fast rammeverk å forholde seg til når de skal lage løsninger for lagring og tolkning av lagret informasjon.

4.1.3 Implementasjon av Xml-schema

Vi valgte xml og xml-schema for å spesifisere lagringsformatet for arbeidsprosesser. Vi vil nå vise eksempler på hvordan dette er blitt gjort. Schema-koden finnes i sin helhet i vedlegg B.1.

Vi ser i figur 4.2 definisjonen på rotnoden i dokumenttreet vårt. Ordet xs:element

```
<xs:element name="arbeidsprosessdokument">
  <xs:complexType>
    <xs:all>
      <xs:element name="dokumentbeskrivelse" type="dokumentbeskrivelseType"/>
      <xs:element name="arbeidsbeskrivelse" type="arbeidsbeskrivelseType"/>
      <xs:element name="vedleggsliste" type="vedleggslisteType"/>
    </xs:all>
  </xs:complexType>
  <xs:key name="nøkkel">
    <xs:selector xpath="arbeidsbeskrivelse/subprosess|arbeidsbeskrivelse/valgpunkt|arbeidsbeskrivelse/
aktivitet|arbeidsbeskrivelse/slutt|arbeidsbeskrivelse/roller/rolle|vedleggsliste/vedlegg"|"dokumentbeskrivelse/
avhengigheter/avhengighet"/>
    <xs:field xpath="id"/>
  </xs:key>
  <xs:keyref name="flytref" refer="nøkkel">
    <xs:selector xpath="arbeidsbeskrivelse/start|arbeidsbeskrivelse/subprosess|arbeidsbeskrivelse/
valgpunkt|arbeidsbeskrivelse/aktivitet|arbeidsbeskrivelse/slutt"/>
    <xs:field xpath="ja-gaar-til|nei-gaar-til|gaar-til|kommer-fra"/>
  </xs:keyref>
  <xs:keyref name="rolleref" refer="nøkkel">
    <xs:selector xpath="arbeidsbeskrivelse/start/forutsetninger|arbeidsbeskrivelse/aktivitet"/>
    <xs:field xpath="hvem|ansvar"/>
  </xs:keyref>
  <xs:keyref name="avhref" refer="nøkkel">
    <xs:selector xpath="arbeidsbeskrivelse/subprosess"/>
    <xs:field xpath="identifikator"/>
  </xs:keyref>
</xs:element>
```

Figur 4.2: Definisjon av rotelementet ”arbeidsprosessdokument”

refererer til navnerommet xs, som defineres tidligere i koden, og typen element som defineres der. Navnerommet xs er definisjonen for xml-schema. Vi beskriver altså et nytt element, og gir det navnet ”arbeidsprosess” gjennom attributten name. Innenfor elementet definerer vi typen for elementet. Denne typen defineres inne i elementet, fordi den kun skal brukes i dette elementet. Dersom vi ønsket å ha flere elementer av samme type, hadde vi definert den utenfor elementet, og gitt den et navn slik at vi kunne skrevet i elementdefinisjonen hvilken type det var en instans av. Typen vi definerer er en complexType, fordi vi skal definere nye elementer innenfor den. Videre definerer xs:all at alle de tre elementene dokumentbeskrivelse, arbeidsbeskrivelse og vedleggsliste må være tilstede. Disse tre er henholdsvis av typene dokumentbeskrivelseType, arbeidsbeskrivelseType og vedleggslisteType. Disse tre typene burde kanskje ha vært definert her, ettersom de ikke skal gjenbrukes andre steder, men de ble trukket ut som egne typer for å gjøre selve koden mer oversiktlig og lesbar. Etter beskrivelsen av elementene ligger det definert en nøkkel (xs:key) og tre nøkkelreferanser (xs:keyref). Disse kommer vi tilbake til.

Vi tar nå for oss den nevnte arbeidsbeskrivelseType, som kan ses i figur 4.3 på neste side. Som tidligere påpekt er dette den delen av dokumentet som definerer rollene og arbeidsflyten i arbeidsprosessen. Vi ser at en arbeidsbeskrivelse består av en xs:sequence som igjen inneholder to xs:sequence og en xs:choice. Den første indre xs:sequence forteller oss at den skal inneholde et element som heter ”roller”

```
<xs:complexType name="arbeidsbeskrivelseType">
  <xs:sequence>
    <xs:sequence>
      <xs:element name="roller" type="rollerType"/>
      <xs:element name="start" type="startType"/>
    </xs:sequence>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="aktivitet" type="aktivitetType"/>
      <xs:element name="subprosess" type="subprosessType"/>
      <xs:element name="valgpunkt" type="valgpunktType"/>
    </xs:choice>
    <xs:sequence>
      <xs:element name="slutt" type="sluttType" maxoccurs="unbounded"/>
    </xs:sequence>
  </xs:sequence>
</xs:complexType>
```

Figur 4.3: Definisjon av arbeidsbeskrivelse

```
<xs:complexType name="aktivitetType">
  <xs:sequence>
    <xs:element name="id" type="xs:ID"/>
    <xs:element ref="dc:title"/>
    <xs:element name="beskrivelse" type="beskrivelseType"/>
    <xs:element name="hvem" type="referanseType"/>
    <xs:element name="status" type="xs:string"/>
    <xs:element name="kommer-fra" type="referanseType" maxOccurs="unbounded"/>
    <xs:element name="gaar-til" type="referanseType"/>
    <xs:element name="vedleggsref" type="vedleggsReferanseType" minOccurs="0" maxOccurs="unbounded"/
>
  </xs:sequence>
</xs:complexType>
```

Figur 4.4: Definisjon av aktivitet

av typen rollerType og et element som heter ”start” av typen startType, i den rekkefølgen. Når ikke annet er angitt kreves det i disse tilfellene at det er nøyaktig ett av hver av disse elementene. I xs:choice-gruppen derimot betyr minOccurs=”0” maxOccurs=”unbounded” at vi skal velge et av elementene ”aktivitet”, ”subprosess” og ”valgpunkt” mellom null og uendelig mange ganger[1]. Så mange ganger vi vil med andre ord. Dette er helt i tråd med den generelle oppfatningen av en arbeidsflyt. Elementet ”slutt” står i sin egen xs:sequence og har definert maxoccurs=”unbounded”. Vi kan ha flere sluttpunkter, i motsetning til startpunkter, men det kreves at det er minst ett sluttpunkt, i motsetning til aktiviteter, subprosesser og valgpunkter. Vi ser her hvordan xml-schema lar oss bestemme hvor mange ganger et element kan forekomme og må forekomme. Dette er svært nyttig, og vil i dette tilfellet sørge for at arbeidsprosesser som mangler et sluttpunkt automatisk blir avvist av dokumentbehandleren.

[1]Det er gjenstand for diskusjon hvorvidt man skal kreve at en arbeidsprosess må inneholde en subprosess eller en aktivitet. Slik det står nå må den ikke det

Figur 4.4 på forrige side viser definisjonen av aktivitetType. Vi kan se at den består av en sekvens av elementer som skal forekomme akkurat en gang. Vi ser at elementet som heter ”id” er av typen xs:ID. Det er slike elementer, som finnes i subprosess, valgpunkt, aktivitet, slutt, rolle, vedlegg og avhengighet, som vi refererer til i med xpath-uttrykket i xs:key-elementet i arbeidsprosess-dokumentdefinisjonen, se figur4.2. Med xs:keyref-elementene nedenfor kobler vi elementene som heter ”ja-gaar-til”, ”nei-gaar-til”, ”gaar-til”, ”kommer-fra”, ”ansvar” og ”identifikator” av typen xs:IDREF i forskjellige elementer til de forskjellige ”id”-er, og spesifiserer således at alle disse må eksistere. For eksempel kreves det at pilen ut fra en aktivitetsboks i et flytskjema faktisk peker til en annen aktivitet, valgpunkt, subprosess eller sluttpunkt.

Nøkkel sikrer at id-feltene som defineres i ulike underelementer av arbeidsbeskrivelsen og i vedleggslisten er unike. Dette er viktig for at referansesystemet skal fungere. Dokumentet er ikke gyldig dersom to id-felter inneholder samme verdi.

$\forall X \in$ *(Alle id-felter)* $\neg\exists X_i = X_j | i \in 0 \ldots n, j \in 0 \ldots n, i \neq j$

Flytref sier at de ulike ”går til” og ”kommer fra” elementene må ha verdier som finnes i id-feltene definert i nøkkel.

$\forall X \in$ *(Alle ”gaar til” og ”kommer fra” felter)* $\exists X = Y | Y \in Nokkel$

Rolleref sier at det må finnes en rolle tilsvarende den som refereres i aktiviteter.

$\forall X \in$ *(start/forutsetninger/ansvar og aktivitet/hvem)* $\exists X = Y$
$| Y \in Nokkel$

Avhref sier at det må finnes en avhengighet tilsvarende den som refereres i subprosess.

$\forall X \in$ *(subprosess/id)* $\exists X = Y | Y \in Nokkel$

Vi har implementert dette dokumentet for å prøve ut konseptet og vise mulighetene man har med xml-schema for å definere sitt eget robuste lagringsformat med innebygget sikring mot feil. Det finnes mange flere beskrankninger som gjelder for arbeidsprosesser enn vi har implementert, men så langt som vi har gått for å definere formatet ser det ikke ut til å mangle på funksjonalitet for dette. Standardiserte metoder for å definere beskrankninger fører til at man kan benytte standardiserte verktøy for verifikasjon av dokumentene slik vi gjør i vår implementasjon. Vi har tatt i bruk Xerces [ASFa], en xml-parser som kan verifisere xml-dokumenter mot dtd eller xml-schema. Dersom vi forsøker å parse et arbeidsprosessdokument som inneholder to startpunkter gir Xerces denne meldingen tilbake:

```
<xs:element name="avhengigheter">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="avhengighet" type="avhengighetType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:complexType name="avhengighetType">
  <xs:sequence>
    <xs:element name="id" type="xs:ID"/>
    <xs:element name="referanseIdentifikator" type="identifikatorType"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="identifikatorType">
  <xs:sequence>
    <xs:element name="id" type="xs:ID"/>
    <xs:element name="versjon" type="xs:token"/>
  </xs:sequence>
</xs:complexType>
```

Figur 4.5: Definisjon av avhengigheter

```
 Error on line 67: cvc-complex-type.2.4.a: Invalid content was
found starting with element 'start'. One of '{"":aktivitet,
"":subprosess, "":valgpunkt, "":slutt}' is expected.
```

En feil ble altså funnet i linje 67 i dokumentet, der et start-element ble funnet hvor aktivitet, subprosess, valgpunkt eller slutt var forventet. Dette stemmer ut i fra definisjonen i figur 4.3. Denne type tilbakemelding er til svært stor hjelp når man skal utvikle applikasjoner som skal generere xml-dokumenter basert på brukerens ønsker, slik vi selv opplevde da vi implementerte funksjonaliteten som beskrives i avsnitt 4.2.1 på side 74.

Vi skal se på en siste del av vårt xml-schema, delen der vi kan spesifisere hvilke andre arbeidsprosesser en arbeidsprosess er avhengig av. Figur 4.5 viser definisjonen av et element ”avhengigheter” og to typer, avhengighetType og identifikatorType. Vi ser at elementet ”avhengigheter” tillater et vilkårlig antall elementer inne i seg selv som er av typen avhengighetType. Et element av typen avhengighetType ser vi inneholder en xs:ID som gjør at den unikt kan refereres til inne i dokumentet av subprosesselementene som befinner seg i flytskjemadelen i arbeidsbeskrivelseelementet. I tillegg inneholder avhengighetsType et element an identifikatorType-typen. Denne typen brukes både til å identifisere den aktuelle arbeidsprosessen, og arbeidsprosesser som denne er avhengig av, og er et typisk eksempel på gjenbruk av egendefinerte typer. Vi identifiserer alle arbeidsprosesser en arbeidsprosess er avhengig av i en liste slik som dette fremfor å bare identifisere dem i subprosesselementene for å gjøre det enklere å finne avhengighetene i et dokument, spesielt dersom en arbeidsprosess forekommer to ganger som sub-

prosess.

4.2 Implementasjon av webapplikasjon

Vi har implementert et enkelt system for å prøve ut og demonstrere lagring av xml-baserte arbeidsprosessdokumenter i en database. Vi har kalt systemet AposServer. Dette kapittelet beskriver denne implementasjonen.

4.2.1 Funksjonalitet

AposServer skal gi brukeren grunnleggende funksjonalitet for å slå opp i og legge til arbeidsprosesser gjennom et webgrensesnitt. Dette ble oppnådd gjennom tre individuelle javaservlets.

Oppslag

Denne delen lar brukeren skrive inn en spesifikk identifikator for en arbeidsprosess bestående av id og versjonsnummer (id-versjon), velge om arbeidsprosessen skal vises i html eller xml, og slå opp i databasen, se figur 4.14 på side 89. Dersom han velger xml-visning får han se dokumentet akkurat slik det er lagret i databasen, mens html-valget vil returnere en html-webside generert ved hjelp av en xslt-prosessor som oversetter xml-arbeidsprosessdokumentet til html etter regler definert i en egen xml-fil, se figur 4.15 på side 90. For å vise hvordan systemet fungerer ved oppslag, se 4.6 på side 75. Et xslt-dokument er i seg selv et valid xml-dokument i henhold til W3Cs xslt-definisjoner. Ideen med dette dokumentet er å beskrive hva en skal gjøre med ulike deler av et xml-dokument etterhvert som det traverseres. En definerer derfor ulike ”templates” som skal benyttes for de ulike oppmerkingene. For hver oppmerking kan en altså definere hva en skal skrive ut. En kan putte inn egen tekst, hente verdier fra xml-dokumentet og kalle andre ”templates”. På den måten kan en bestemme akkurat hvordan det transformerte dokumentet skal se ut.

Opprettelse og endring av arbeidsprosesser

Det implementerte systemet støtter to ulike måter å legge inn og endre arbeidsprosesser på. Dette er en skjemabasert og en filbasert løsning. Begge to benytter den samme systeminteraksjonen som kan ses i figur 4.7 på side 76.

Skjemabasert opprettelse av arbeidsprosesser For at brukeren på en enkel måte skal kunne legge til arbeidsprosesser i databasen har vi laget et grensesnitt i form av et skjema i html/forms som fylles ut og sendes til en servlet som samler verdiene fra skjemaet og bygger et xml-dokument av dem. Se figur 4.11 på side 86. Dokumentet blir så validert og lagt inn i databasen dersom det er gyldig. En så enkel løsning som dette som bare består av en statisk webside muliggjør ikke at brukeren spesifiserer vilkårlige antall elementer av bestemte typer, slik dokumentspesifikasjonen tillater. Arbeidsprosessene man kan opprette her er derfor av begrenset kompleksitet, og består bare av ett aktivitetspunkt. De kan også bare ha avhengighet til en annen arbeidsprosess. En webapplikasjon som lar brukeren definere komplekse arbeidsprosesser krever en viss dynamikk. Jsp/php/asp egner seg fint til dette.

Filbasert opprettelse av arbeidsprosesser Denne delen har til hensikt å la brukeren laste opp en ferdig xml-representasjon av arbeidsprosessen for å få den validert og lagt til i databasen. Dette gir brukeren frihet til å spesifisere arbeidsprosessen helt etter eget ønske, så lenge den følger spesifikasjonen. Det viste seg å være betraktelig enklere å implementere dette med en tekstboks der brukeren kan lime inn koden fremfor filopplastingsfunksjonalitet, derfor ble det gjort slik.

Det er ikke meningen at brukeren skal behøve å manuelt skrive xml-kode for å bruke et slikt system. Derimot er det en naturlig funksjonalitet at andre applikasjoner skal kunne aksessere systemet og be om lagring av ferdige dokumenter.

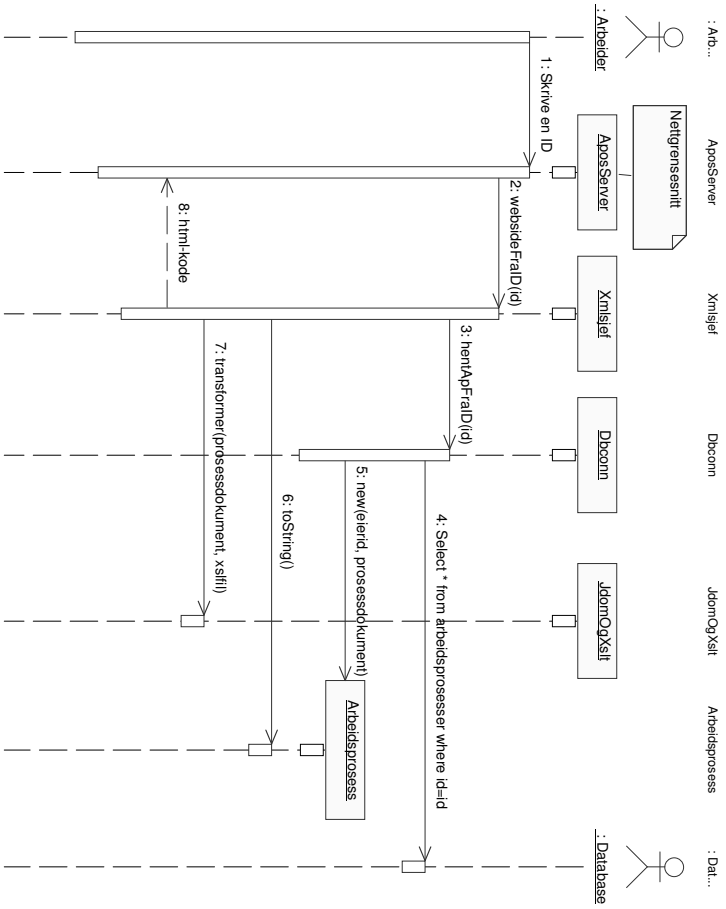
4.2.2 Klassediagram

Klassediagrammet med oversikt over klasser og metoder i AposServer systemet vises i figur 4.8 på side 77

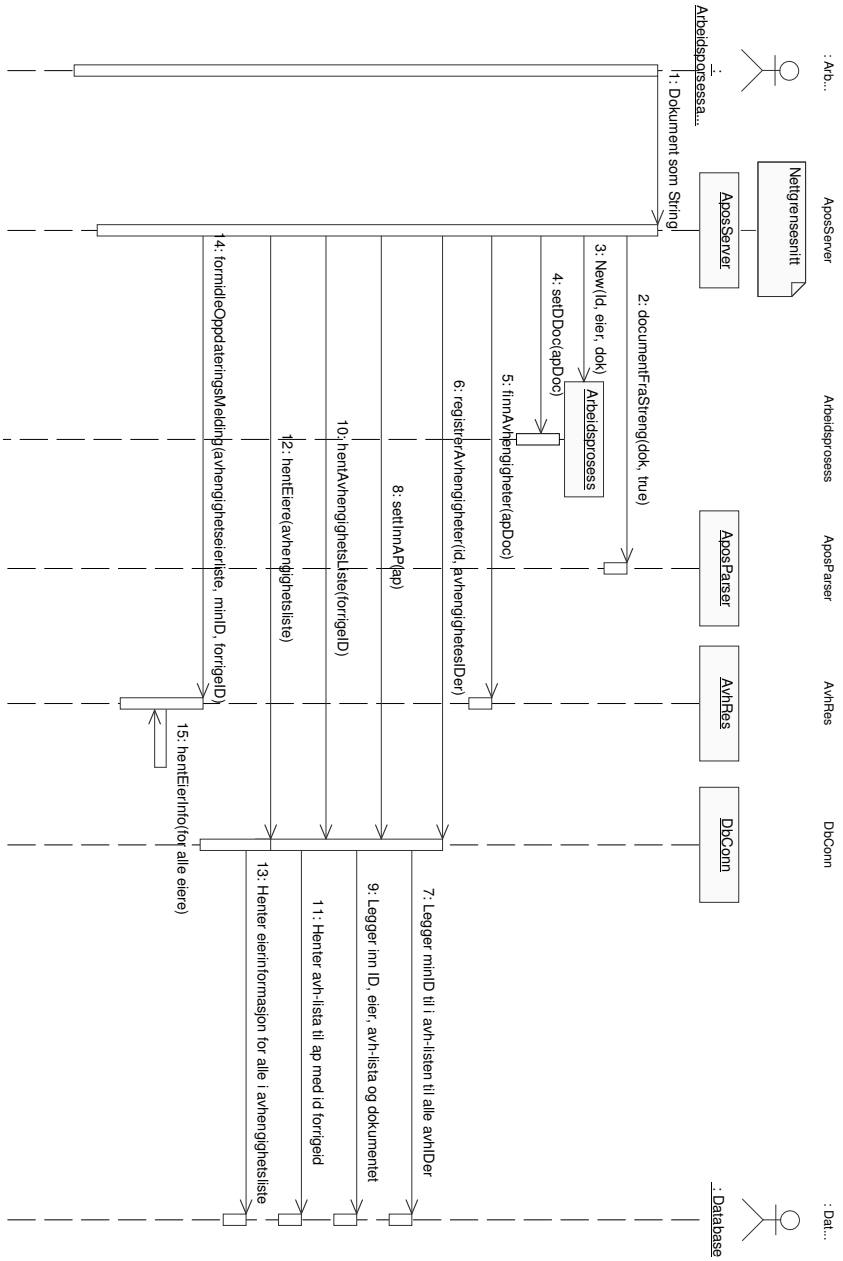
Prototypen er delt opp i en del ulike klasser. Figur 4.8 på side 77 viser oppbygningen av AposServer. Under følger en mer utførende beskrivelse av de enkelte klassene. Kildekoden finnes i tillegg C på side 135.

AposServlet, AposInnServlet, AposFilInnServlet

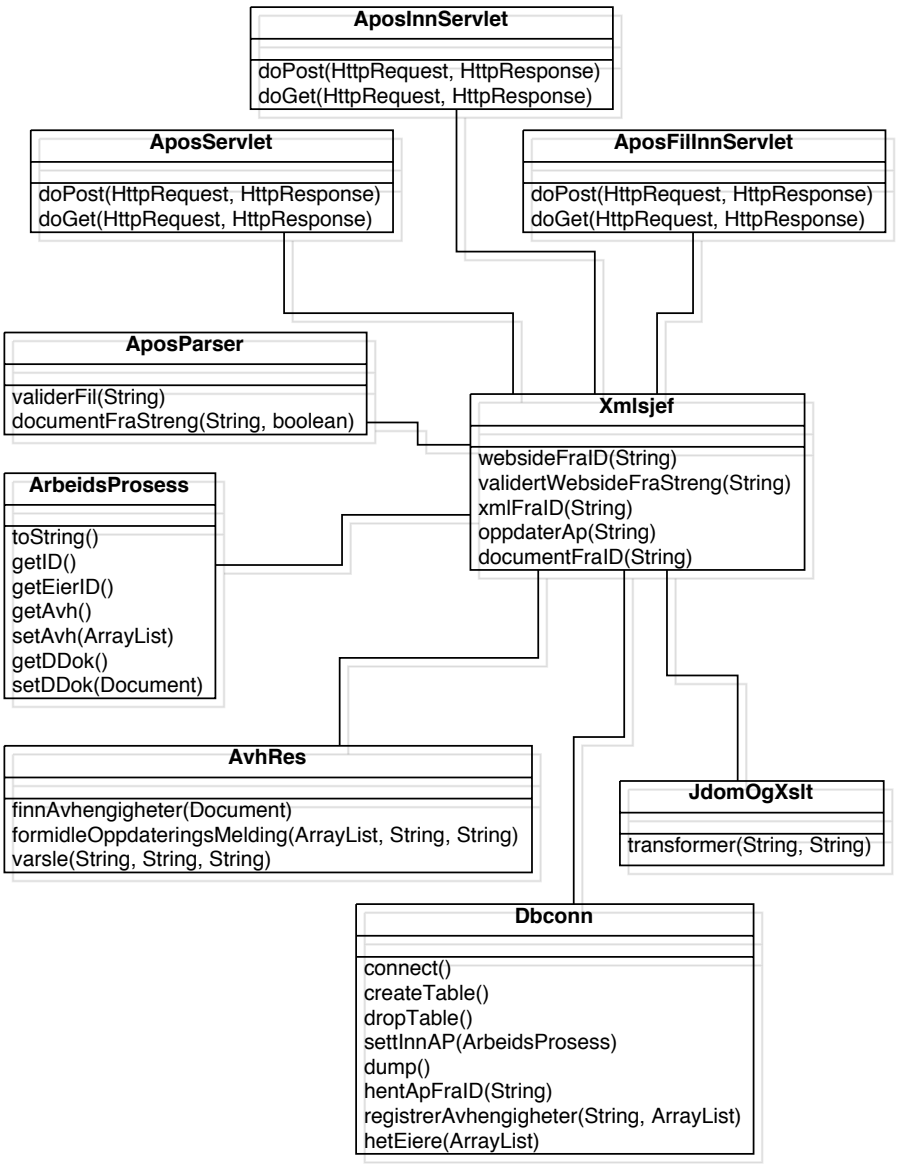
Dette er klasser som representerer de tre bruksscenariene beskrevet tidligere. En servlet er en instans av en javaklasse som går på en web- eller applikasjonsserver og tar imot forespørsler fra for eksempel websider som de nevnt her, og returnerer et svar til nettleseren. I dette tilfellet er forretningslogikken flyttet ut av



Figur 4.6: Systemets interne interaksjon ved henting av en arbeidsprosess basert på kjent ID.



Figur 4.7: Systemets interaksjon ved oppdatering av en eksisterende arbeidsprosess. I denne prototypen brukes samme funksjon for å legge til nye arbeidsprosesser.



Figur 4.8: Klassediagram for AposServer

servletklassene, slik at det skal være lett for flere typer eksterne forespørsler å bruke samme funksjonalitet. AposInnServlet og AposFilInnServlet bruker nøyaktig samme metodekall, forskjellen ligger i at AposInnServlet selv bygger xml-dokumentet basert på verdiene fra feltene i websiden som sendte forespørselen, mens AposFilInnServlet mottar xml-dokumentet ferdig formatert. Begge klassene benytter metoden oppdaterAp i klassen Xmlsjef. AposServlet er klassen som mottar brukerens forespørsler om oppslag i arbeidsprosessene. På bakgrunn av brukerens visningsvalg kaller den enten websideFraId eller xmlFraId i sin Xmlsjef-instans.

Xmlsjef

Vi har samlet kontrollflyt og metoder som skal være synlige fra andre pakker i en klasse som kalles Xmlsjef. Hver metode her vil representere et usecase der enten en bruker eller et annet system er aktør. Datamanipulasjon skjer som hovedregel ikke her, men i andre klasser som igjen bruker andre biblioteker.

AposParser

For å gjøre validering og parsing av dokumenter benytter vi oss av Xerces [ASFa] og Jdom [TJP]. Jdom er et høynivåverktøy som bruker Xerces for validering og parsing, og leverer dokumentobjekter og metoder for å manipulere disse. I de tilfellene der man ønsker å hente ut eller sette spesifikke verdier i et xml-dokument er det bekvemt å abstrahere vekk manipulasjonen av teksten direkte.

ArbeidsProsess

Dette er en datatypeklasse. Vi ønsker å lagre alle data som omhandler selve arbeidsprosessen på en plass slik at det er enkelt å finne fram for senere manipulering. Det som lagres er prosessens ID, en referanse til eier, et sett identifikatorer til prosesser denne avhenger av samt et Jdom-objekt av arbeidsprosessdokumentet. Har ellers kun funksjonalitet for å hente og sette variable.

AvhRes

Funksjonaliteten for behandling av avhengigheter er skilt ut i en egen klasse. Klassen jobber med dokumentobjektrepresentasjonen av dokumentet for å finne

ut hvem denne er avhengig av og gjør også en jobb med å formidle melding til eiere av berørte arbeidsprosesser.

JdomOgXslt

Visning av xml-dokumenter på andre formater krever en transformasjon. Dette gjøres her ved hjelp av et xslt-dokument og en xslt-prosessor. Xslt-dokumentet beskriver regler for transformeringen av xml-koden, i vårt tilfelle til html-kode. Vi benytter Xalan [ASFa] til xsl-transformasjoner. Dette har lite med arbeidsprosesser å gjøre, men illustrerer en mulig visning av informasjonen i databasen.

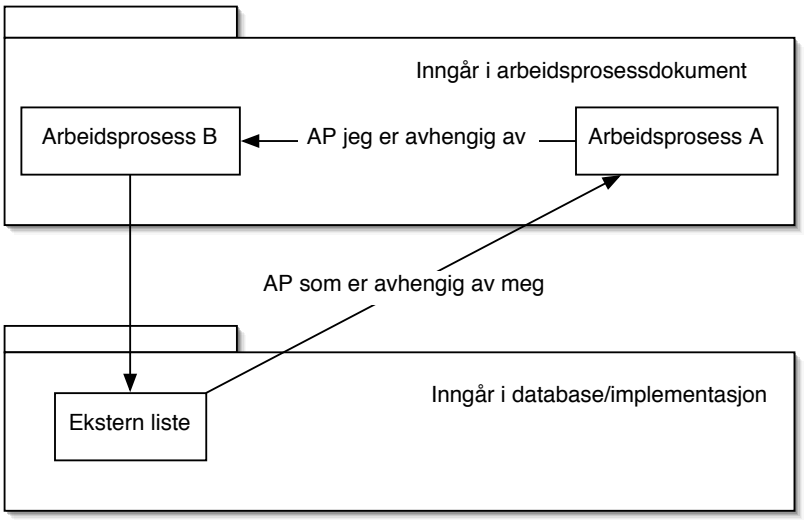
Dbconn

Samhandlingen med databasen er utført ved hjelp av jdbc. Det blir her utført både relativt enkle operasjoner som innsetting og uthenting, men det skjer også en del datamanipulering på bakgrunn av den informasjonen som ligger i databasen og den som sendes inn. Det kan her nevnes oppdatering av eksterne avhengighetslister og bygging av lister med eiere til arbeidsprosesser.

4.2.3 Avhengighetsbehandling

Som beskrevet tidligere blir det registrert i enhver arbeidsprosess hvilke arbeidsprosesser denne avhenger av. Vi skal nå se på hvordan vi følger opp disse avhengighetene i systemet. Ut fra dokumentformatet vi spesifiserte i kapittel 4.1 kan man ikke i en arbeidsprosess se hvilke andre arbeidsprosesser som igjen avhenger av den. Avhengigheter er med andre ord kun referert en vei. Dette er i tråd med at vi mener det ikke burde gjøres forandringer i et dokument dersom et nytt dokument tar det i bruk som subprosess.

Når en arbeidsprosess oppdateres har vi behov for å formidle melding om dette til de som er ansvarlige for arbeidsprosessene som avhenger av den. Dette er fordi vi ikke ønsker at andre enn arbeidsprosessansvarlig skal kunne gjøre forandringer i en arbeidsprosess gjennom å forandre i subprosessen. Når en arbeidsprosess oppdateres, vil den lagres som en ny versjon, og den gamle vil fortsatt være tilgjengelig. Arbeidsprosesser som avhenger av den som ble oppdatert refererer fortsatt til den gamle versjonen, inntil ansvarlige for disse, som følge av at de fikk nevnte type melding, har godkjent at de avhenger av den nye versjonen. Denne endringen



Figur 4.9: Oppbygning av referanser for en avhengighet mellom to arbeidsprosesser

fører normalt ikke til nytt versjonsnummer for den avhengige prosessen[2], ettersom dette ville føre til at en endring ble propagert helt opp i hierarkiet unødvendig.

For at vi skal kunne sende disse meldingene om oppdatering må vi ha muligheten til å finne de arbeidsprosessene som er avhengige av den vi forandrer, slik at vi vet hvor meldingene skal sendes. I vår implementasjon har vi en liste liggende i databasen for hver arbeidsprosess, der man registrerer avhengighetene i motsatt retning av slik de er registrert i selve dokumentet. Se figur 4.9.

Figur 4.10 på neste side viser metoden som sørger for å registrere arbeidsprosessen som skal legges til i databasen i avhengighetslistene. Den tar imot den aktuelle arbeidsprosessens identifikator og en liste over dens subprosesser, henter alle disses eksterne avhengighetslister, legger identifikatoren inn i dem, og legger dem tilbake i databasen. Dersom en av arbeidsprosessene ikke finnes i databasen blir en UgyldigeAvhengigheterException kastet og naturligvis blir ingen endringer lagret.

[2]Det ville være på sin plass å ha mindre versjonsnnumre som kunne reflektere disse små endringene

```
protected synchronized void registrerAvhengigheter(String minID,
    ArrayList avhengighetsIDer) throws UgyldigeAvhengigheterException {

  try {
    PreparedStatement ut = c
        .prepareStatement("SELECT avh FROM arbeidsprosesser where id=?");
    PreparedStatement innigjen = c
        .prepareStatement("UPDATE arbeidsprosesser SET avh=? WHERE id=?");
    HashMap hm = new HashMap();

    for (int i = 0; i < avhengighetsIDer.size(); i++) {
      ut.setString(1, (String) avhengighetsIDer.get(i));
      rs = ut.executeQuery();
      if (!rs.next())
        throw new UgyldigeAvhengigheterException("En arbeidsprosess ble ikke funnet i DB");
      hm.put((String) avhengighetsIDer.get(i), (ArrayList) rs.getObject("avh"));
    }

    ArrayList al;

    for (int i = 0; i < avhengighetsIDer.size(); i++) {
      al = (ArrayList) hm.get((String) avhengighetsIDer.get(i));
      if (al == null) al = new ArrayList();
      al.add(minID);
      innigjen.setObject(1, al);
      innigjen.setString(2, (String) avhengighetsIDer.get(i));
      innigjen.execute();
    }

  } catch (SQLException e) {
    e.printStackTrace();
  }
}
```

Figur 4.10: Utdrag fra Dbconn.java, metode registrerAvhengigheter

4.2.4 System

Vår implementasjon baserer seg på bruk av Apache Tomcat web-/applikasjonsserver [ASFb] og MySQL database [MA]. Tomcat er åpen kildekode skrevet i Java, og har støtte for JSP 2.0 og Java Servlet 2.4. AposServer er en servletbasert webapplikasjon som aksesserer MySQL-databasen gjennom jdbc. MySQL er portabel, åpen kildekode. Serveren og databasen er blitt kjørt på Mac OS X, men kan brukes på de fleste plattformer.

Vi gjorde disse valgene basert på tilgjengelighet og plattformuavhengighet. Valget av database sto mellom PostgreSQL og MySQL. Disse er begge åpen kildekode, men det er likevel forskjeller å merke seg. PostgreSQL er mer avansert og funksjonsrik, mens MySQL anses å være mer effektiv. Dette spilte ingen stor rolle for oss, så lenge jdbc var implementert for dem begge. Vi var uansett kun interessert i standardfunksjoner, og ytelse var ikke en bekymring for oss. En annen forskjell er lisensene de distribueres under; PostgreSQL bruker en BSD-lisens, som er svært fri og fleksibel, mens MySQL er tilgjengelig enten under GNU General Public License eller en kommersiell lisens. Dette spiller en rolle dersom man ønsker å utnytte systemet kommersielt.

Vi valgte å bruke Tomcat med Java Servlets fordi dette er en vel ansett og utbredt plattform. Applikasjoner utviklet på Tomcat kan brukes på alle J2EE-servere.

4.3 Muligheter for utvidelse

Vårt bidrag har vært å definere et system fra grunnen av med en arkitektur og implementasjon av en prototype for å vise dens kjernefunksjonalitet. Det gjenstår derfor mye arbeid før et ferdig produkt foreligger, funksjonaliteten kartlagt i avsnitt 2.3 tatt i betraktning. Det er likevel elementer som er nærliggende å implementere slik at AposServer kan tas i bruk som modul i et større system. En arbeidsprosessdatabase står i de fleste tilfeller ikke alene, men må samhandle med bedriftens eksisterende systemer, for eksempel arbeidsordresystemer og personell/kompetansesystemer. I tiden etter at implementasjonen vår var ferdig har vi fått vite at en viktig funksjonalitet i prosessbehandlingen vil være eksplisitt lagring av krav og beste praksis som separate entiteter som igjen kan koples til den faktiske utførelsen av arbeidsprosesser. Dette vil være en naturlig utvidelse av vårt system og vår datamodell dersom det ikke vil være å finne i et separat system.

Web Services[W3C] er et stadig mer populært grensesnitt for kommunikasjon mellom applikasjoner over internett. Web Services benytter seg av Simple Ob-

ject Access Protocol, soap, og Web Service Description Language, wsdl. I dette tilfellet ville det være interessant å implementere web services for vårt systems funksjonalitet. Slik det står i dag ville denne funksjonaliteten først og fremst være å kunne oppdatere og lagre nye arbeidsprosesser og få tilgang til validerte xml-dokumenter i databasen. Det vil også være interessant å bruke web services mot andre systemer for å få tilgang til ressurser som krav- og rolledefinisjoner, dersom disse eksisterer.

4.4 Casestudie - Resertifisering av PZV

I denne delen skal vi utføre en test av prototypen for å demonstrere den sentrale funksjonaliteten. Vi vil gjøre dette ved å skissere et hendelsesforløp basert på use-casene utviklet i kapittel 2.3 på side 18. De ulike aktørene vil utføre handlinger og systemet vil respondere. Vi dokumenterer testen med skjermbilder og annen nødvendig informasjon. Til slutt vil vi oppsummere og bruke resultatene og erfaringene til å skissere videre implementasjon og hvor forbedringspotensialet ligger.

I testcaset tas det utgangspunkt i den tidligere nevnte arbeidsprosessen ”resertifisering av pzv”. I korttekst går denne arbeidsprosessen ut på å planlegge, klargjøre, gjennomføre og tilbakestille en ventil som skal forsikre at trykket i en olje-/gass-/biprodukt-separator ikke blir for stort. I forbindelse med separatoren står det flere uavhengige pzv-ventiler. Med jevne mellomrom, eller etter spesielle hendelser, er det nødvendig å resertifisere disse. Det vil si å ta dem ut av bruk og sjekke dem for slitasje og skader, kalibrere dem og sette dem tilbake. Dette for å forhindre at ulykker skjer. Vår tilgjengelige informasjon i forbindelse med dette caset begrenser seg til tre dokumenter på tre ulike styringsdokumentnivåer. Det øverste, ”Hovedprosess”, forklarer hele gangen i prosessen, det neste forklarer ”Gjennomføring” i mer detalj og er altså en subprosess av hovedprosessen. Som en subprosess til gjennomføringen igjen finner vi en detaljert beskrivelse av ”demontere blinding”.

4.4.1 Hendelsesforløp

Her skisseres testcaset som brukes for å vise prototypens funksjonalitet:

1. Arbeidsprosessansvarlig legger inn informasjonen om ”resertifisering av pzv”
2. Brukeren henter hovedprosessen til ”resertifisering av pzv” basert på identifikator
3. Brukeren fordyper seg i ”gjennomføring” ved hjelp av identifikatoren han fant i hovedprosessen
4. Brukeren fordyper seg i ”demontere blinding” ved hjelp av identifikatoren han fant i gjennomføringsprosessen
5. Brukeren gir tilbakemelding om at ”demontere blinding” kan gjøres mer effektivt

6. Prosessansvarlig for ”demontere blinding” leser tilbakemeldingen. Det er ulike prosessansvarlige for ”demontere blinding” og ”gjennomføring”.
7. Prosessansvarlig for ”demontere blinding” gjør endringer av prosessen basert på tilbakemeldingen
8. Prosessansvarlig for ”gjennomføre” får beskjed om at endring er gjort i ”demontere blinding”
9. Prosessansvarlig for ”gjennomføre” godkjenner endringen som fornuftig for gjennomføringen og linker til den nye versjonen av ”demontere blinding”

Som en kommentar før vi ser på hendelsesforløpet nevnes at all nødvendig funksjonalitet ikke er fullstendig implementert. På de punktene der implementasjonen ikke er tilstrekkelig god vil dette kommenteres og en reaksjon skisseres ut ifra den forventede virkemåten.

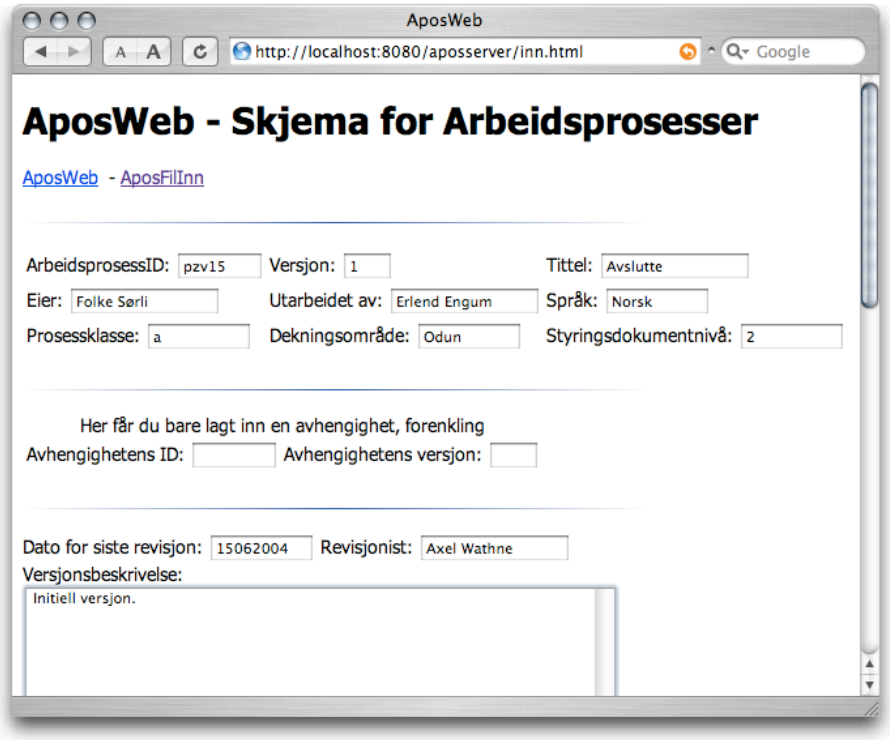
4.4.2 Gjennomføring av case

Vi gjennomførte caset som planlagt. Under følger en beskrivelse av hva som skjedde underveis. Vi går gjennom steg for steg og beskriver hva som skjedde både med tekst og bilder. Etter gjennomgangen ser vi på hvilke feil og mangler som en full implementasjon må ta for seg.

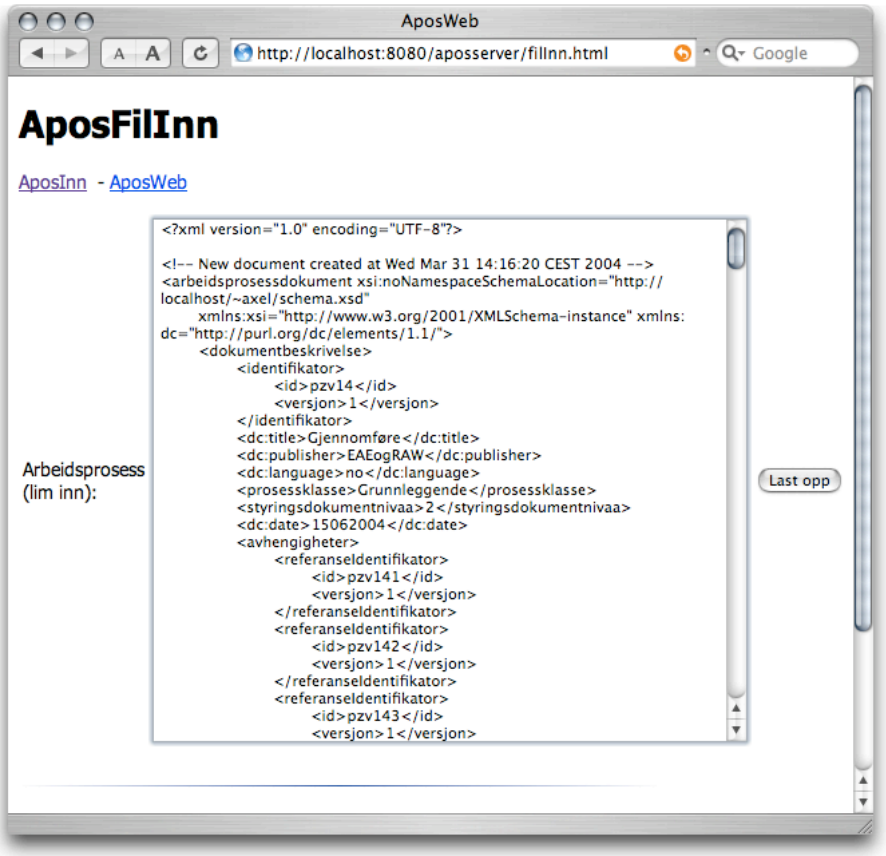
Punkt 1: AposServer ble aksessert og de ulike aktivitetene ble lagt til. Til dette benyttet vi først grensesnittet AposInn som vist i figur 4.11 på neste side. Vi fulgte hele veien med i databasen at prosessene ble lagt til. Dette viste seg å være nødvendig da vi opplevde noen runtime-feil. Disse ble til i tilbakemeldingen til brukeren, etter at arbeidsprosessdokumentet var lagret i databasen, og hadde ingen videre innvirkning på arbeidet.

Når de prosessene vi ikke har detaljert beskrevet var lagret i databasen kunne vi skrive de mer detaljerte prosessene, ”demontere blinding”, ”gjennomføre” og ”hovedprosess”. Disse ble lastet inn i databasen med AposFilInn-grensesnittet som vist i figur 4.12 på side 87. Disse prosessene ble skrevet for hånd, og da de ble lastet inn fant xml-validatoren frem til skrivefeil og referanseintegretetsfeil slik at disse enkelt og raskt kunne rettes.

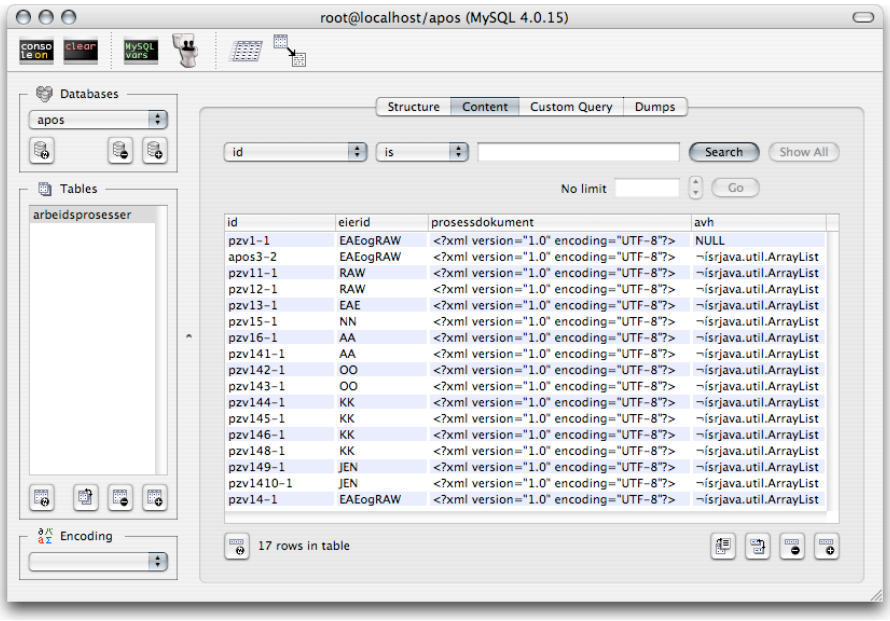
Etter at alt var lagret kunne vi se i databasen at alle prosessene var innlagt og at alle bortsett fra hovedprosessen nå hadde lagret informasjon om høyerenivå-prosesser som var avhengig av dem. Dette viser vi i figur 4.13 på side 88.



Figur 4.11: Skjermbilde fra AposServers arbeidsprosess-skjema



Figur 4.12: Skjermbilde fra AposServers filinput grensesnitt



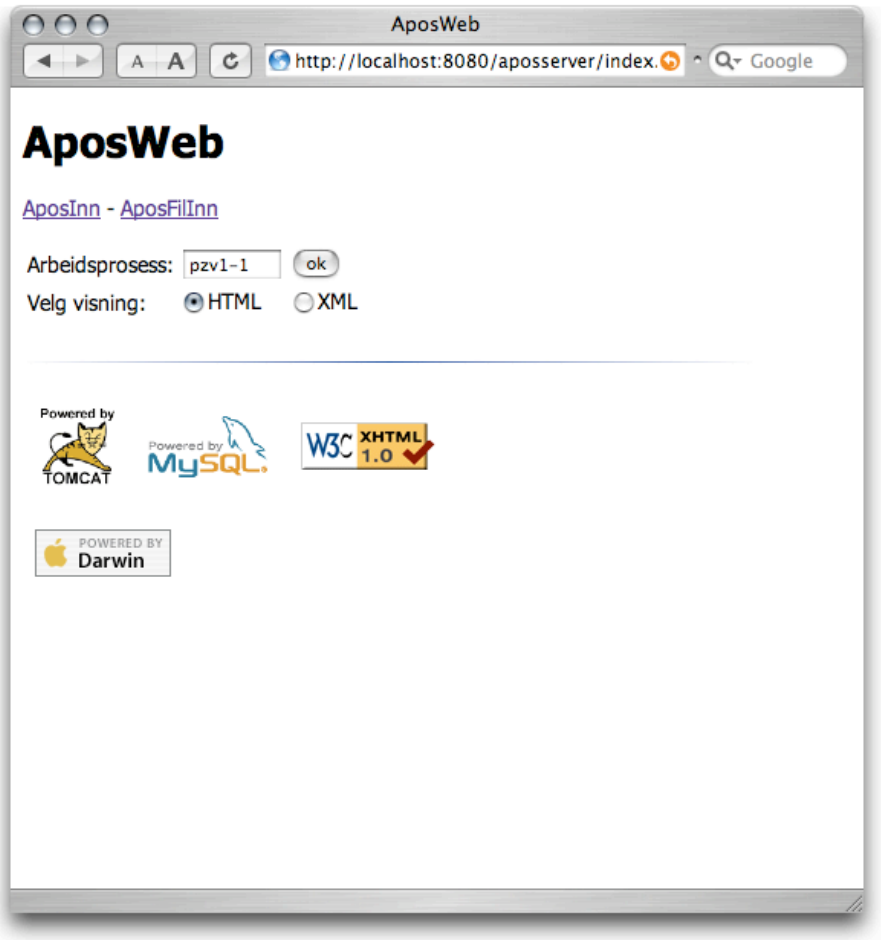
Figur 4.13: Skjermbilde fra databasen etter at alle arbeidsprosessene er lagt inn

Punkt 2: Brukeren skulle her aksessere hovedprosessen ”Resertifisering av pzv” og vi brukte grensesnittet AposWeb som vist i figur 4.14 på neste side til dette. Vi skrev inn prosesskoden ”pzv1-1”. Visningsformen ”html” ble valgt og ”ok” ble trykket. Responsen ble som vist i figur 4.15 på side 90.

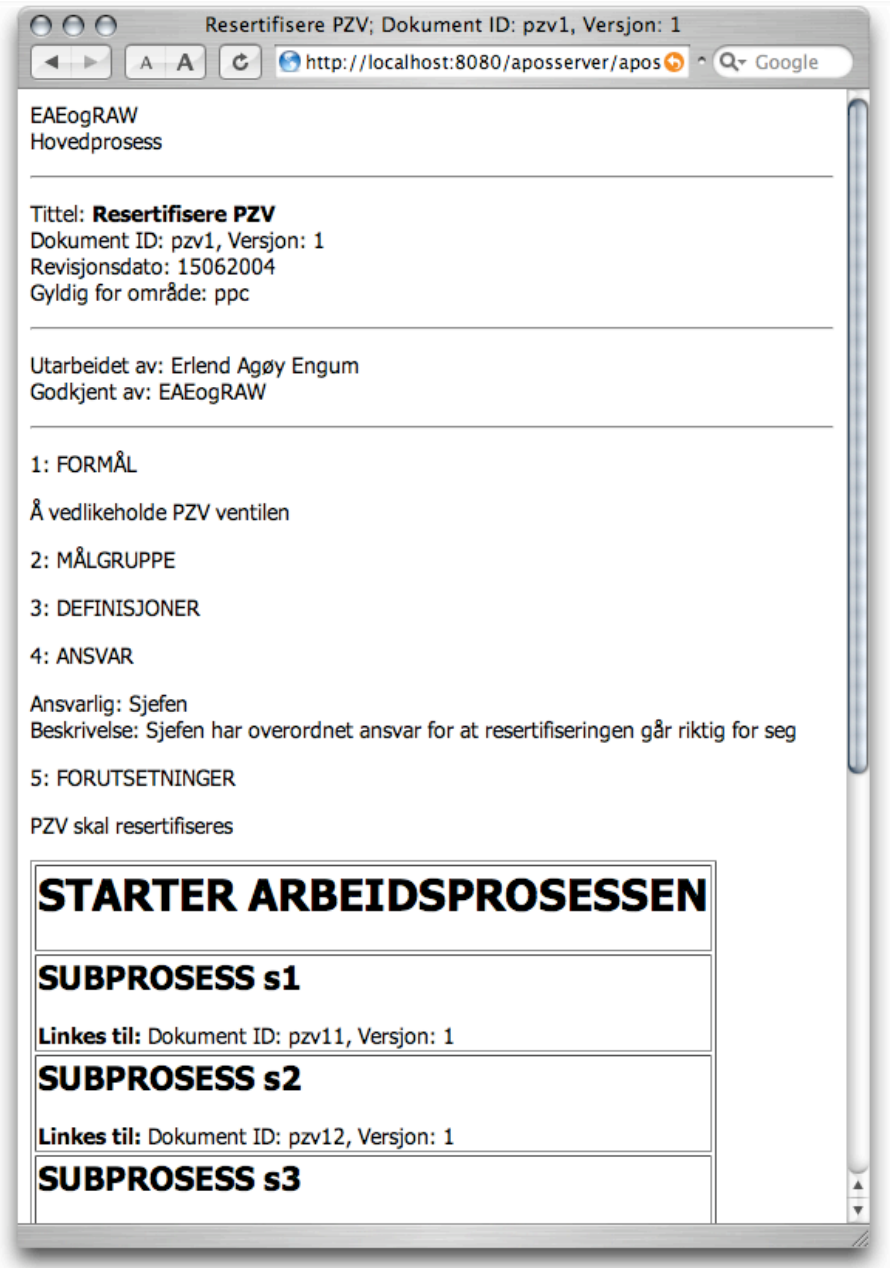
Punkt 3: Brukeren skulle nå fordype seg i gjennomføring. Dette viste seg å være vanskelig da han kun så en liste med ulike subprosesser med tilhørende identifikatorer. Heldigvis hadde brukeren forkunnskap og kunne fort fastslå at prosessen med identifikator pzv14 inneholdt det han lette etter. Vi brukte koden ”pzv14-1” og fant mer detaljert informasjon om flyten.

Punkt 4: Når han så på flyten til gjennomføringen var dette egentlig like forvirrende som hovedprosessen. Han kunne se at prosessen delvis foregikk i parallell, eller at han kunne velge ulike veier. Målet var her å fordype seg i demonteringen av blinding. Det var bare en prosess som skilte seg ut, den med navn ”apos3-2”.

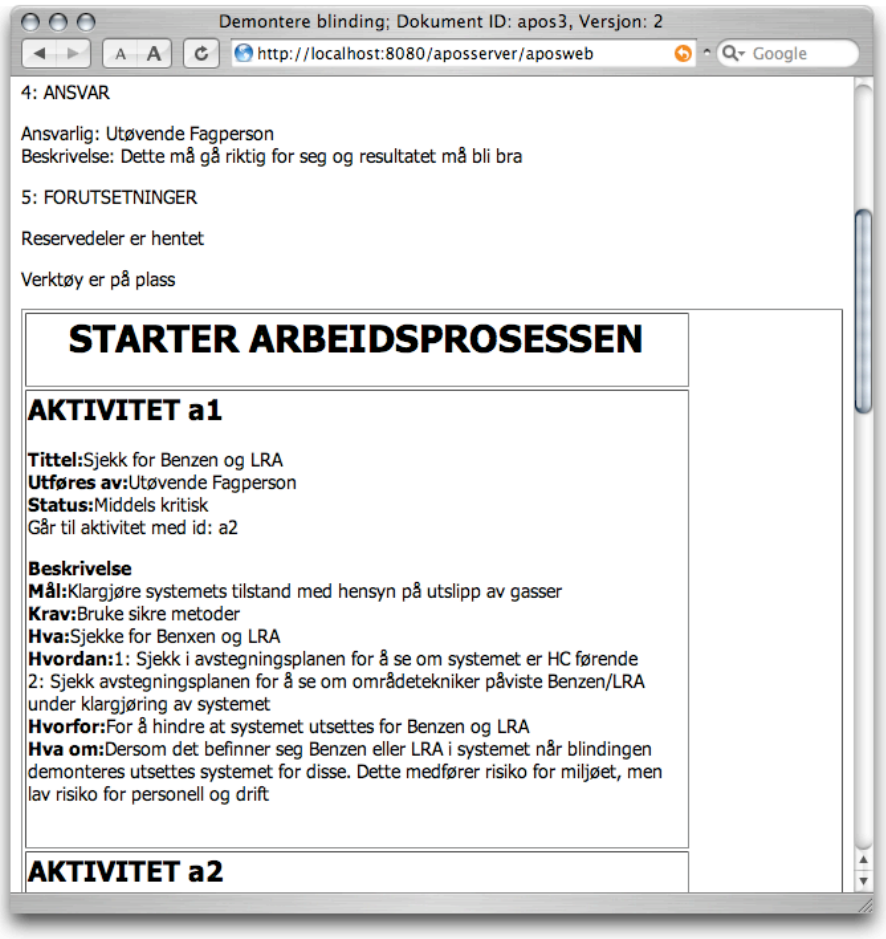
Punkt 5: Opphenting av arbeidsprosessen ”apos3-2” gav resultatet vist i figur 4.16 på side 91. Etter grundig gjennomgang av prosessen og vedleggene fant han enkelte elementer han ikke var enig i. Brukeren bestemmer seg for å gi tilbakemelding til arbeidsprosessansvarlig om hva som kan gjøres bedre. Han åpner derfor tilbakemeldingsgrensesnittet og skriver sine kommentar-



Figur 4.14: Skjermbilde fra AposServers oppslagssside



Figur 4.15: Skjermbilde fra AposServers HTML-visning av hovedprosess



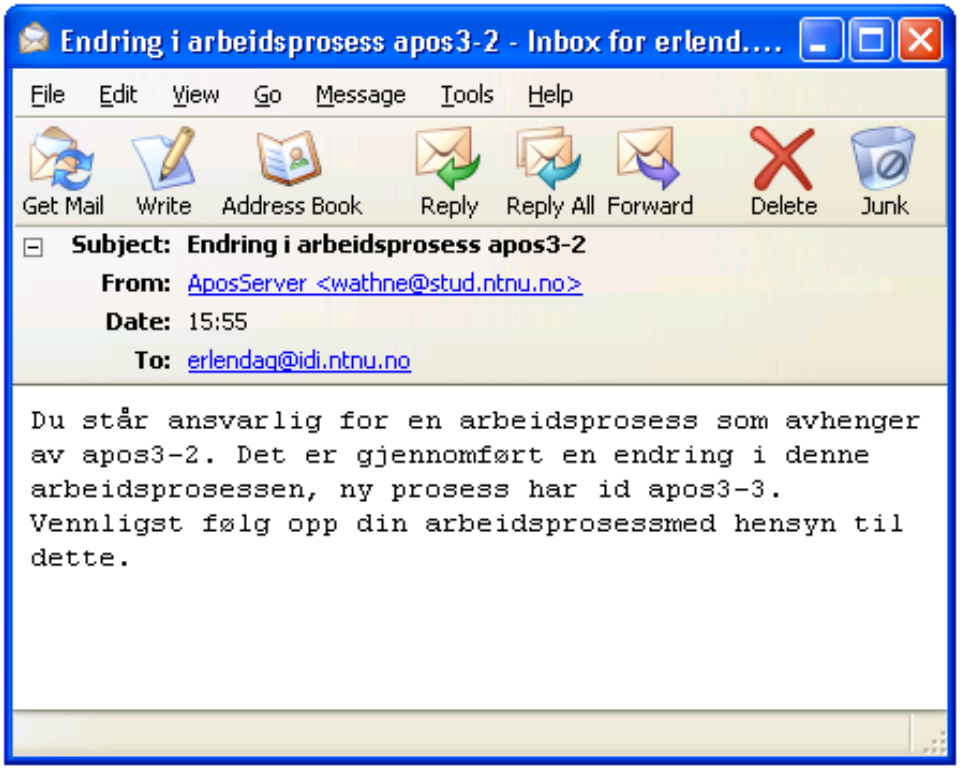
Figur 4.16: Skjermbilde fra AposServers HTML-visning av demontere blinding

er. Denne funksjonaliteten er ikke implementert og vi kan derfor ikke vise skjermbilde.

Punkt 6: Tilbakemeldingen tikker med en gang inn på arbeidsprosessansvarliges e-post og en sms-notis om tilbakemeldingen blir gitt. Prosessansvarlig for ”demontere blinding” går inn i systemet og leser tilbakemeldingen. Denne funksjonaliteten er ikke implementert og vi kan ikke derfor ikke vise skjermbilder og meldinger.

Punkt 7: Prosessansvarlig for ”demontere blinding” gjør endringer basert på tilbakemeldingen og lagrer den nye arbeidsprosessen som ”apos3-3”.

Punkt 8: Prosessansvarlig for ”gjennomføre” mottar en e-post som sier at en endring har skjedd i ”apos3-3”, se figur 4.17 på neste side. Han er da pliktig å



Figur 4.17: Mailen som gjør arbeidsprosessansvarlig oppmerksom på oppdatering i underprosess

sjekke ut hvorvidt denne endringen er riktig i henhold til kravene som stilles til ”gjennomføre”.

Punkt 9: Endringen er konsistent med gjennomføring. Prosessansvarlig endrer referansen. Dette avslutter caset.

4.4.3 Kommentarer og lærdom

Hva har vi lært av dette caset? For det første har vi funnet mange større og mindre feil som med enkle oppdateringer kan gjøre systemet langt mere brukervennlig. Her følger en punktliste med disse svakhetene.

- Subprosesser er ikke beskrevet med navn og andre attributter
- Det finnes ikke direkte hyperlenke fra en subprosess i et dokument til doku-

mentet denne viser til

– E-posten som sendes til prosessansvarlig for en overordnet prosess når underprosessen oppdateres mangler prosessidentifikasjon til den faktisk overordnede prosessen.

Caset førte også til at en del konstrukter ble lagt til i ontologien. Det viktigste av disse er delingspunkt som er en naturlig del av en prosessflyt når det skal gjøres flere aktiviteter i parallell. Andre elementer er årsak, risiko og konsekvensfelter i beskrivelsen til aktivitet.

Ved siden av å vise en del svakheter ved systemet, noe en prototyp er ment å gjøre, har vi også fått bekreftet det vi ønsket. For det første har vi vist at lagring kan skilles fra behandling som igjen kan skilles fra presentasjon. For det andre har vi gjort en fungerende implementasjon som kan behandle endringer i arbeidsprosesser. Når endringene skjer forblir systemet stabilt og fungerende. Vi sikrer også kvaliteten ved å la ansvarlig for overordnede prosesser godta endringen som gyldig for sin prosess, eller forkaste den som ikke brukbar.

Kapittel 5

Konklusjoner og videre arbeid

5.1 Konklusjoner

Arbeidsprosesser kan formaliseres slik at de kan danne grunnlag for opplæring og arbeidsutførelse ved senere anledninger. Det er derfor viktig at prosessene bevares på en konsekvent, oversiktlig og stabil måte. Dette må gjøres for at en senere skal kunne standardisere, oppdatere og presentere dem.

Denne rapporten viste ulike aspekter ved arbeidsprosesser. Først gikk vi gjennom kunnskapsbehandling i et bredt perspektiv. Det viste at et arbeidsprosessystem er kodifisert, som vil si at ekspertenes underforståtte kunnskap blir overført til eksplisitt form. Denne kunnskapen må lagres i et datasystem slik at informasjonen kan endres og gjenfinnes på en måte brukeren oppfatter som enkel og fleksibel. Dette vanskeliggjøres av at arbeidsprosesser er ordnet som flerbruksprosesser med flere referanser både opp og ned i hierarkiet. Kontinuerlig utvikling av arbeidsprosessene kan potensielt forringe konsistensen i avhengighetene mellom disse.

Usecase-analysen kartla bruksområde og interaksjon mellom brukerne og et arbeidsprosessystem. Det hjalp oss i å identifisere og spesifisere behovet for lagring av denne typen informasjon. Dette omfatter struktur, datatyper, metainformasjon og multiplisitet i dokumentformatet, og en arkitektur for implementasjon av et arbeidsprosessystem.

Xml er en teknologi som er velegnet for representasjon av arbeidsprosesser. Xml-schema kan benyttes for å definere et lagringsformat som sikrer integritet i innholdet. Et veldefinert xml-format spesifiserer ikke hvordan informasjonen skal presenteres, men beskriver hva informasjonen er, slik at den kan presenteres på forskjellige måter. Vi definerte et format vi kaller *arbeidsprosessdokument* der vi prøvde ut metoder for å sikre intern integritet i dokumentet.

Arbeidsprosessdokumentene kan forholde seg til store datamengder samtidig som de også forholder seg til andre arbeidsprosesser. Et rammeverk for systemet rundt dokumentene må kunne håndtere dette samtidig som det skal være brukervennlig og oppfylle de kravene som stilles til funksjonalitet. Arkitekturen vi har skissert baseres på analysene og krav som stilles til håndtering av multimediadata. Vi foreslår en tradisjonell trelagsarkitektur med replisering av mediabiblioteker. Dette gjør det mulig å lage presentasjonsverktøy der de mest ekspressive virkemidlene, som film og animasjon, tas i bruk for å vise frem arbeidsprosessen.

Med prototypen har vi gjort det mulig å legge til, hente frem og endre arbeidsprosesser. Når nye prosesser legges til vil dette registreres i en databasepost hos alle subprosesser. Slik er det mulig å gi beskjed til alle som avhenger av en flerbruksprosess når denne endres. Dette forhindrer råtne lenker i systemet og hjelper til slik at beste praksis til envher tid gjenspeiles.

5.2 Videre arbeid

Etter arbeidet med denne oppgaven står vi igjen med en del ubesvarte spørsmål vi mener det kan være interessant å arbeide videre med. Her vil det skisseres noen av dem, og hvordan vi ser for oss at det er best å begynne med dette arbeidet.

Et område der mye arbeid gjenstår er multimediabehandlingen. Vi har foreslått en arkitektur for lagring og fremføring av store datamengder ved hjelp av et multimediabibliotek og replisering av dette. Hvor mange replikater trenger en og hvor de geografisk sett skal befinne seg er interessante spørsmål. I den forbindelse ser vi også for oss muligheten til å distribuere ulike deler av multimediabiblioteket til ulike lokasjoner slik at filer som brukes ofte ligger nær bruker og andre ligger lengre borte. Utvikling av algoritmer for dette er en interessant oppgave.

Det kan også være interessant å analysere behov for sikkerhet i forbindelse med en slik distribuert lagring og overføring av store datamengder.

Hvor stor del av mediefilene som skal annoteres og hvordan dette skal gjøre er et annet spørsmål. En kan si at kvaliteten på presentasjonen og effektiviteten i forbindelse med overføring og navigasjon kan økes betraktlig med god annotering. Selve arbeidet med beskrivelse av multimediaressurser er tids- og arbeidskrevende. Forbedrede metoder for dette er derfor nødvendig om en ønsker å øke bruken av multimedia i dynamiske presentasjoner.

Da vi ikke har lagt noe særlig arbeid i framvisningsdelen av systemet er det arbeid som gjenstår før brukeren får presentert arbeidsprosessene tilpasset de ulike behovene som finnes. Vi har skissert mulige bruksområder og hvilke plattformer som praktisk kan benyttes. Eksempelvis er det svært interessant å se på mulighetene som ligger i bruk av håndholdte enheter til presentasjon av arbeidsprosesser.

Kapittel 6

Bibliografi

Bibliografi

[AG04] Elias M. Awad og Hassan M. Ghaziri. *Knowledge Management*. Prentice Hall, international edition utgave, 2004.

[Alv01] H Alvestrand. Tags for the identification of languages. http://www.ietf.org/rfc/rfc3066.txt, januar 2001.

[App02] Apple Computer, inc., http://developer.apple.com/darwin/projects/streaming/. *Quicktime/Darwin Streaming Server Administrator's Guide*, 2002.

[APW03] Mark S. Ackerman, Volkmar Pipek og Volker Wulf, redaktører. *Sharing expertise : Beyond knowledge management*. The MIT Press, 2003.

[ASFa] Apache Software Foundation. The Apache Xml Project. http://xml.apache.org/.

[ASFb] Apache Software Foundation. Tomcat Webserver. http://jakarta.apache.org/tomcat/.

[Cha01] Michael Champion. Storing xml in databases. *Enterprise Application Integration Journal*, (2):53–55, 2001.

[Cha02] Don Chamberlin. Xquery: An xml query language. *IBM Systems Journal*, 41(4):597 – 615, 2002.

[Coh03] Bram Cohen. Incentives build robustness in bittorrent. http://bitconjurer.org/BitTorrent/bittorrentecon.pdf, mai 2003.

[Cor00] Rational Software Corporation. Using rose. ftp://ftp.software.ibm.com/software/rational/docs/v2002/Rose_usingrose.pdf, 2000.

[D+03] Mehmet Emin Dönderler et al. Bilvideo: a video database management system. *IEEE Multimedia*, 10(1), januar - mars 2003.

[DB03] Mark Drake og Sandeepan Banerjee. Oracle xml db. White paper, Oracle Corporation, http://www.oracle.com, januar 2003.

[DCMI] Dublin Core Metadata Initiative. Dublin core metadata element set. http://dublincore.org.

[DR03] Torgeir Dingsøyr og Emil Røyrvik. An empirical study of an informal knowledge repository in a medium-sized software consulting company. *IEEE*, side 84–92, 2003.

[Dåv03] Ingunn Dåvøy. Arbeidsprosessorientet styring. *Internt skriv, Norsk Hydro A/S*, 2003. For tilgang til dokumentet send epost til davoy@stud.ntnu.no.

[EF] Eclipse Foundation. Eclipse Project. http://www.eclipse.org/.

[Ehr03] Karl Ehrlich. Locating expertise: Design issues for an expertise locator system. I Ackerman et al. [APW03], side 136–158.

[Fow04] Martin Fowler. *UML Distilled*. Pearson Education, Inc., 3 utgave, 2004.

[GP01] Charles F. Goldfarb og Paul Prescod. *The XML Handbook*. The Charles F. Goldfarb Series on Open Information Management. Prentice Hall PTR, third edition utgave, 2001.

[HdW02] Marleen Huysman og Dirk de Wit. *Knowledge Sharing in Practice*, bind 4 av *Information Science and Knowledge Management*. Kluwer Academic Publishers, 2002.

[HdW03] Marleen Huysman og Dirk de Wit. A critical ecaluation of knowledge management practices. I Ackerman et al. [APW03], side 27–55.

[HGL+03] Pål Halvorsen, Carsten Griwodz, Ketil Lund, Vera Goebel og Thomas Plagemann. Storage systems support for multimedia applications. UiO Research Report No. 307, ISBN 82-7368-259-5, ISSN 0806-3036, juni 2003.

[HNT99] Morten T. Hansen, Nitin Nohria og Thomas Tierney. What's your strategy for managing knowledge? *Harvard Business Review*, side 106–117, mars - april 1999.

[Hol04] David Hollingsworth. The workflow reference model: 10 years on. I Layna Fischer, redaktør, *Workflow Handbook*, side 295 – 312. Future Stategies Inc., April 2004.

[HP03] Pamela J. Hinds og Jeffery Pfeffer. Why organizations don't "know what they know": Cognitive and motivational factors affecting the transfer of expartise. I Ackerman et al. [APW03], side 3–26.

[JWG] ISO TC37/SC2-TC46/SC4 Joint Working Group. Codes for representation of names of languages. http://www.loc.gov/standards/iso639-2/.

[KA97] Wolfgang Klas og Karl Aberer. Multimedia applications and their implications on database architectures. I *Multimedia Databases in Perspective*, kapittel 3. Springer Verlag, 1997.

[KWW00] Michael Kozuch, Wayne Wolf og Andrew Wolfe. An experimental analysis of digital video library servers. *Multimedia Systems*, 8(2):135–145, 2000.

[MA] MySql AB. MySql Database Server. http://www.mysql.com/.

[Mau03] Joseph Mauro. Oracle intermedia: Managing multimedia content. White paper, Oracle Corporation, http://www.oracle.com, desember 2003.

[OMG] Object Management Group. Uml resource page. http://www.omg.org/uml/.

[PC03] Bill Penuel og Andrew Cohen. Coming to the crossroads of knowledge, learning, and technology: integrating knowledge management and workplace learning. I Ackerman et al. [APW03], side 57–76.

[RHM02] Elliotte Rusty Harold og W. Scott Means. *XML in a nutshell*. O'Reilly & Associates, Inc, second edition utgave, juni 2002.

[SA] Software AG. Tamino. http://www.tamino.com.

[SGV02] Prashant Sheony, Pawan Goyal og Harrick M. Vin. Architectural considerations for next-generation file system. *Multimedia Systems*, 8(2):270–283, 2002.

[Som01] Ian Sommerville. *Software Engineering*. Addison-Wesley Publishers Limited, sixth edition utgave, 2001.

[SOS] SoluDyne2.05 og Optient Systems. Optient system // connecting intelligence // resellers of soludyne. http://www.optientsystems.com/index2.html, SoluDyne-linken.

[TJP] The JDOM Project. JDOM. http://jdom.org.

[vdV02] Eric van der Vlist. *XML Schema*. O'Reilly & Associates, Inc, first edition utgave, 2002.

[W3C] W3C. World Wide Web Consortium. http://www.w3.org.

[WMC02] Workflow Management Coalition. Xml process definition language. http://www.wfmc.org/standards/docs/TC-1025_10_xpdl_102502.pdf, oktober 2002.

[WW97] Misha Wolf og Charles Wicksteed. Date and time formats. http://www.w3.org/TR/NOTE-datetime, september 1997.

Tillegg A

Metodikk

A.1 Verktøy

Denne rapporten er skrevet i LaTeXved hjelp av teTeX/MikTeX, Ant og Eclipse[EF], som vi også brukte til å utvikle prototypen vår. Eclipse er en fleksibel og utvidbar utviklingsplattform skrevet i Java/C. Vi har jobbet på forskjellig operativsystem, likevel har det plattformuvahengige og fleksible verktøyet latt oss skrive både implementasjon og rapport i samme program, og bruke samme system for versjonskontroll og sikring mot datatap, cvs. Ettersom vi begge har gjort alt vårt arbeid på bærbare datamaskiner fant vi det nødvendig med gode rutiner for dette, og vi har daglig flettet vårt arbeid og synkronisert det med cvs-lagringsplassen på idis unix-system.

A.2 Kritiske elementer

Problemer vi anså som kunne være kritiske for gjennomføringen av oppgaven:

Kommentarer i ettertid: Vi har ikke hatt problemer med sykdom eller annen form for ufrivillig fravær. Vi har heller ikke hatt problemer med datatap eller tap av utstyr/verktøy. Utilgjengelighet av nøkkelpersoner og -informasjon har til tider vært en utfordring, og har påvirket retningen i oppgaven. Vi har ikke jobbet med feil problemstilling, men har heller hatt problemer med å konkretisere den.

A.3 Opprinnelig målsetning

”Prosjektet vårt skal føre frem til en god forståelse av databaser for bruk i multimedia kunnskapsformidlingssystemer gjennom generell tankegang og design/impementasjon for et spesifikt område.

Målsetting 1 er å danne en generell forståelse av ”state of the art” innen elektroniske informasjons- og kunnskapsbehandlings-/formidlingssystemer, med fokus på lagring og databasearkitektur. Denne forståelsen skal konkretiseres i modeller og en rapport som skal være nyttige for oppdragsgiveren også i fremtiden.

Målsetting 2 er å danne en forståelse tilsvarende den i den første målsetningen i en mer domenespesifikk kontekst. Forståelsen gir et grunnlag for å velge beste løsning på de utfordringer en her har. Hensikten er å avgjøre teoretiske krav og suksesskriterier for det systemet som utvikles.

Målsetting 3 er å validere SoluDynes systemer opp mot applikasjonsområdet. Valideringen gir grunnlag for å anbefale videre utvikling av prosjektet. Dette vil også være en validering av våre egne modeller og løsninger. Dette gjøres ved å sette seg inn i de eksisterende løsningene, modellere en del av systemet, og forhåpentligvis implementere dette. Sammen vil det danne et sterkt fundament for vurdering av eksisterende løsning.”

Problem	Indikasjon	Sannsynlighet	Alvorlighet	Tiltak
En av oss må forlate prosjektet	Vil være innlysende	Lav	Høy	Gjenværende deltager må redefinere oppgaven
Sykdom	Sviktende helse, legeerklæring	Middels	Middels eller lav	Utsette innlevering
Utilgjengelighet av nøkkelpersoner og/eller informasjon	Kommunikasjonssvikt	Middels eller lav	Middels eller høy	Bruke andre, revurdere informasjonsbehov
Datatap	Diskhavari, tyveri, brann, data blir utilgjengelig og ugjenopprettelig	Lav	Høy	Preventive tiltak: CVS, versjonskontroll
Jobbe med feil problemstilling	Tilbakemeldinger	Middels	Middels	Forandre oppgaveformulering og/eller ta tak i de riktige oppgavene

Tabell A.1: Analyse av kritiske elementer

Tillegg B

XML

B.1 XML-schema for arbeidsprosessdokument

```
<?xml version="1.0" encoding="UTF-8"?>
<!--W3C XML Schema for arbeidsprosessdokumenter

xs refererer til W3C XML Schema
dc refererer til Dublin Core Metadata Initiative, som er et sett grunnleggende metadatatagger.
-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:dc="http://purl.org/dc/elements/1.1/" elementFormDefault="qualified">
  <xs:import namespace="http://purl.org/dc/elements/1.1/"
    schemaLocation="http://dublincore.org/schemas/xmls/simpledc20021212.xsd"/>

  <!--Et arbeidsprosessdokument består av en dokumentbeskrivelse, en arbeidsbeskrivelse, og en liste med
vedlegg -->
  <xs:element name="arbeidsprosessdokument">
    <xs:complexType>
      <xs:all>
        <!--En dokumentbeskrivelse må inneholde identifikator, tittel og eier (publisher), i tillegg bør den
inneholde punktene nevnt i xs:choice-blokken.-->
        <xs:element name="dokumentbeskrivelse" type="dokumentbeskrivelseType"/>
        <xs:element name="arbeidsbeskrivelse" type="arbeidsbeskrivelseType"/>
        <xs:element name="vedleggsliste" type="vedleggslisteType"/>
      </xs:all>
    </xs:complexType>
    <!-- Denne nøkkelen skal gjøre id-feltene for subprosess, valgpunkt, aktivitet, sluttpunkt og vedlegg unike
og søkbare med keyref-->
    <xs:key name="nøkkel">
      <xs:selector
        xpath="arbeidsbeskrivelse/subprosess|arbeidsbeskrivelse/valgpunkt|arbeidsbeskrivelse/
aktivitet|arbeidsbeskrivelse/slutt|arbeidsbeskrivelse/roller/rolle|vedleggsliste/vedlegg"|"dokumentbeskrivelse/
avhengigheter/avhengighet"/>
      <xs:field xpath="id"/>
    </xs:key>
    <!-- Denne referansen skal sirkre referanseintgriteten mellom id-feltet i subprosess, valgpunkt, aktivitet og
slutt og de respektives går-til og kommer-fra felter.-->
    <xs:keyref name="flytref" refer="nøkkel">
      <xs:selector
        xpath="arbeidsbeskrivelse/start|arbeidsbeskrivelse/subprosess|arbeidsbeskrivelse/
valgpunkt|arbeidsbeskrivelse/aktivitet|arbeidsbeskrivelse/slutt"/>
      <xs:field xpath="ja-gaar-til|nei-gaar-til|gaar-til|kommer-fra"/>
    </xs:keyref>
    <!-- Denne referansen sikrer referanseintegriteten til roller fra hvem felter-->
    <xs:keyref name="rolleref" refer="nøkkel">
      <xs:selector xpath="arbeidsbeskrivelse/start/forutsetninger|arbeidsbeskrivelse/aktivitet"/>
      <xs:field xpath="hvem|ansvar"/>
    </xs:keyref>
    <!-- Denne referansen sikrer referanseintegriteten mellom avhengighetslisten og subprosessene-->
    <xs:keyref name="avhref" refer="nøkkel">
      <xs:selector xpath="arbeidsbeskrivelse/subprosess"/>
      <xs:field xpath="identifikator"/>
    </xs:keyref>
  </xs:element>

  <!-- DokumentbeskrivelseType beskriver dokumentbeskrivelse, en del av arbeidsprosessdokument.-->
  <!-- En dokumentbeskrivelse består av en identifikator samt masse metaiformasjon fra DCMI. Det skal også
finnes en liste med avhengigigheter og endringer.-->
  <xs:complexType name="dokumentbeskrivelseType">
    <xs:sequence>
      <xs:sequence>
        <xs:element name="identifikator" type="identifikatorType"/>
        <xs:element ref="dc:title"/>
        <!-- Dette er eier, temmelig sammenfallende med begrepet publisher: An entity responsible for making
the resource available  -->
        <xs:element ref="dc:publisher"/>
```

```
            <!--Hvilket språk dokumentasjonen er skrevet i. Anbefalt å bruke RFC3066 og ISO639 -->
            <xs:element ref="dc:language"/>
            <!-- Prosessklassen kan for eksempel være "intern" -->
            <xs:element name="prosessklasse" type="xs:token"/>
            <!-- Beskriver hvilket detaljnivå dokumentet er på -->
            <xs:element name="styringsdokumentnivaa" type="xs:token"/>
            <!-- Viser til dato for siste revisjon av arbeidsprosessen -->
            <xs:element ref="dc:date"/>
            <!-- Beskriver hvilke andre arbeidsprosesser denne prosessen er avhengig av. Dette gjøres ved en liste
av identifkatorer. En liste er et sett av null til uendelig elementer separert med "whitespace" -->
            <xs:element name="avhengigheter">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="avhengighet" type="avhengighetType" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <!-- Endringsloggen består av en mengde endringer-->
            <xs:element name="endringslogg">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="endring" type="endringType" maxOccurs="unbounded"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
          <xs:choice minOccurs="0" maxOccurs="unbounded">
            <!-- Dette er en referanse til den eller de som har vært med å lage dokumentet -->
            <xs:element ref="dc:contributor"/>
            <!-- Virkeområder for arbeidsprosessen -->
            <xs:element ref="dc:coverage"/>
          </xs:choice>
        </xs:sequence>
      </xs:complexType>

      <xs:complexType name="avhengighetType">
        <xs:sequence>
          <xs:element name="id" type="xs:ID"/>
          <xs:element name="referanseIdentifikator" type="identifikatorType"/>
        </xs:sequence>
      </xs:complexType>

      <!-- endringsType beskriver en endring i endringsloggen i dokumentbeskrivelsen.-->
      <!-- En endring består av versjonsnummer på endringen, hvem som har gjort endringen, når endringen er gjort
og hva som er gjort-->
      <xs:complexType name="endringType">
        <xs:sequence>
          <xs:element name="versjon" type="xs:token"/>
          <xs:element ref="dc:creator"/>
          <xs:element ref="dc:date"/>
          <xs:element ref="dc:description"/>
        </xs:sequence>
      </xs:complexType>

      <!-- identifikatorTypen beskriver en identifikator. Brukes flere plasser i arbeidsprosessdokumenet. -->
      <!-- Dokumentet identifiseres ut fra kombinasjonen av versjon og identifikasjonsnummeret forbundet med
arbeidsprosessen -->
      <xs:complexType name="identifikatorType">
        <xs:sequence>
          <!-- Dokumentets identifikasjonsnummer vil være konstant for denne arbeidsprosessen.
        Den identifiseres med en xs:ID. Databasen eller aplikasjonslaget må passe på at denne er unik (kan vi her
bruke xLink/xPath for å gjøre dette?)
```

```
      En xs:ID kan _ikke_ starte med et nummer og kan ikke inneholde kolon(":")-->
        <xs:element name="id" type="xs:ID"/>
        <!-- Versjonsnummeret vil endre seg hver gang det gjøres endringer av arbeidsprosessen.
Versjonsnummeret er en tekststreng uten "whitespace"-->
        <xs:element name="versjon" type="xs:token"/>
      </xs:sequence>
    </xs:complexType>

    <!-- Arbeidsbeskrivelsen beskrives med denne typen og er en egen tag under arbeidsprosessdokument -->
    <!-- En arbeidsbeskrivelse bestå av et sett roller, et start punkt og etter det følger ulike aktiviteter,
subprosesser, valgpunkt og sluttpunkt-->
    <xs:complexType name="arbeidsbeskrivelseType">
      <xs:sequence>
        <xs:sequence>
          <!-- I forbindelse med en arbeidsprosess kan det finnes mange roller som beskrives i dette elementet--
>
          <xs:element name="roller" type="rollerType"/>
          <!-- Sørger for at hver rolle har unik id-->
          <xs:element name="start" type="startType"/>
        </xs:sequence>
        <xs:choice minOccurs="0" maxOccurs="unbounded">
          <xs:element name="aktivitet" type="aktivitetType"/>
          <xs:element name="subprosess" type="subprosessType"/>
          <xs:element name="valgpunkt" type="valgpunktType"/>
        </xs:choice>
        <xs:sequence>
          <xs:element name="slutt" type="sluttType" maxoccurs="unbounded"/>
        </xs:sequence>
      </xs:sequence>
    </xs:complexType>


    <!-- Type som definerer en liste som kan inneholde forskjellige roller. Roller hører inn under
arbeidsbeskrivelsen-->
    <xs:complexType name="rollerType">
      <xs:sequence>
        <xs:element name="rolle" type="rolleType" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>


    <!--Beskriver en rolle som refereres til i arbeidsflyten, kunne kobles mot eksisterende begrepsapparat-->
    <xs:complexType name="rolleType">
      <xs:sequence>
        <xs:element name="id" type="xs:ID"/>
        <xs:element ref="dc:title"/>
        <xs:element name="kompetansekrav" type="xs:string"/>
        <xs:element name="vedlegg" type="vedleggsReferanseType" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>


    <!-- Prosessbeskrivelsen beskrives med denne typen. Den koples til start-elementet under arbeidsbeskrivelsen
og skal definere en del konstante verdier vedrørende arbeidsprosessen.-->
    <xs:complexType name="prosessbeskrivelseType">
      <xs:sequence>
        <!-- Formålet med prosessen som helhet. Dette er globalt for hele prosessen og må være tilgjengelig for
subprosesser og aktiviteter -->
        <xs:element name="formaal" type="xs:string"/>
        <xs:element name="maalgruppe" type="xs:string"/>
        <xs:element name="definisjoner" type="definisjonerType"/>
        <xs:element name="ansvar" type="ansvarType"/>
      </xs:sequence>
```

```
  </xs:complexType>

  <!-- En liste med definisjoner som må være klare innad i prosessen-->
  <xs:complexType name="definisjonerType">
    <xs:sequence>
      <xs:element name="definisjon" type="beskrivesmedType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <!-- Ansvaret for en arbeidsprosess kobles til en rolle. For eksempel "Leder feltenhet".
  Denne rollen kan følges av en ekplisitt beskrivelse av hva ansvaret går ut på (i praksis det samme som formål,
men beskriver at han er _ansvarlig_ for at dette skjer gjennom hele prosessen-->
  <xs:complexType name="ansvarType">
    <xs:sequence>
      <xs:element name="hvem" type="referanseType"/>
      <xs:element ref="dc:description" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <!-- Hvoren en befinner seg i arbeidsprosessen skal det være mulig å navigere seg til en subprosess, valgboks
eller aktivitet-->
  <xs:simpleType name="referanseType">
    <xs:restriction base="xs:IDREF"/>
  </xs:simpleType>

  <!-- Typen koples til beskrivelse under aktivitet i en arbeidsbeskrivelse. -->
  <!-- En full beskrivelse av en aktivitet består av et sett mål og krav samt beskrivelse av hva, fordan, hvorfor
ting skal gjøres samt hva som skjer om ting går feil-->
  <xs:complexType name="beskrivelseType">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="maal" type="beskrivesmedType"/>
      <xs:element name="krav" type="beskrivesmedType"/>
      <xs:element name="hva" type="beskrivesmedType"/>
      <xs:element name="hvordan" type="beskrivesmedType"/>
      <xs:element name="hvorfor" type="beskrivesmedType"/>
      <xs:element name="aarsak" type="beskrivesmedType"/>
      <xs:element name="risiko" type="beskrivesmedType"/>
      <xs:element name="konsekvens" type="beskrivesmedType"/>
      <xs:element name="hva-om" type="beskrivesmedType"/>
    </xs:choice>
  </xs:complexType>

  <!-- En del elementer kan beskrives med en enkel tittel og eventuelt en beskrivelse. Her er det mulig å gjøre
restriksjoner eller utvide-->
  <xs:simpleType name="beskrivesmedType" >
    <xs:restriction base="xs:string"/>
  </xs:simpleType>

  <!-- I arbeidsbeskrivelsen skal det være mulig å referere til vedlegg fra de forksjellige elementene. -->
  <!-- vedleggsReferanseTypen består av en vedleggsID og en beskrivelse av hvordan vedlegget skal brukes. -->
  <xs:complexType name="vedleggsReferanseType">
    <xs:sequence>
      <xs:element name="vedleggID" type="xs:IDREF"/>
      <!-- Skal beskrive hvordan programmet skal bruke vedlegget i denne konteksten-->
      <xs:element ref="dc:description"/>
    </xs:sequence>
  </xs:complexType>

  <!--I en arbeidsprosess finnes det et startpunkt. Dette startpunktet inneholder forutsetninger og referanse til
en aktivitet-->
  <xs:complexType name="startType">
    <xs:sequence>
      <xs:element name="id" type="xs:ID"/>
```

```
        <xs:element name="forutsetninger">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="forutsetning" type="xs:string" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="prosessbeskrivelse" type="prosessbeskrivelseType"/>
        <xs:element name="gaar-til" type="referanseType"/>
        <xs:element name="vedleggsref" type="vedleggsReferanseType" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>

    <!-- En aktivitet beskrives med identifikator, tittel, en beskrivelsesblokk, hvem som utfører aktiviteten
    status for aktiviteten samt foregående og etterfølgende aktivitet-->
    <xs:complexType name="aktivitetType">
      <xs:sequence>
        <!-- Identifikator for aktiviteten-->
        <xs:element name="id" type="xs:ID"/>
        <!-- Kort, beskrivende tittel på aktiviteten-->
        <xs:element ref="dc:title"/>
        <!-- En utfyllende beskrivelse av aktiviteten-->
        <xs:element name="beskrivelse" type="beskrivelseType"/>
        <!-- Hvem som utfører aktiviteten. Referanse til en av rollene definert i arbeidsbeskrivelsen-->
        <xs:element name="hvem" type="referanseType"/>
        <!-- Forklarer status til aktiviteten-->
        <xs:element name="status" type="xs:string"/>
        <xs:element name="kommer-fra" type="referanseType" maxOccurs="unbounded"/>
        <xs:element name="gaar-til" type="referanseType"/>
        <!-- Liste med mulige vedlegg-->
        <xs:element name="vedleggsref" type="vedleggsReferanseType" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>

    <!-- En subprosess har sin egen identitet og referanse til forrige og neste punkt i arbeidsprosessen.
    I tillegg har den en identifikator som skal viser til en annen arbeidsprosess i avhengighetslista.-->
    <xs:complexType name="subprosessType">
      <xs:sequence>
        <xs:element name="id" type="xs:ID"/>
        <xs:element name="identifikator" type="xs:IDREF"/>
        <xs:element name="kommer-fra" type="referanseType" maxOccurs="unbounded"/>
        <xs:element name="gaar-til" type="referanseType"/>
        <!-- Liste med mulige vedlegg-->
        <xs:element name="vedleggsref" type="vedleggsReferanseType" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>

    <!-- Et valgpunkt identifiseres med en ID. I tillegg så inneholder det et ja/nei spørsmål og lenker til hvor en
går videre ved de to alternativene-->
    <xs:complexType name="valgpunktType">
      <xs:sequence>
        <xs:element name="id" type="xs:ID"/>
        <xs:element name="spoersmaal" type="xs:string"/>
        <xs:element ref="dc:description"/>
        <xs:element name="kommer-fra" type="referanseType" maxOccurs="unbounded"/>
        <xs:element name="ja-gaar-til" type="referanseType"/>
        <xs:element name="nei-gaar-til" type="referanseType"/>
        <!-- Liste med mulige vedlegg-->
        <xs:element name="vedleggsref" type="vedleggsReferanseType" minOccurs="0"
maxOccurs="unbounded"/>
```

```
    </xs:sequence>
  </xs:complexType>

  <!-- En slutt type består av en ID, beskrivelse av hvor aktiviteten kommer fra og hvilke tilstand systemet er i
etter at prosessen er utført.-->
  <xs:complexType name="sluttType">
    <xs:sequence>
      <xs:element name="id" type="xs:ID"/>
      <xs:element name="kommer-fra" type="referanseType" maxOccurs="unbounded"/>
      <xs:element name="postconditions">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="postcondition" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <!-- Liste med mulige vedlegg-->
      <xs:element name="vedleggsref" type="vedleggsReferanseType" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <!-- Vedleggsliste er en del av arbeidsprosessdokumentet. Beskrives med denne typen. -->
  <!-- Vedleggslisten består av et sett vedlegg. Det kan forekomme at det ikke finens vedlegg.-->
  <xs:complexType name="vedleggslisteType">
    <xs:sequence>
      <xs:element name="vedlegg" type="vedleggType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <!-- Et vedlegg består av en unik ID, en beskrivelse av vedlegget (tittel, beskrivende tekst, laget av , dato)
samt en URI til vedlegget-->
  <xs:complexType name="vedleggType">
    <xs:sequence>
      <xs:element name="id" type="xs:ID"/>
      <xs:element ref="dc:title"/>
      <xs:element ref="dc:description"/>
      <xs:element ref="dc:type"/>
      <xs:element ref="dc:format"/>
      <xs:element ref="dc:creator" minOccurs="0"/>
      <xs:element ref="dc:date" minOccurs="0"/>
      <xs:element name="URI" type="xs:anyURI"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

B.2 XML-dokument for beskrivelse av arbeidsprosess

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

<!-- New document created at Wed Mar 31 14:16:20 CEST 2004 -->
<arbeidsprosessdokument xsi:noNamespaceSchemaLocation="http://localhost/~axel/schema.xsd"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:dc="http://purl.org/dc/elements/1.1/">
      <dokumentbeskrivelse>
            <identifikator>
                  <id>apos3</id>
                  <versjon>2</versjon>
            </identifikator>
            <dc:title>Demontere blinding</dc:title>
            <dc:publisher>EAEogRAW</dc:publisher>
            <dc:language>no</dc:language>
            <prosessklasse>Grunnleggende</prosessklasse>
            <styringsdokumentnivaa>3</styringsdokumentnivaa>
            <dc:date>10062004</dc:date>
            <avhengigheter>
            </avhengigheter>
            <endringslogg>
                  <endring>
                        <versjon>1</versjon>
                        <dc:creator>EAE</dc:creator>
                        <dc:date>10062004</dc:date>
                        <dc:description>Første utgave, basert på Folkes konsept om "Resertifisering av PZV"</
dc:description>
                  </endring>
            </endringslogg>
            <dc:contributor>Erlend Agøy Engum</dc:contributor>
            <dc:contributor>Folke BMB Sørli</dc:contributor>
            <dc:coverage>ppc</dc:coverage>
      </dokumentbeskrivelse>

      <arbeidsbeskrivelse>
            <roller>
                  <rolle>
                        <id>r1</id>
                        <dc:title>Utøvende Fagperson</dc:title>
                        <kompetansekrav>Grunnleggende arbeidsutførsel og sikkerhestutsjekk</
kompetansekrav>
                  </rolle>
            </roller>

            <start>
                  <id>start</id>
                  <forutsetninger>
                        <forutsetning>Reservedeler er hentet</forutsetning>
                        <forutsetning>Verktøy er på plass</forutsetning>
                  </forutsetninger>
                  <prosessbeskrivelse>
                        <formaal>Å ta av blindingen slik at videre arbeid kan utføres</formaal>
                        <maalgruppe></maalgruppe>
                        <definisjoner>
                              <definisjon>Blinding - Lokk som sitter i enden av et rør for å sperre dette</
definisjon>
                              <definisjon>Benzen - Gass</definisjon>
                              <definisjon>LRA - Gass?</definisjon>
                        </definisjoner>
                        <ansvar>
                              <hvem>r1</hvem>
                              <dc:description> Dette må gå riktig for seg og resultatet må bli bra </dc:
description>
                        </ansvar>
                  </prosessbeskrivelse>
```

```
            <gaar-til>a1</gaar-til>
            <vedleggsref>
                <vedleggID>film1</vedleggID>
                <dc:description>Mann som prater om resertifisering av PZV generelt</dc:description>
            </vedleggsref>
        </start>

        <aktivitet>
            <id>a1</id>
            <dc:title>Sjekk for Benzen og LRA</dc:title>
            <beskrivelse>
                <maal>Klargjøre systemets tilstand med hensyn på utslipp av gasser</maal>
                <krav>Bruke sikre metoder</krav>
                <hva>Sjekke for Benxen og LRA</hva>
                <hvordan>1: Sjekk i avstegningsplanen for å se om systemet er HC førende</hvordan>
                <hvordan>2: Sjekk avstegningsplanen for å se om områdetekniker påviste Benzen/LRA
under klargjøring av systemet</hvordan>
                <hvorfor>For å hindre at systemet utsettes for Benzen og LRA</hvorfor>
                <hva-om>Dersom det befinner seg Benzen eller LRA i systemet når blindingen
demonteres utsettes systemet for disse. Dette medfører risiko for miljøet, men lav risiko for personell og drift</
hva-om>
            </beskrivelse>
            <hvem>r1</hvem>
            <status>Middels kritisk</status>
            <kommer-fra>start</kommer-fra>
            <gaar-til>a2</gaar-til>
            <vedleggsref>
                <vedleggID>manual1</vedleggID>
                <dc:description>Dokument med avstengningsplan</dc:description>
            </vedleggsref>
        </aktivitet>

        <aktivitet>
            <id>a2</id>
            <dc:title>Sjekk at trykkløst</dc:title>
            <beskrivelse>
                <maal>Klargjøre systemets tilstand med hensyn på trykk</maal>
                <krav>Bruke sikre metoder</krav>
                <hva>Forsikre at systemet er trykkløst før blinding demonteres</hva>
                <hvordan>1: Sjekk avstegningsplanen for at systemet er kontrollert trykkløst</hvordan>
                <hvordan>2: Ved usikkerhet, kontakt områdetekniker</hvordan>
                <hvorfor>For å hindre arbeid på trykksatt system</hvorfor>
                <hva-om>Man kan få en utblåsning noe som er veldig farlig for hele systemet</hva-om>
            </beskrivelse>
            <hvem>r1</hvem>
            <status>Kritisk</status>
            <kommer-fra>a1</kommer-fra>
            <gaar-til>v1</gaar-til>
        </aktivitet>

        <valgpunkt>
            <id>v1</id>
            <spoersmaal>Er systemet trykkløst?</spoersmaal>
                        <dc:description>Det må ikek være trykk i systemet</dc:description>
            <kommer-fra>a2</kommer-fra>
            <ja-gaar-til>a3</ja-gaar-til>
            <nei-gaar-til>slutt2</nei-gaar-til>
        </valgpunkt>

        <aktivitet>
            <id>a3</id>
            <dc:title>Skru ut bolter/muttere</dc:title>
            <beskrivelse>
```

```
                    <maal>Løsne blindingen</maal>
                    <krav>Systemet er bekreftet trykkløst</krav>
                    <krav>Utførende fagarbeider utfører jobben ved siden av røres og som om røret var
under trykk</krav>
                    <krav>Det skal brukes godkjent gnistsikkert utstyr</krav>
                    <hva>Løsne bolter og muttere</hva>
                    <hvordan>1: Løsne mutter eller bolt lengst vekk fra mekaniker</hvordan>
                    <hvordan>2: Sjekke for gass i systemet</hvordan>
                    <hvordan>3: Vurdere om det er trykk i systemet</hvordan>
                    <hvordan>4: Løsne boltene i en sekvens som fører eventuelle utslipp bort fra
utførende</hvordan>
                    <hvorfor>For å hindre skader på personale, miljø og utstyr</hvorfor>
                    <hva-om>Arbeid på trykksatt system kan føre til utblåsning</hva-om>
                </beskrivelse>
                <hvem>r1</hvem>
                <status>Kritisk</status>
                <kommer-fra>v1</kommer-fra>
                <gaar-til>v2</gaar-til>
                <vedleggsref>
                    <vedleggID>film2</vedleggID>
                    <dc:description>Film som viser korrekt arbeidsposisjon</dc:description>
                </vedleggsref>
                <vedleggsref>
                    <vedleggID>manual2</vedleggID>
                    <dc:description>Dokument som viser prosedyre for avblinding</dc:description>
                </vedleggsref>
                <vedleggsref>
                    <vedleggID>audio1</vedleggID>
                    <dc:description>Intervju med fagansvarlig</dc:description>
                </vedleggsref>
                <vedleggsref>
                    <vedleggID>film3</vedleggID>
                    <dc:description>Film som viser utblåsning</dc:description>
                </vedleggsref>
                <vedleggsref>
                    <vedleggID>film4</vedleggID>
                    <dc:description>Film som viser en flens som skyter ut</dc:description>
                </vedleggsref>
            </aktivitet>

            <valgpunkt>
                <id>v2</id>
                <spoersmaal>Antydninger til gass?</spoersmaal>
                            <dc:description>Det må ikke være antydninger til gass</dc:description>
                <kommer-fra>a3</kommer-fra>
                <ja-gaar-til>a4</ja-gaar-til>
                <nei-gaar-til>a5</nei-gaar-til>
            </valgpunkt>

            <aktivitet>
                <id>a4</id>
                <dc:title>Skru igjen bolter/muttere</dc:title>
                <beskrivelse>
                    <maal>Stenge røret</maal>
                    <krav>Utførende fagarbeider utfører jobben ved siden av røres og som om røret var
under trykk</krav>
                    <krav>Det skal brukes godkjent gnistsikkert utstyr</krav>
                    <hva>Feste bolter og muttere</hva>
                    <hvordan>1: Feste de boltene som er løsnet</hvordan>
                    <hvorfor>Gjenoprette sikkert system</hvorfor>
                    <hva-om>Utslipp av gass</hva-om>
                </beskrivelse>
                <hvem>r1</hvem>
```

```
                <status>Kritisk</status>
                <kommer-fra>v2</kommer-fra>
                <gaar-til>slutt2</gaar-til>
            </aktivitet>

            <aktivitet>
                <id>a5</id>
                <dc:title>Ta av blinding</dc:title>
                <beskrivelse>
                    <maal>Å fjerne blindingen fra enden av røret slik at videre arbeid kan utføres</maal>
                    <krav>Utførende fagarbeider utfører jobben ved siden av røres og som om røret var
under trykk</krav>
                    <krav>Det skal brukes godkjent gnistsikkert utstyr</krav>
                    <hva>Ta av blindingen</hva>
                    <hvordan>1: Demontere bolter/muttere en om gangen, siste bolt/mutter skal alltid være
øverst</hvordan>
                    <hvordan>2: Feste løfteanordning til blindingen</hvordan>
                    <hvordan>3: Fjerne den siste mutteren/bolten og kontrollert ta vekk blindingen</
hvordan>
                    <hvorfor>Unngå skader påført ved ukontrollert bevegelse fra blindingen</hvorfor>
                    <hva-om>Ved freil boltsekvens kan klemskader forekomme</hva-om>
                </beskrivelse>
                <hvem>r1</hvem>
                <status>Middels kritisk</status>
                <kommer-fra>v2</kommer-fra>
                <gaar-til>slutt1</gaar-til>
                <vedleggsref>
                    <vedleggID>film5</vedleggID>
                    <dc:description>Film som viser pendeleffekten ved feil boltsekvens</dc:description>
                </vedleggsref>
            </aktivitet>

            <slutt>
                <id>slutt1</id>
                <kommer-fra>a5</kommer-fra>
                <postconditions>
                    <postcondition>Blinding demontert</postcondition>
                </postconditions>
            </slutt>

            <slutt>
                <id>slutt2</id>
                <kommer-fra>v1</kommer-fra>
                <kommer-fra>a4</kommer-fra>
                <postconditions>
                    <postcondition>Systemet er i en usikker tilstand</postcondition>
                    <postcondition>Driftsoperatør må kontaktes</postcondition>
                </postconditions>
            </slutt>
        </arbeidsbeskrivelse>
        <vedleggsliste>
            <vedlegg>
                <id>film1</id>
                <dc:title>Resertifisering av PZV</dc:title>
                <dc:description>Film som viser sammendrag av Resertifisering av PZV</dc:description>
                <dc:type>Film</dc:type>
                <dc:format>mpeg4</dc:format>
                <dc:creator>FS</dc:creator>
                <dc:date>10062004</dc:date>
                <URI>anyURI</URI>
            </vedlegg>

            <vedlegg>
```

```
            <id>film2</id>
            <dc:title>Korrekt arbeidsposisjon</dc:title>
            <dc:description>Film som viser korrekt arbeidsposisjon ved demontering av blinding</dc:
description>
            <dc:type>Film</dc:type>
            <dc:format>mpeg4</dc:format>
            <dc:creator>FS</dc:creator>
            <dc:date>10062004</dc:date>
            <URI>anyURI</URI>
        </vedlegg>

        <vedlegg>
            <id>film3</id>
            <dc:title>Utblåsning</dc:title>
            <dc:description>Film som viser utblåsning ved demontering av blinding</dc:description>
            <dc:type>Animasjon</dc:type>
            <dc:format>flash</dc:format>
            <dc:creator>FS</dc:creator>
            <dc:date>10062004</dc:date>
            <URI>anyURI</URI>
        </vedlegg>

        <vedlegg>
            <id>film4</id>
            <dc:title>Flens</dc:title>
            <dc:description>Film som viser en flens som skyter ut</dc:description>
            <dc:type>Animasjon</dc:type>
            <dc:format>flash</dc:format>
            <dc:creator>FS</dc:creator>
            <dc:date>10062004</dc:date>
            <URI>anyURI</URI>
        </vedlegg>

        <vedlegg>
            <id>film5</id>
            <dc:title>Feil boltsekvens</dc:title>
            <dc:description>Film som viser pendeleffekten ved feil boltsekvens</dc:description>
            <dc:type>Animasjon</dc:type>
            <dc:format>flash</dc:format>
            <dc:creator>FS</dc:creator>
            <dc:date>10062004</dc:date>
            <URI>anyURI</URI>
        </vedlegg>

        <vedlegg>
            <id>manual1</id>
            <dc:title>Korrekt arbeidsposisjon</dc:title>
            <dc:description>Dokumentet avstengningsplan</dc:description>
            <dc:type>Tekst</dc:type>
            <dc:format>pdf</dc:format>
            <dc:creator>FS</dc:creator>
            <dc:date>10062004</dc:date>
            <URI>anyURI</URI>
        </vedlegg>

        <vedlegg>
            <id>manual2</id>
            <dc:title>Sektorprosedyre</dc:title>
            <dc:description>Prosedyrer for avblinding av flenser</dc:description>
            <dc:type>Tekst</dc:type>
            <dc:format>pdf</dc:format>
            <dc:creator>FS</dc:creator>
            <dc:date>10062004</dc:date>
```

```
            <URI>anyURI</URI>
        </vedlegg>

        <vedlegg>
            <id>audio1</id>
            <dc:title>Fagansvarlig</dc:title>
            <dc:description>Intervju med fagansvarlig</dc:description>
            <dc:type>Lyd</dc:type>
            <dc:format>mp3</dc:format>
            <dc:creator>FS</dc:creator>
            <dc:date>10062004</dc:date>
            <URI>anyURI</URI>
        </vedlegg>
    </vedleggsliste>
</arbeidsprosessdokument>
```

B.3 XSLT for transformasjon fra xml til html

```
<?xml version="1.0"?>

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:dc="http://purl.org/dc/elements/
1.1/"
	version="1.0">
	<xsl:output method="html" indent="yes" encoding="UTF-8"/>
	<xsl:template match="/">
		<xsl:apply-templates/>
	</xsl:template>
	<!-- Malen for arbeidsprosessdokument legger til hode og kropp til dokumentet. Tittelen blir bestående av
tittel, dokument ID og versjon.
		Deretter legges dokumentbeskrivelsen til dokumentet. Videre følger grunnleggende informasjon
som ligger i start-noden.
		-->
	<xsl:template match="arbeidsprosessdokument">
		<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
			<head>
				<title> <xsl:value-of select="dokumentbeskrivelse/dc:title"/>; <xsl:apply-templates
					select="dokumentbeskrivelse/identifikator"/> </title>
			</head>
			<body>
				<xsl:apply-templates select="dokumentbeskrivelse"/>
				<xsl:apply-templates select="arbeidsbeskrivelse"/>
				<xsl:apply-templates select="vedleggsliste"/>
			</body>
		</html>
	</xsl:template>
	<!-- Malen for dokumentbeskrivelse legger til informasjon om hvem som eier dokumentet og hvilke klasse
det tilhører.
		Deretter vises tittelen, identifikatorinformasjon, siste revisjonsdato og gyldighetsområde. Til slutt
legges det til
		informasjon angående hvem som har utarbeidet og godkjent arbeidsprosessen.-->
	<xsl:template match="dokumentbeskrivelse">
		<p>
			<xsl:value-of select="dc:publisher"/>
			<br/>
			<xsl:value-of select="prosessklasse"/>
			<br/>
		</p>
		<hr/>
		<p> Tittel: <b> <xsl:value-of select="dc:title"/> </b><br/> <xsl:apply-templates
select="identifikator"/><br/>
			Revisjonsdato: <xsl:value-of select="dc:date"/><br/> Gyldig for område: <xsl:value-of
			select="dc:coverage"/><br/> </p>
		<hr/>
		<p> Utarbeidet av: <xsl:value-of select="dc:contributor"/><br/> Godkjent av: <xsl:value-of
			select="dc:publisher"/><br/> </p>
		<hr/>
	</xsl:template>
	<!-- Malen identifikator sier at en skriver ut Dokument ID og Versjon.-->
	<xsl:template match="identifikator"> Dokument ID: <xsl:value-of select="id"/>, Versjon: <xsl:value-of
		select="versjon"/> </xsl:template>
	<!-- Malen arbeidsbeskrivelse skriver først ut generell informasjon om hele prosessen...-->
	<xsl:template match="arbeidsbeskrivelse">
		<xsl:apply-templates select="start"/>
	</xsl:template>
	<!-- Malen start definerer første arbeidsoppgave-->
	<xsl:template match="start">
		<xsl:apply-templates select="prosessbeskrivelse"/>
		<xsl:apply-templates select="forutsetninger"/>
		<table border="1">
			<tr>
				<th>
```

```
                        <!-- <hr/> -->
                        <h1>STARTER ARBEIDSPROSESSEN</h1>
                        <xsl:if test="count(vedlegg)>0">
                            <b>Vedlegg:</b>
                            <br/>
                            <xsl:for-each select="vedlegg"> <!-- Skal skrive ut <a
href="#id">Vedleggstittel</a>-->
                                <xsl:text disable-output-escaping="yes">&lt;a href&#61;&quot;#</xsl:
text> <xsl:value-of
                                select="id"/> <xsl:text disable-output-escaping="yes">&quot;&gt;</
xsl:text>
                                <xsl:apply-templates select="../../../../vedleggsliste/vedlegg"> <xsl:
with-param name="vID"
                                select="id"/> </xsl:apply-templates> <xsl:text
                                disable-output-escaping="yes">&lt;/a&gt;</xsl:text><br/>
Beskrivelse: <xsl:value-of
                                select="dc:description"/> </xsl:for-each>
                        </xsl:if>
                    </th>
                </tr>
                <xsl:apply-templates select="../aktivitet|../valgpunkt|../subprosess|../slutt">
                    <xsl:with-param name="referanse" select="normalize-space(gaar-til)"/>
                    <xsl:with-param name="kf" select="'start'"/>
                </xsl:apply-templates>
            </table>
        </xsl:template>
        <!-- Malen for Valgpunkt skriver ut valgpunkt og spørsmål før den genererer de to ulike stiene videre-->
        <xsl:template match="valgpunkt">
            <xsl:param name="referanse" select="'null'"/>
            <!--Parameter med referanse til denne prosessen-->
            <xsl:param name="altref" select="'null'"/>
            <xsl:param name="kf" select="'null'"/>
            <!-- Parameter som inneholder referanse til elementet arbeidsflyten kom fra-->
            <xsl:param name="altkf" select="'null'"/>
            <xsl:param name="knr" select="1"/>
            <xsl:choose>
                <xsl:when test="$referanse=normalize-space(id)">
                    <xsl:choose>
                        <xsl:when test="kommer-fra[position()=1]=$kf or kommer-
fra[position()=1]=$altkf">
                            <tr>
                                <th>
                                    <!-- <hr/> -->
                                    <!--referanse: <xsl:value-of select="$referanse"/><br/>
                                altref: <xsl:value-of select="$altref"/><br/>
                                kf: <xsl:value-of select="$kf"/><br/>
                                altkf: <xsl:value-of select="$altkf"/><br/>
                                knr: <xsl:value-of select="$knr"/><br/>       -->
                                    <h2> VALGPUNKT <xsl:value-of select="id"/></h2>
                                    <xsl:if test="count(vedlegg)>0">
                                        <b>Vedlegg:</b>
                                        <br/>
                                        <xsl:for-each select="vedlegg"> <!-- Skal skrive ut <a
href="#id">Vedleggstittel</a>-->
                                            <xsl:text disable-output-escaping="yes">&lt;a
href&#61;&quot;#</xsl:text>
                                            <xsl:value-of select="id"/> <xsl:text
                                            disable-output-escaping="yes">&quot;&gt;</xsl:text>
<xsl:apply-templates
                                            select="../../../vedleggsliste/vedlegg"> <xsl:with-
param name="vID"
                                            select="id"/> </xsl:apply-templates> <xsl:text
                                            disable-output-escaping="yes">&lt;/a&gt;</xsl:
```

```
text><br/> Beskrivelse:
                                                <xsl:value-of select="dc:description"/> </xsl:for-
each>
                                        </xsl:if>
                                    </th>
                                </tr>
                                <xsl:if test="$altref=normalize-space(id)">
                                    <tr>
                                        <th> Her møtes trådene fra: <xsl:for-each select="kommer-fra">
<i><xsl:value-of
                                                select="."/> </i> </xsl:for-each> </th>
                                    </tr>
                                </xsl:if>
                                <xsl:if test="$knr=1">
                                    <xsl:text disable-output-escaping="yes">&lt;tr&gt;</xsl:text>
                                </xsl:if>
                                <!-- Kolonne 1, header-->
                                <th> <!-- <hr/> --> <h3>Ved svar JA</h3> På spørsmål: <xsl:value-of
select="spoersmaal"/> </th>
                                <!-- Kolonne 2, header-->
                                <th> <!-- <hr/> --> <h3>Ved svar NEI</h3> På spørsmål: <xsl:value-of
select="spoersmaal"/> </th>
                                <xsl:if test="$knr=1">
                                    <xsl:text disable-output-escaping="yes">&lt;/tr&gt;</xsl:text>
                                </xsl:if>
                                <!-- Dersom dette er første VP eller sammenslåing av tråder: få laget
rader(knr=1) og kolonner-->
                                <xsl:choose>
                                    <xsl:when test="$altref='null' or $altref=normalize-space(id)">
                                        <!-- Når det ikke er gitt alternativ referanse finnes kun en tråd-->
                                        <!-- Dersom alternativ referanse er "meg selv" så er det ikke
lengre vits i å ha to tråder. Dette sies her fra om.-->
                                        <xsl:apply-templates select="../aktivitet|../valgpunkt|../
subprosess|../slutt">
                                            <xsl:with-param name="referanse" select="normalize-
space(ja-gaar-til)"/>
                                            <xsl:with-param name="kf" select="normalize-space(id)"/>
                                            <xsl:with-param name="altkf" select="normalize-
space(id)"/>
                                            <xsl:with-param name="altref" select="normalize-
space(nei-gaar-til)"/>
                                            <xsl:with-param name="knr" select="1"/>
                                        </xsl:apply-templates>
                                    </xsl:when>
                                    <xsl:when test="$altkf!='null'">
                                        <!-- Når alternativ referanse er deg selv møtes trådene og en tråd
fortsetter-->
                                        <xsl:apply-templates select="../aktivitet|../valgpunkt|../
subprosess|../slutt">
                                            <xsl:with-param name="referanse" select="$altref"/>
                                            <xsl:with-param name="altref" select="normalize-
space(nei-gaar-til)"/>
                                            <xsl:with-param name="kf" select="$altkf"/>
                                            <xsl:with-param name="altkf" select="normalize-
space(id)"/>
                                        </xsl:apply-templates>
                                    </xsl:when>
                                    <xsl:otherwise>
                                        <!-- Ellers, det vil si at det finnes en annen alternativ referanse,
må det hoppes fram og tilbake.-->
                                        <xsl:apply-templates select="../aktivitet|../valgpunkt|../
subprosess|../slutt">
                                            <xsl:with-param name="referanse" select="$altref"/>
```

```
                                                <xsl:with-param name="altref" select="normalize-
space(nei-gaar-til)"/>
                                                <xsl:with-param name="kf" select="$kf"/>
                                                <xsl:with-param name="altkf" select="normalize-
space(id)"/>
                                                </xsl:apply-templates>
                                        </xsl:otherwise>
                                </xsl:choose>
                        </xsl:when>
                        <xsl:otherwise>
                                <xsl:if test="$knr=1">
                                        <xsl:text disable-output-escaping="yes">&lt;tr&gt;</xsl:text>
                                </xsl:if>
                                <td> <!-- <hr/> --> Se videre gang etter spørsmål <i><xsl:value-of
select="spoersmaal"/></i> en
                                        annen plass i tråden. </td>
                                <xsl:if test="$knr=1">
                                        <xsl:text disable-output-escaping="yes">&lt;/tr&gt;</xsl:text>
                                </xsl:if>
                        </xsl:otherwise>
                </xsl:choose>
        </xsl:when>
        <xsl:otherwise/>
    </xsl:choose>
</xsl:template>
<xsl:template match="rolle">
    <xsl:param name="rolleID" select="r-1"/>
    <xsl:choose>
        <xsl:when test="$rolleID=normalize-space(id)">
            <xsl:value-of select="dc:title"/>
            <br/>
        </xsl:when>
        <xsl:otherwise/>
    </xsl:choose>
</xsl:template>
<!-- Malen for en aktivitet skriver ut AKTIVITET, tittel roller, staturs og beskrivelse og sender stafettpinnen
til
    neste punkt i planen-->
<xsl:template match="aktivitet">
    <xsl:param name="referanse" select="'null'"/>
    <!--Parameter med referanse til denne prosessen-->
    <xsl:param name="altref" select="'null'"/>
    <!-- Alternativ referanse som brukes når det finnes en valgboks og en må hoppe mellom to tråder-->
    <xsl:param name="kf" select="'null'"/>
    <!-- Parameter som inneholder referanse til elementet arbeidsflyten kom fra-->
    <xsl:param name="altkf" select="'null'"/>
    <!-- Alternativ kommer-fra referanse som brukes når en hopper mellom to tråder-->
    <xsl:param name="knr" select="1"/>
    <!-- Hvilken kolonne en skal skrive til, dersom den står som 1 skrives ny rad, som 2 skrives kun
kolonne -->
    <xsl:choose>
        <xsl:when test="$referanse=normalize-space(id)">
            <xsl:choose>
                <xsl:when test="kommer-fra[position()=1]=$kf or kommer-
fra[position()=1]=$altkf">
                        <xsl:if test="$knr=1">
                            <xsl:text disable-output-escaping="yes">&lt;tr&gt;</xsl:text>
                            <!-- setter inn <tr>-->
                        </xsl:if>
                        <td>
                            <!-- <hr/> -->
                            <!--referanse: <xsl:value-of select="$referanse"/><br/>
                altref: <xsl:value-of select="$altref"/><br/>
```

```
                    kf: <xsl:value-of select="$kf"/><br/>
                    altkf: <xsl:value-of select="$altkf"/><br/>
                    knr: <xsl:value-of select="$knr"/><br/>-->
                                <p>
                                    <h2>AKTIVITET <xsl:value-of select="id"/></h2>
                                </p>
                                <p> <b>Tittel:</b> <xsl:value-of select="dc:title"/><br/>
<b>Utføres av:</b>
                                    <xsl:apply-templates select="../roller/rolle"> <xsl:with-param
name="rolleID"
                                    select="hvem"/> </xsl:apply-templates> <b>Status:</b> <xsl:
value-of
                                    select="status"/><br/> Går til aktivitet med id: <xsl:value-of
select="gaar-til"/><br/>
                                    <xsl:apply-templates select="beskrivelse"/><br/> <xsl:if
test="count(vedlegg)>0">
                                    <b>Vedlegg:</b><br/> <xsl:for-each select="vedlegg"> <!--
Skal skrive ut <a href="#id">Vedleggstittel</a>-->
                                     <xsl:text disable-output-escaping="yes">&lt;a href&#61;&quot;
#</xsl:text>
                                    <xsl:value-of select="id"/> <xsl:text
                                    disable-output-escaping="yes">&quot;&gt;</xsl:text> <xsl:
apply-templates
                                    select="../../../vedleggsliste/vedlegg"> <xsl:with-param
name="vID" select="id"/>
                                    </xsl:apply-templates> <xsl:text
                                    disable-output-escaping="yes">&lt;/a&gt;</xsl:text><br/>
Beskrivelse: <xsl:value-of
                                    select="dc:description"/> </xsl:for-each> </xsl:if> </p>
                            </td>
                            <xsl:if test="$knr=1  and ($altref='null' or $altref=normalize-space(id))">
                                <xsl:text disable-output-escaping="yes">&lt;/tr&gt;</xsl:text>
                            </xsl:if>
                            <xsl:if test="$knr>1">
                                <xsl:text disable-output-escaping="yes">&lt;/tr&gt;</xsl:text>
                            </xsl:if>
                            <xsl:choose>
                                <!-- Når det ikke er gitt alternativ referanse finnes kun en tråd eller
trådene samles-->
                                <xsl:when test="$altref='null' or $altref=normalize-space(id)">
                                    <xsl:if test="$altref=normalize-space(id)"> Her møtes trådene:
<xsl:for-each
                                        select="kommer-fra"> <i><xsl:value-of select="."/></i>
</xsl:for-each> </xsl:if>
                                    <xsl:apply-templates select="../aktivitet|../valgpunkt|../
subprosess|../slutt">
                                        <xsl:with-param name="referanse" select="normalize-
space(gaar-til)"/>
                                        <xsl:with-param name="kf" select="normalize-space(id)"/>
                                    </xsl:apply-templates>
                                </xsl:when>
                                <!-- Når alternativ referanse finnes og vi er i kolonne 1...-->
                                <xsl:when test="$altref!='null' and $knr=1">
                                    <xsl:apply-templates select="../aktivitet|../valgpunkt|../
subprosess|../slutt">
                                        <xsl:with-param name="referanse" select="$altref"/>
                                        <xsl:with-param name="altref" select="normalize-
space(gaar-til)"/>
                                        <xsl:with-param name="kf" select="$altkf"/>
                                        <xsl:with-param name="altkf" select="normalize-
space(id)"/>
                                        <xsl:with-param name="knr" select="$knr+1"/>
                                    </xsl:apply-templates>
```

```
                                        </xsl:when>
                                        <xsl:when test="$altref!='null' and $knr>1">
                                                <!-- Mulig? trolig ikke-->
                                                <xsl:apply-templates select="../aktivitet|../valgpunkt|../
subprosess|../slutt">
                                                        <xsl:with-param name="referanse" select="$altref"/>
                                                        <xsl:with-param name="altref" select="normalize-
space(gaar-til)"/>
                                                        <xsl:with-param name="kf" select="$altkf"/>
                                                        <xsl:with-param name="altkf" select="normalize-
space(id)"/>
                                                        <xsl:with-param name="knr" select="1"/>
                                                </xsl:apply-templates>
                                        </xsl:when>
                                        <!-- Ellers, det vil si at det finnes en annen alternativ referanse, må det
hoppes fram og tilbake.-->
                                        <xsl:otherwise>
                                                <xsl:apply-templates select="../aktivitet|../valgpunkt|../
subprosess|../slutt">
                                                        <xsl:with-param name="referanse" select="$altref"/>
                                                        <xsl:with-param name="altref" select="normalize-
space(gaar-til)"/>
                                                        <xsl:with-param name="kf" select="$kf"/>
                                                        <xsl:with-param name="altkf" select="normalize-
space(id)"/>
                                                </xsl:apply-templates>
                                        </xsl:otherwise>
                                </xsl:choose>
                        </xsl:when>
                        <xsl:otherwise>
                                <xsl:if test="$knr=1">
                                        <xsl:text disable-output-escaping="yes">&lt;tr&gt;</xsl:text>
                                </xsl:if>
                                <td> Aktiviteten <i><xsl:value-of select="id"/>; <xsl:value-of select="dc:
title"/></i> finnes
                                        et annet sted i tråden </td>
                                <xsl:if test="$knr=1  and $altref='null'">
                                        <xsl:text disable-output-escaping="yes">&lt;/tr&gt;</xsl:text>
                                </xsl:if>
                                <xsl:if test="$knr>1">
                                        <xsl:text disable-output-escaping="yes">&lt;/tr&gt;</xsl:text>
                                </xsl:if>
                                <xsl:apply-templates select="../aktivitet|../valgpunkt|../subprosess|../slutt">
                                        <xsl:with-param name="referanse" select="$altref"/>
                                        <xsl:with-param name="kf" select="$altkf"/>
                                </xsl:apply-templates>
                        </xsl:otherwise>
                </xsl:choose>
        </xsl:when>
        <xsl:otherwise/>
    </xsl:choose>
</xsl:template>
<!-- Malen for en beskrivelse av en aktivitet-->
<xsl:template match="beskrivelse">
    <p>
        <b> Beskrivelse </b>
        <br/>
        <b>Mål:</b>
        <xsl:apply-templates select="maal"/>
        <b>Krav:</b>
        <xsl:apply-templates select="krav"/>
        <b>Hva:</b>
        <xsl:apply-templates select="hva"/>
```

```
                <b>Hvordan:</b>
                <xsl:apply-templates select="hvordan"/>
                <b>Hvorfor:</b>
                <xsl:apply-templates select="hvorfor"/>
                <b>Hva om:</b>
                <xsl:apply-templates select="hva-om"/>
            </p>
        </xsl:template>
        <xsl:template match="maal|krav|hva|hvordan|hvorfor|hva-om|postcondition">
            <xsl:value-of select="."/>
            <br/>
        </xsl:template>
        <!-- Malen skriver SUBPROSESS, viser identifikatoren til ekstern prosess (bør her linke, XLink?) og
            sender stafettpinnen videre-->
        <xsl:template match="subprosess">
            <xsl:param name="referanse" select="'null'"/>
            <!--Parameter med referanse til denne prosessen-->
            <xsl:param name="kf" select="'null'"/>
            <!-- Parameter som inneholder referanse til elementet arbeidsflyten kom fra-->
            <xsl:param name="altref" select="'null'"/>
            <xsl:param name="altkf" select="'null'"/>
            <xsl:param name="knr" select="1"/>
            <xsl:choose>
                <xsl:when test="$referanse=normalize-space(id)">
                    <xsl:choose>
                        <xsl:when test="kommer-fra[position()=1]=$kf or kommer-
fra[position()=1]=$altkf">
                            <xsl:if test="$knr=1">
                                <xsl:text disable-output-escaping="yes">&lt;/tr&gt;</xsl:text>
                            </xsl:if>
                            <td>
                                <!-- <hr/> -->
                                <p>
                                    <h2>SUBPROSESS <xsl:value-of select="id"/></h2>
                                </p>
                                <b>Linkes til:</b>
                                <xsl:apply-templates select="identifikator"/>
                                <br/>
                                <xsl:if test="count(vedlegg)>0">
                                    <b>Vedlegg:</b>
                                    <br/>
                                    <xsl:for-each select="vedlegg"> <!-- Skal skrive ut <a
href="#id">Vedleggstittel</a>-->
                                        <xsl:text disable-output-escaping="yes">&lt;a href&#61;
&quot;#</xsl:text>
                                        <xsl:value-of select="id"/> <xsl:text
                                        disable-output-escaping="yes">&quot;&gt;</xsl:text> <xsl:
apply-templates
                                        select="../../../vedleggsliste/vedlegg"> <xsl:with-param
name="vID" select="id"/>
                                        </xsl:apply-templates> <xsl:text
                                        disable-output-escaping="yes">&lt;/a&gt;</xsl:text><br/>
Beskrivelse: <xsl:value-of
                                        select="dc:description"/> </xsl:for-each>
                                </xsl:if>
                            </td>
                            <xsl:if test="$knr=1  and $altref='null'">
                                <xsl:text disable-output-escaping="yes">&lt;/tr&gt;</xsl:text>
                            </xsl:if>
                            <xsl:if test="$knr>1">
                                <xsl:text disable-output-escaping="yes">&lt;/tr&gt;</xsl:text>
                            </xsl:if>
                            <xsl:choose>
```

```
                                <!-- Når det ikke er gitt alternativ referanse finnes kun en tråd eller
trådene samles-->
                                <xsl:when test="$altref='null' or $altref=normalize-space(id)">
                                        <xsl:if test="$altref=normalize-space(id)"> Her møtes trådene:
<xsl:for-each
                                                select="kommer-fra"> <i><xsl:value-of select="."/></i>
</xsl:for-each> </xsl:if>
                                        <xsl:apply-templates select="../aktivitet|../valgpunkt|../
subprosess|../slutt">
                                                <xsl:with-param name="referanse" select="normalize-
space(gaar-til)"/>
                                                <xsl:with-param name="kf" select="normalize-space(id)"/>
                                        </xsl:apply-templates>
                                </xsl:when>
                                <!-- Når alternativ referanse finnes og vi er i kolonne 1...-->
                                <xsl:when test="$altref!='null' and $knr=1">
                                        <xsl:apply-templates select="../aktivitet|../valgpunkt|../
subprosess|../slutt">
                                                <xsl:with-param name="referanse" select="$altref"/>
                                                <xsl:with-param name="altref" select="normalize-
space(gaar-til)"/>
                                                <xsl:with-param name="kf" select="$altkf"/>
                                                <xsl:with-param name="altkf" select="normalize-
space(id)"/>
                                                <xsl:with-param name="knr" select="$knr+1"/>
                                        </xsl:apply-templates>
                                </xsl:when>
                                <xsl:when test="$altref!='null' and $knr>1">
                                        <!-- Mulig? trolig ikke-->
                                        <xsl:apply-templates select="../aktivitet|../valgpunkt|../
subprosess|../slutt">
                                                <xsl:with-param name="referanse" select="$altref"/>
                                                <xsl:with-param name="altref" select="normalize-
space(gaar-til)"/>
                                                <xsl:with-param name="kf" select="$altkf"/>
                                                <xsl:with-param name="altkf" select="normalize-
space(id)"/>
                                                <xsl:with-param name="knr" select="1"/>
                                        </xsl:apply-templates>
                                </xsl:when>
                                <!-- Ellers, det vil si at det finnes en annen alternativ referanse, må det
hoppes fram og tilbake.-->
                                <xsl:otherwise>
                                        <xsl:apply-templates select="../aktivitet|../valgpunkt|../
subprosess|../slutt">
                                                <xsl:with-param name="referanse" select="$altref"/>
                                                <xsl:with-param name="altref" select="normalize-
space(gaar-til)"/>
                                                <xsl:with-param name="kf" select="$kf"/>
                                                <xsl:with-param name="altkf" select="normalize-
space(id)"/>
                                        </xsl:apply-templates>
                                </xsl:otherwise>
                        </xsl:choose>
                </xsl:when>
                <xsl:otherwise>
                        <xsl:if test="$knr=1">
                                <xsl:text disable-output-escaping="yes">&lt;tr&gt;</xsl:text>
                        </xsl:if>
                        <td> Subprosessen <i><xsl:value-of select="id"/>; <xsl:value-of
select="dc:title"/></i> finnes
                                et annet sted i tråden </td>
                        <xsl:if test="$knr=1  and $altref='null'">
```

```
                                        <xsl:text disable-output-escaping="yes">&lt;/tr&gt;</xsl:text>
                                    </xsl:if>
                                    <xsl:if test="$knr>1">
                                        <xsl:text disable-output-escaping="yes">&lt;/tr&gt;</xsl:text>
                                    </xsl:if>
                                    <xsl:apply-templates select="../aktivitet|../valgpunkt|../subprosess|../slutt">
                                        <xsl:with-param name="referanse" select="$altref"/>
                                        <xsl:with-param name="kf" select="$altkf"/>
                                    </xsl:apply-templates>
                                </xsl:otherwise>
                            </xsl:choose>
                        </xsl:when>
                        <xsl:otherwise/>
                    </xsl:choose>
                </xsl:template>
                <!-- Malen skriver SLUTT og lister opp de eksisterende postconditions-->
                <xsl:template match="slutt">
                    <xsl:param name="referanse" select="'null'"/>
                    <!--Parameter med referanse til denne prosessen-->
                    <xsl:param name="kf" select="'null'"/>
                    <!-- Parameter som inneholder referanse til elementet arbeidsflyten kom fra-->
                    <xsl:param name="altref" select="'null'"/>
                    <xsl:param name="altkf" select="'null'"/>
                    <xsl:choose>
                        <xsl:when test="$referanse=normalize-space(id)">
                            <tr>
                                <td>
                                    <!-- <hr/> -->
                                    <xsl:choose>
                                        <xsl:when test="kommer-fra[position()=1]=$kf or kommer-
fra[position()=1]=$altkf">
                                            <p>
                                                <h2>SLUTT</h2>
                                            </p>
                                            <b>Tilstanden i systemet:</b>
                                            <br/>
                                            <xsl:apply-templates select="postconditions/postcondition"/>
                                            <br/>
                                            <xsl:if test="count(vedlegg)>0">
                                                <b>Vedlegg:</b>
                                                <br/>
                                                <xsl:for-each select="vedlegg"> <!-- Skal skrive ut <a
href="#id">Vedleggstittel</a>-->
                                                    <xsl:text disable-output-escaping="yes">&lt;a
href&#61;&quot;#</xsl:text>
                                                    <xsl:value-of select="id"/> <xsl:text
                                                    disable-output-escaping="yes">&quot;&gt;</xsl:text>
<xsl:apply-templates
                                                    select="../../../vedleggsliste/vedlegg"> <xsl:with-
param name="vID"
                                                    select="id"/> </xsl:apply-templates> <xsl:text
                                                    disable-output-escaping="yes">&lt;/a&gt;</xsl:
text><br/> Beskrivelse:
                                                    <xsl:value-of select="dc:description"/> </xsl:for-
each>
                                            </xsl:if>
                                        </xsl:when>
                                        <xsl:otherwise> Slutten <i><xsl:value-of select="id"/>; <xsl:value-of
                                            select="dc:title"/></i> finnes et annet sted i tråden </xsl:
otherwise>
                                    </xsl:choose>
                                </td>
                            </tr>
```

```
                </xsl:when>
                <xsl:otherwise/>
            </xsl:choose>
        </xsl:template>
        <!-- Mal for utskrift av forutsetninger-->
        <xsl:template match="forutsetninger">
            <p>
                <p> 5: FORUTSETNINGER </p>
                <xsl:apply-templates select="forutsetning"/>
            </p>
        </xsl:template>
        <!-- Mal for utskrift av forutsetning-->
        <xsl:template match="forutsetning">
            <p>
                <xsl:value-of select="."/>
            </p>
        </xsl:template>
        <!-- Malen prosessbeksrivelse skriver ut arbeidsprosessens formål, målgruppe, definisjoner (flere av disse
kan skives) og ansvar-->
        <xsl:template match="prosessbeskrivelse">
            <p>
                <p> 1: FORMÅL </p>
                <xsl:value-of select="formaal"/>
            </p>
            <p>
                <p> 2: MÅLGRUPPE </p>
                <xsl:value-of select="maalgruppe"/>
            </p>
            <p>
                <p> 3: DEFINISJONER </p>
                <xsl:apply-templates select="definisjoner/definisjon"/>
            </p>
            <p>
                <p> 4: ANSVAR </p>
                <xsl:apply-templates select="ansvar"/>
            </p>
        </xsl:template>
        <!-- Malen definisjon putter inn en definisjon i et avsnitt-->
        <xsl:template match="definisjon">
            <p>
                <xsl:value-of select="beskrivelse"/>
            </p>
        </xsl:template>
        <!-- Malen skriver ut noe om ansvar...-->
        <xsl:template match="ansvar">
            <p>
                <xsl:choose>
                    <xsl:when
                        test="normalize-space(//arbeidsprosessdokument/arbeidsbeskrivelse/roller/rolle/
id)=normalize-space(hvem)">
                         Ansvarlig: <xsl:value-of
                        select="//arbeidsprosessdokument/arbeidsbeskrivelse/roller/rolle/dc:title"/><br/>
Beskrivelse:
                        <xsl:value-of select="dc:description"/> </xsl:when>
                    <xsl:otherwise> FEIL! Finner ikke igjen rollen: <xsl:value-of select="hvem"/><br/> Som
er ansvarlig
                        for: <xsl:value-of select="dc:description"/> </xsl:otherwise>
                </xsl:choose>
            </p>
        </xsl:template>
        <xsl:template match="vedlegg">
            <xsl:param name="vID" select="'null'"/>
            <xsl:if test="$vID=id">
```

```
                    <xsl:value-of select="dc:title"/>
                </xsl:if>
        </xsl:template>
        <xsl:template match="vedleggsliste"> <h1>VEDLEGGSLISTE</h1> Antall vedlegg: <xsl:value-of
select="count(vedlegg)"/>
                <xsl:for-each select="vedlegg"> <p> <!-- <a name="idfeltet">tittel</a>--> <xsl:text
                disable-output-escaping="yes">&lt;a name&#61;&quot;</xsl:text> <xsl:value-of select="id"/
><xsl:text
                disable-output-escaping="yes">&quot;&gt;</xsl:text> <b><xsl:value-of select="dc:title"/></b>
<xsl:text
                disable-output-escaping="yes">&lt;/a&gt;</xsl:text>, <i><xsl:value-of select="dc:creator"/></i>,
<xsl:value-of
                select="dc:date"/> , Format: <xsl:value-of select="dc:format"/>, Type: <xsl:value-of select="dc:
type"/>, <!-- <a href="URI">Ekstern Link</a>-->
                 <xsl:text disable-output-escaping="yes">&lt;a href&#61;&quot;</xsl:text> <xsl:value-of
select="URI"/><xsl:text
                disable-output-escaping="yes">&quot;&gt;</xsl:text> <xsl:value-of select="URI"/><xsl:text
                disable-output-escaping="yes">&lt;/a&gt;</xsl:text> </p> </xsl:for-each> </xsl:template>
</xsl:stylesheet>
```

Tillegg C

Java Kildekode

```java
package no.ntnu.idi.aposserver;
import java.io.DataOutputStream;
import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet med metoder for å lagre xml-dokument i systemet ut ifra en allerede
 * skrevet fil.
 *
 * @author Axel Wathne
 * @author Erlend Agøy Engum
 */
public class AposFilInnServlet extends HttpServlet {

    Xmlsjef sjef;
    /**
     * Konstruktor som kaller superklassens konstruktor og oppretter en Xmlsjef.
     */
    public AposFilInnServlet() {
        super();
        sjef = new Xmlsjef();
    }

    /**
     * Kaller doPost
     *
     * @see javax.servlet.http.HttpServlet#doGet(javax.servlet.http.HttpServletRequest,
     *      javax.servlet.http.HttpServletResponse)
     */
    protected void doGet(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException {
        doPost(req, res);
    }
    /**
     * Metode som svarer på forespørsler. Forespørselen vil her være å lagre en
     * fil og xmlsjefen brukes til å utføre oppdraget.
     *
     * @param req
     *            Forespørsel
     * @param res
     *            Respons
     * @throws ServletException
     * @throws IOException
     * @see javax.servlet.http.HttpServlet#doPost(javax.servlet.http.HttpServletRequest,
     *      javax.servlet.http.HttpServletResponse)
     */
    protected void doPost(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException {

        res.setContentType("text/html; charset=UTF-8");
        res.setBufferSize(8192);
        DataOutputStream dout = new DataOutputStream(res.getOutputStream());

        try {
            sjef.oppdaterAp(req.getParameter("fil"));
            dout.writeBytes("Arbeidsprosess lagt til.");
        } catch (Exception e) {
            e.printStackTrace();
            dout.writeBytes(e.getMessage());
```

```
    }
  }
}
```

```java
package no.ntnu.idi.aposserver;
import java.io.DataOutputStream;
import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet som, med utgangspunkt i en webform lager et gyldig
 * arbeidsprosessdokument og får lagret dette ved hjelp av Xmlsjef.
 *
 * @author Axel Wathne
 * @author Erlend Agøy Engum
 */
public class AposInnServlet extends HttpServlet {

    Xmlsjef sjef;

    /**
     * Konstruktor kaller superklassens konstruktor og setter en xmlsjef.
     */
    public AposInnServlet() {
        super();
        sjef = new Xmlsjef();
    }

    /**
     * Kaller doPost
     *
     * @see javax.servlet.http.HttpServlet#doGet(javax.servlet.http.HttpServletRequest,
     *      javax.servlet.http.HttpServletResponse)
     */
    protected void doGet(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException {
        doPost(req, res);
    }
    /**
     * Metoden aktiveres etter at et nettskjema er fyllt ut. Det gjøres først
     * noen tester på om skjemaet er riktig utfylt, deretter lages et gyldig
     * xml-dokument av informasjonen.
     *
     * @param req -
     *            Forespørselen
     * @param res -
     *            Responsen
     * @throws ServletException
     * @throws IOException
     * @see javax.servlet.http.HttpServlet#doPost(javax.servlet.http.HttpServletRequest,
     *      javax.servlet.http.HttpServletResponse)
     */
    protected void doPost(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException {

        res.setContentType("text/html; charset=UTF-8");
        res.setBufferSize(8192);
        DataOutputStream dout = new DataOutputStream(res.getOutputStream());
        StringBuffer buf = new StringBuffer();
        buf
                .append("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n")
                .append(
                        "<arbeidsprosessdokument xsi:noNamespaceSchemaLocation=")
```

```
                .append(
                        "\"http://localhost/~axel/schema.xsd\" xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-
instance\"")
                .append(
                        " xmlns:dc=\"http://purl.org/dc/elements/1.1/\">\n                    <dokumentbeskrivelse>");

        //inni dokumentbeskrivelse
        if (req.getParameter("id").trim().equals("")
                || req.getParameter("versjon").trim().equals("")
                || req.getParameter("tittel").trim().equals("")) {
            dout.writeChars("- > id og/eller versjon mangler\n");
            dout.close();
            return;
        }

        buf.append("<identifikator>\n<id>").append(req.getParameter("id"))
                .append("</id><versjon>").append(req.getParameter("versjon"))
                .append("</versjon>\n</identifikator>\n<dc:title>").append(
                        req.getParameter("tittel")).append("</dc:title>\n");
        buf.append("<dc:publisher>").append(req.getParameter("publisher"))
                .append("</dc:publisher>\n");
        buf.append("<dc:language>").append(req.getParameter("language"))
                .append("</dc:language>\n");
        buf.append("<prosessklasse>").append(req.getParameter("prosessklasse"))
                .append("</prosessklasse>\n");

        buf.append("<styringsdokumentnivaa>").append(
                req.getParameter("styringsdokumentnivaa")).append(
                "</styringsdokumentnivaa>\n");
        buf.append("<dc:date>").append(req.getParameter("date")).append(
                "</dc:date>\n<avhengigheter>");
        if (!req.getParameter("avhid").equals(""))
            buf.append("<referanseIdentifikator><id>").append(
                    req.getParameter("avhid")).append("</id>\n<versjon>")
                    .append(req.getParameter("avhversjon")).append(
                            "</versjon></referanseIdentifikator>\n");
        //inni avhengigheter/referanseIdentifikator
        buf.append("</avhengigheter>\n");
        //ut
        buf.append("<endringslogg><endring><versjon>").append(
                req.getParameter("versjon")).append("</versjon><dc:creator>")
                .append(req.getParameter("creator")).append("</dc:creator>\n");
        buf.append("<dc:date>").append(req.getParameter("date")).append(
                "</dc:date><dc:description>").append(
                req.getParameter("description")).append(
                "</dc:description></endring></endringslogg>");
        buf.append("<dc:contributor>").append(req.getParameter("contributor"))
                .append("</dc:contributor>\n");
        buf.append("<dc:coverage>").append(req.getParameter("coverage"))
                .append("</dc:coverage>\n</dokumentbeskrivelse>");

        //arbeidsbeskrivelse
        //rolle
        buf.append("<arbeidsbeskrivelse><roller><rolle><id>").append(
                req.getParameter("rolleid")).append("</id><dc:title>").append(
                req.getParameter("rolletittel")).append(
                "</dc:title><kompetansekrav>").append(
                req.getParameter("kompetansekrav")).append(
                "</kompetansekrav></rolle></roller>");
        //start
        buf.append("<start><id>start</id><forutsetninger><forutsetning>")
                .append(req.getParameter("forutsetning")).append(
                        "</forutsetning></forutsetninger>");
```

```
        buf.append("<prosessbeskrivelse><formaal>").append(
                req.getParameter("formaal")).append("</formaal><maalgruppe>")
                .append(req.getParameter("maalgruppe")).append("</maalgruppe>");
        buf.append("<definisjoner><definisjon>").append(
                req.getParameter("definisjon")).append(
                "</definisjon></definisjoner>");
        buf
                .append("<ansvar><hvem>")
                .append(req.getParameter("rolleid"))
                .append("</hvem><dc:description>")
                .append(req.getParameter("ansvar"))
                .append(
                        "</dc:description></ansvar></prosessbeskrivelse><gaar-til>a1</gaar-til></start>");
        //aktivitet
        buf.append("<aktivitet><id>a1</id><dc:title>").append(
                req.getParameter("atitle")).append("</dc:title>");
        buf.append("<beskrivelse><maal>").append(req.getParameter("maal"))
                .append("</maal><krav>").append(req.getParameter("krav"))
                .append("</krav><hva>").append(req.getParameter("hva")).append(
                        "</hva><hvordan>").append(req.getParameter("hvordan"))
                .append("</hvordan><hvorfor>").append(
                        req.getParameter("hvorfor"));
        buf.append("</hvorfor><hva-om>").append(req.getParameter("hvaom"))
                .append("</hva-om></beskrivelse>");
        buf
                .append("<hvem>")
                .append(req.getParameter("rolleid"))
                .append(
                        "</hvem><status/><kommer-fra>start</kommer-fra><gaar-til>slutt1</gaar-til></
aktivitet>");
        //slutt
        buf
                .append(
                        "<slutt><id>slutt1</id><kommer-fra>a1</kommer-fra><postconditions><postcondition>")
                .append(req.getParameter("postcondition"))
                .append(
                        "</postcondition></postconditions></slutt></arbeidsbeskrivelse><vedleggsliste/>")
                .append("</arbeidsprosessdokument>");
        try {
            sjef.oppdaterAp(buf.toString());
            dout.writeBytes("Arbeidsprosess lagt til.");
        } catch (Exception e) {
            e.printStackTrace();
            dout.writeBytes(e.getMessage());
        }
        dout.close();
    }
}
```

```java
package no.ntnu.idi.aposserver;
import java.io.IOException;
import java.io.StringReader;

import org.jdom.Document;
import org.jdom.JDOMException;
import org.jdom.input.SAXBuilder;

/**
 *
 * Bygger JDOM-dokumenter av strenger og validerer XML-filer opp mot skjemaet de
 * hører til.
 *
 * @author Axel Wathne
 * @author Erlend Agøy Engum
 *
 *
 */
public class AposParser {

    /**
     * Metoden parser dokumentet og returnerer hvorvidt det lot seg validere.
     * IOException kastes ved andre feil
     *
     * @param dokument -
     *            Strengversjon av xml-dokumentet
     * @return true dersom dokumentet er valid, false ellers.
     */
    protected boolean validerFil(String dokument) throws IOException {
        SAXBuilder bob = new SAXBuilder("org.apache.xerces.parsers.SAXParser",
                true);
        bob
                .setFeature("http://apache.org/xml/features/validation/schema",
                        true);
        bob.setFeature("http://xml.org/sax/features/namespaces", true);
        bob
                .setProperty(
                        "http://apache.org/xml/properties/schema/external-noNamespaceSchemaLocation",
                        "http://localhost/~axel/schema.xsd");
        try {
            bob.build(dokument);
        } catch (JDOMException e) {
            return false;
        }
        return true;
    }

    /**
     * Metoden tar inn en streng-representasjon av xml-dokumentet og lager et
     * jdom.Document av den.
     *
     * @param dokument -
     *            strengrepresentasjon av xml-dokument
     * @param valider -
     *            settes true dersom dokumentet skal valideres mot xml-schema
     */
    protected Document documentFraStreng(String dokument, boolean valider)
            throws JDOMException, IOException {

        SAXBuilder bob = new SAXBuilder("org.apache.xerces.parsers.SAXParser",
                valider);
        bob
                .setFeature("http://apache.org/xml/features/validation/schema",
```

```
                true);
        bob.setFeature("http://xml.org/sax/features/namespaces", true);
        bob
                .setProperty(
                        "http://apache.org/xml/properties/schema/external-noNamespaceSchemaLocation",
                        "http://localhost/~axel/schema.xsd");
        bob
                .setProperty(
                        "http://apache.org/xml/properties/schema/external-schemaLocation",
                        "http://localhost/~axel/schema.xsd");
        return bob.build(new StringReader(dokument));
    }
}
```

```
package no.ntnu.idi.aposserver;
import java.io.DataOutputStream;
import java.io.IOException;
import java.util.StringTokenizer;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Enkel servlet som leser inn en arbeidsprosess-ID fra websiden den kalles fra,
 * og ber om å få tilhørende webside i retur.
 *
 * @author Axel Wathne
 * @author Erlend Agøy Engum
 *
 */
public class AposServlet extends HttpServlet {

    /**
     * Metoden gjør det samme som doGet
     *
     * @param request
     * @param response
     * @throws ServletException
     * @throws IOException
     */
    public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        //Vi gjør samme handling for post og get
        doGet(request, response);

    }

    /**
     * Metoden behandler forespørseler. Kaller på Xmlsjef for å lage svar.
     * Forskjellige svar gis ut ifra om brukeren vil se xml eller html.
     *
     * @param request
     * @param response
     * @throws ServletException
     * @throws IOException
     */
    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html; charset=UTF-8");
        response.setBufferSize(8192);

        //Det er hit vi skriver det som skal returneres til browser
        DataOutputStream dout = new DataOutputStream(response.getOutputStream());
        String answer = "";
        try {
            Xmlsjef sjef = new Xmlsjef();
            if (request.getParameter("visning").equals("xml")) {
                answer = "<?xml version=\"1.0\" encoding=\"UTF-8\"?><!DOCTYPE html "
                        + "PUBLIC \"-//W3C//DTD XHTML 1.0 Strict//EN\" "
                        + "\"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd\"> "
                        + "<html xmlns=\"http://www.w3.org/1999/xhtml\" lang=\"en\" xml:lang=\"en\">"
                        + "<head><title>AposWeb</title></head><body><pre>"
                        + xmlsafe(sjef.xmlFraID(request.getParameter("id")))
```

```
                + "</pre></body></html>";
        } else {
            //Her ber vi XMLsjefen om å gi oss en webside med
            // arbeidsprosessen spesifisert i parameteret fra requesten.
            answer = sjef.websideFraID(request.getParameter("id"));
            System.out.println(answer);
        }
    } catch (Exception e) {
        answer = "<html><head><title>error</title></head><body>"
                + e.getMessage() + "</body></html>";
    }
    dout.writeBytes(answer);
    dout.close();
}

/**
 * Metoden sier ifra at vi ikke har servletinfo
 *
 * @return "Servletinfo, det har vi ikke"
 */
public String getServletInfo() {
    return "Servletinfo, det har vi ikke";
}

/**
 * Metoden brukes for å erstatte reserverte html-tegn fra xml-strengen før
 * den sendes til browser, slik at den blir vist riktig.
 *
 * Denne er enkel og grei. Egentlig tungvint. Finnes nok en bedre måte
 * String.replace(String target, String with) el.l eksisterer ikke, finnes
 * for character.
 *
 * Kommentar: jeg skrev denne metoden pre-java1.4, da fantes ikke java.regex
 *
 * @param innput -
 *              xml-dokument med reserverte html-tegn
 * @return Streng der reserverte tegn er erstattet
 */
private String xmlsafe(String innput) {
    StringBuffer ut;
    String tmp = innput;
    StringBuffer inn = new StringBuffer(innput);

    StringTokenizer st = new StringTokenizer(inn.toString(), "&"); //Denne
    // blokken
    // bytter
    // ut
    // ulovlige
    // tegn
    // i
    // strengen
    ut = new StringBuffer(st.nextToken());
    while (st.hasMoreElements()) {
        ut.append("&amp;").append(st.nextToken());
    }

    StringTokenizer su = new StringTokenizer(ut.toString(), "<");
    inn = new StringBuffer(su.nextToken());
    while (su.hasMoreElements()) {
        inn.append("&lt;").append(su.nextToken());
    }

    StringTokenizer sv = new StringTokenizer(inn.toString(), ">");
```

```
        ut = new StringBuffer(sv.nextToken());
        while (sv.hasMoreElements()) {
            ut.append("&gt;").append(sv.nextToken());
        }

        StringTokenizer sw = new StringTokenizer(ut.toString(), "\"");
        inn = new StringBuffer(sw.nextToken());
        while (sw.hasMoreElements()) {
            inn.append("&quot;").append(sw.nextToken());
        }

        StringTokenizer sx = new StringTokenizer(inn.toString(), "'");
        ut = new StringBuffer(sx.nextToken());
        while (sx.hasMoreElements()) {
            ut.append("&apos;").append(sx.nextToken());
        }

        if (tmp.endsWith("<")) {//denne blokken bytter ut ulovlige
            // tegn sist i strengen
            ut.append("&lt;");
        } else if (tmp.endsWith(">")) {
            ut.append("&gt;");
        } else if (tmp.endsWith("&")) {
            ut.append("&amp;");
        } else if (tmp.endsWith("\"")) {
            ut.append("&quot;");
        } else if (tmp.endsWith("'")) {
            ut.append("&apos;");
        }

        if (tmp.startsWith("<")) {//denne blokken bytter ut ulovlige
            // tegn først i strengen
            ut.insert(0, "&lt;");
        } else if (tmp.startsWith(">")) {
            ut.insert(0, "&gt;");
        } else if (tmp.startsWith("&")) {
            ut.insert(0, "&amp;");
        } else if (tmp.startsWith("\"")) {
            ut.insert(0, "&quot;");
        } else if (tmp.startsWith("'")) {
            ut.insert(0, "&apos;");
        }

        return ut.toString();//strengen er ferdig. Dette virker faktisk
    }
}
```

```java
package no.ntnu.idi.aposserver;
import java.util.ArrayList;

import org.jdom.Document;

/**
 * Beholder for informasjon om en arbeidsprosess
 *
 * @author Axel Wathne
 * @author Erlend Agøy Engum
 *
 */
public class Arbeidsprosess {

    private String id, eierid, dok;
    private ArrayList avh;
    private Document dDok;

    /**
     * Konstruktor for arbeidsprosess
     *
     * @param id
     * @param eierid
     * @param dok -
     *            dokumentet på strengform
     */
    public Arbeidsprosess(String id, String eierid, String dok) {
        this.id = id;
        this.eierid = eierid;
        this.dok = dok;
    }

    /**
     * Overlagrer toString();
     *
     * @return xml-dokumentet på strengform
     */
    public String toString() {
        return dok;
    }

    /**
     * Metode for å hente arbeidsprosessens id
     *
     * @return arbeidsprosessens id på form "arbeidsprosessid-versjon"
     */
    public String getID() {
        return id;
    }

    /**
     * Metode for å hente identifikator til arbeidsprosessens eier.
     *
     * @return referanse til arbeidsprosessens eier
     */
    public String getEierID() {
        return eierid;
    }

    /**
     * Metode for å få avhengighetene til en arbeidsprosess. En ArrayList med
     * identifikatorer til andre arbeidsprosesser.
     *
```

```
     * @return en liste med avhengigheter
     */
    public ArrayList getAvh() {
        return avh;
    }

    /**
     * Metode for å lagre avhengighetsliste.
     *
     * @param avh
     *            avhengighetslisten
     */
    public void setAvh(ArrayList avh) {
        this.avh = avh;
    }

    /**
     * Metode for å få xml-dokumentet på dokumentobjektform.
     *
     * @return xml-dokumentet på dokumentobjektform
     */
    public Document getDDok() {
        return dDok;
    }

    /**
     * Metode for å lagre dokumentobjektformen til et xml-dokumen.
     *
     * @param dok
     *            DOM representasjon av xml-dokument.
     */
    public void setDDok(Document dok) {
        dDok = dok;
    }

}
```

```java
package no.ntnu.idi.aposserver;
import java.util.ArrayList;
import java.util.Properties;

import org.jdom.Document;
import org.jdom.Element;

import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;

import com.sun.mail.smtp.SMTPMessage;

/**
 * Inneholder metoder for behandling av avhengigheter.
 *
 * @author Axel Wathne
 * @author Erlend Agøy Engum
 *
 *
 */
public class AvhRes {

    //hente ut dette dokumentets avhengigheter og legger seg til i
    // avhengighetenes lister utenfor selve dokumentet
    /**
     * Metoden finnAvhengingheter: 1: Finne fram avhengighetene internt i
     * dokumentet 2: For hver avhengighet: - Lag en id av "id-versjon" 3:Send
     * tilbake som arrayList
     *
     * @param doc
     *              Dokumentet som skal avhengighetsbehandles
     */
    protected ArrayList finnAvhengigheter(Document doc) {
        Element dokumentbeskrivelse = (doc.getRootElement())
                .getChild("dokumentbeskrivelse");

        ArrayList avhengighetsElementer = new ArrayList(dokumentbeskrivelse
                .getChild("avhengigheter")
                .getChildren("referanseIdentifikator"));
        if (avhengighetsElementer != null) {
            ArrayList avhengighetsIDer = new ArrayList();
            for (int i = 0; i < avhengighetsElementer.size(); i++) {
                Element avhengighet = (Element) avhengighetsElementer.get(i);
                avhengighetsIDer.add(avhengighet.getChildTextTrim("id") + "-"
                        + avhengighet.getChildTextTrim("versjon"));
            }
            return avhengighetsIDer;
        }

        return null;
    }

    /**
     * Metoden formidler melding om oppdatering av en arbeidsprosess til alle
     * eiere av arbeidsprosesser som avhenger av denne.
     *
     * @param eierListe -
     *              liste med eiere av arbeidsprosesser som avhenger av den som nå
     *              oppdateres.
```

```
 * @param apID -
 *             identifikator for arbeidsprosess som er oppdatert.
 * @return boolean - dersom melding ikke kunne formidles
 */
protected boolean formidleOppdateringsMelding(ArrayList eierListe,
        String apID, String gammelID) {
    for (int i = 0; i < eierListe.size(); i++) {
        String eier = (String) eierListe.get(i);
        ArrayList eierInfo = hentEksternEierInfo(eier);
        if (!varsle(
                (String) eierInfo.get(0),
                "Endring i arbeidsprosess " + gammelID,
                "Du står ansvarlig for en arbeidsprosess som avhenger av "
                        + gammelID
                        + ". Det er "
                        + "gjennomført en endring i denne arbeidsprosessen, ny prosess har id "
                        + apID + ". Vennligst følg opp din arbeidsprosess"
                        + "med hensyn til dette.")) {
            System.err.println("Fikk ikke varslet eieren " + eier
                    + " om oppdatering av arbeidsprosess " + apID);
            return false;
        }
    }
    return true;
}

/**
 * Denne metoden må kontakte utenforstående systemer for å finne
 * informasjonen den trenger om eieren
 *
 * @param eier
 * @return en liste med informasjon om eieren.
 */
private ArrayList hentEksternEierInfo(String eier) {
    ArrayList eierInfo = new ArrayList();
    //Kaller opp ekstern enhet og får informasjon som e-post adresse eller
    // lignende. Kan her legge til mange ting
    eierInfo.add(new String("erlendag@idi.ntnu.no"));
    return eierInfo;
}

//hente ut de ap som er avhengige av denne ap (ligger i tabell i db) og
// meddele til eiere av disse at denne er oppdatert
/**
 * Metoden varsle benytter javamail-biblioteket til å sende en gitt melding
 * til en gitt mottaker.
 *
 * @param til -
 *             gyldig epostadresse
 * @param tema
 * @param melding
 * @return true dersom eposten ble sendt, false ellers.
 */
private boolean varsle(String til, String tema, String melding) {
    //Her kan vi sette opp mail-varsel, internt systemvarsel eller noe
    // lignende også
    //Dette er enkel utskrift til skjerm
    System.out.println("Eier " + til
            + " Varsles herved om at arbeidsprosessen " + tema
            + " er oppdatert ");

    //Sender her mail:
    Properties props = new Properties();
```

```
        props.put("mail.smtp.host", "smtp.stud.ntnu.no");
        Session s = Session.getDefaultInstance(props);
        SMTPMessage mesg = new SMTPMessage(s);
        try {
            mesg
                    .setFrom(new InternetAddress(
                            "AposServer <wathne@stud.ntnu.no>"));
            mesg.setRecipient(Message.RecipientType.TO,
                    new InternetAddress(til));
            mesg.setSubject(tema);
            mesg.setContent(melding, "text/plain; charset=UTF-8");
            Transport.send(mesg);

        } catch (AddressException e) {
            e.printStackTrace();
            return false;
        } catch (MessagingException e) {
            e.printStackTrace();
            return false;
        }
        return true;
    }
}
```

```java
package no.ntnu.idi.aposserver;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.HashMap;

/**
 * Benytter seg av JDBC for å snakke med databasen. Den gjør også noe
 * datamanipulasjon.
 *
 * @author Axel Wathne
 * @author Erlend Agøy Engum
 */
public class Dbconn {

    private String URI;
    private PreparedStatement s1;
    private ResultSet rs;
    private Connection c;

    /**
     * Konstruktor.
     *
     * @param URI
     */
    protected Dbconn(String URI) {
        this.URI = URI;

    }

    /**
     * Gjør oppkobling til database. Laster først riktig driver, i dette
     * tilfellet MySql ConnectorJ.
     *
     * @return true dersom alt går bra, false ellers.
     */
    protected boolean connect() {
        try {

            Class.forName("com.mysql.jdbc.Driver").newInstance();
            c = DriverManager.getConnection("jdbc:" + URI);
        } catch (Exception e) {
            // Catch SQLExceptions og ClassNotFoundExceptions.
            e.printStackTrace();
            return false;
        }
        return true;
    }

    /**
     * Metode for å opprette en tabell. Inneholder derfor databseskjema.
     *
     * @return true dersom alt går bra, false ellers.
     */
    protected boolean createTable() {
        try {
            s1 = c
                    .prepareStatement("CREATE TABLE arbeidsprosesser (id char(50) PRIMARY KEY,"
                            + " eierid char(50), prosessdokument text, avh blob)");
            s1.execute();
```

```
    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
    return true;
}

/**
 * Metode for å slette tabellen fra databasen.
 *
 * @return true dersom alt går bra, false ellers.
 */
protected boolean dropTable() {
    try {
        s1 = c.prepareStatement("DROP TABLE arbeidsprosesser");
        s1.execute();
    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
    return true;

}

/**
 * Setter inn en arbeidsprosess i databasen.
 *
 * @param ap -
 *             et arbeidsprosessobjekt
 * @return true dersom alt går bra
 * @throws SQLException
 */
protected boolean settInnAP(Arbeidsprosess ap) throws SQLException {
    s1 = c
            .prepareStatement("INSERT INTO arbeidsprosesser VALUES (?, ?, ?, ?)");
    s1.setString(1, ap.getID());
    s1.setString(2, ap.getEierID());
    s1.setString(3, ap.toString());
    s1.setObject(4, ap.getAvh());
    s1.execute();
    return true;
}

/**
 * Returnerer alt som ligger i databasen til System.out.
 *
 * @return true dersom alt går bra, false ellers.
 */
protected boolean dump() {
    try {
        s1 = c.prepareStatement("SELECT * FROM arbeidsprosesser");
        rs = s1.executeQuery();
        while (rs.next())
            System.out.println(rs.getString(1) + ": \n" + rs.getString(2));

    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
    return true;
}

/**
```

```
 * Gjør en enkel SELECT * på id'en den tar som innparameter. Returnerer
 * treff som streng, ellers null.
 *
 * @return et arbeidsprosessdokument med all info om ap.
 */
protected Arbeidsprosess hentApFraID(String id) {

    try {
        s1 = c
                .prepareStatement("SELECT * FROM arbeidsprosesser where id=?");
        s1.setString(1, id);
        rs = s1.executeQuery();
        Arbeidsprosess ap;
        while (rs.next()) {
            ap = new Arbeidsprosess(id, rs.getString("eierid"), rs
                    .getString("prosessdokument"));
            ap.setAvh((ArrayList) rs.getObject("avh"));
            return ap;
        }
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
    return null;
}

/**
 * @param minID
 * @param avhengighetsIDer
 *
 * Denne metoden tar inn en id, minID, og en arraylist med ider,
 * avhengighetsID. Idene i arraylisten refererer til objekter som skal ligge
 * i databasen og som skal ha sin egen arraylist med ider, der den minID
 * legges til.
 *
 */
protected synchronized void registrerAvhengigheter(String minID,
        ArrayList avhengighetsIDer) throws UgyldigeAvhengigheterException {

    try {
        PreparedStatement ut = c
                .prepareStatement("SELECT avh FROM arbeidsprosesser where id=?");
        PreparedStatement innigjen = c
                .prepareStatement("UPDATE arbeidsprosesser SET avh=? WHERE id=?");
        HashMap hm = new HashMap();

        for (int i = 0; i < avhengighetsIDer.size(); i++) {
            ut.setString(1, (String) avhengighetsIDer.get(i));
            rs = ut.executeQuery();
            if (!rs.next())
                throw new UgyldigeAvhengigheterException(
                        "En arbeidsprosess ble ikke funnet i DB");
            hm.put((String) avhengighetsIDer.get(i), (ArrayList) rs
                    .getObject("avh"));
        }

        ArrayList al;

        for (int i = 0; i < avhengighetsIDer.size(); i++) {
            al = (ArrayList) hm.get((String) avhengighetsIDer.get(i));
            if (al == null)
                al = new ArrayList();
            al.add(minID);
```

```
            innigjen.setObject(1, al);
            innigjen.setString(2, (String) avhengighetsIDer.get(i));
            innigjen.execute();

        }

    } catch (SQLException e) {
        e.printStackTrace();
    }

}

/**
 * Metode som henter den eksterne avhengighetslista, det vil si listen over
 * alle prosesser som er avhengige av denne, til en gitt arbeidsprosess.
 *
 * @param id
 * @return liste med eksterne avhengigheter
 */
protected ArrayList henteAvhengighetsListe(String id) {

    try {
        s1 = c
                .prepareStatement("SELECT avh FROM arbeidsprosesser where id=?");
        s1.setString(1, id);
        rs = s1.executeQuery();
        Arbeidsprosess ap;
        while (rs.next()) {
            return ((ArrayList) rs.getObject("avh"));
        }
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
    return null;
}

/**
 * Metode som finner eieridene til et sett arbeidsprosesser
 *
 * @param ider -
 *             Arraylist med ider til arbeidsprosesser
 * @return Arraylist med eieridеer.
 */
protected ArrayList hentEiere(ArrayList ider) {
    try {
        s1 = c
                .prepareStatement("SELECT eierid FROM arbeidsprosesser where id=?");

        for (int i = 0; i < ider.size(); i++) {

            s1.setString(1, (String) ider.get(i));
            rs = s1.executeQuery();
            if (rs.next()) {
                ider.set(i, rs.getString("eierid"));
            } else {
                ider.set(i, null);
            }

        }

    } catch (SQLException e) {
        e.printStackTrace();
```

```
            return null;
        }
        return ider;
    }
}
```

```java
package no.ntnu.idi.aposserver;
import java.io.File;
import java.io.StringReader;
import java.io.StringWriter;

import javax.xml.transform.Result;
import javax.xml.transform.Source;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerConfigurationException;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.stream.StreamSource;

/**
 * Har ansvar for å transformere xml til html ved hjelp av xsl. Benytter Xalan-J
 * 2.
 *
 * @author Erlend Agøy Engum
 */
public class JdomOgXslt {

    /**
     * Metoden Transformer gjennomfører en transformasjon fra xml-dokument til
     * html ved hjelp av et xsl. Psaudokode: Filer: xmlFil, xslFil, htmlFil
     * (utfil) Strømmer: xmlKilde = new StreamSource(xmlFil) xslKilde = new
     * StreamSource(xslFil) utStrøm = new StreamResult(htmlFil) Transformer: new
     * TransformerFactory tF = TransformerFactory.newInstance(); new Transformer
     * t = tF.newTransformer(xslKilde) transformer.transform(xmlKilde, utStrøm)
     *
     *
     * @param xmlIn -
     *            xml-dokumentet
     * @param xslLokasjon
     *            Lokasjonen til xsl-dokumentet
     * @return En streng som inneholder html-kode
     */
    protected String transformer(String xmlIn, String xslLokasjon) {
        //lager filer av inputstrenger. Her kan det bygges inn logikk slik at
        // en også kan ta inn URI og lage strømmer.
        StringReader xml = null;
        File xsl = null;
        StringWriter html = null;
        try {
            System.out.println("Laster filene...");
            xml = new StringReader(xmlIn);
            xsl = new File(xslLokasjon);
            html = new StringWriter();
        } catch (NullPointerException e) {
            System.err.println("Problemer med å finne en av filene: "
                    + xslLokasjon);
            e.printStackTrace();
        }

        //Strømmer som må til for transformasjonen. Må være av typen Source og
        // Result
        System.out.println("... lager ut og inn strømmer...");
        Source xmlKilde = new StreamSource(xml);
        Source xslKilde = new StreamSource(xsl);
        Result htmlUt = new StreamResult(html);

        //Lage transformer av xsl Templatene
        System.out.println("... lager transformer og omformer dokumentet...");
```

```
        TransformerFactory tF = TransformerFactory.newInstance();
        try {
            Transformer transform = tF.newTransformer(xslKilde);
            transform.transform(xmlKilde, htmlUt);
        } catch (TransformerConfigurationException e) {
            System.err
                    .println("Her har det skjedd en feil under konfigurering av transformer med xsldokument");
            e.printStackTrace();
        } catch (TransformerException e) {
            System.err
                    .println("Her har det skjedd en feil under transformering fra xml til html");
            e.printStackTrace();
        }

        StringWriter sw = (StringWriter) ((StreamResult) htmlUt).getWriter();
        String retur = sw.toString();
        System.out.println("... da skal alt være klart!");
        return retur;
    }
}
```

```java
package no.ntnu.idi.aposserver;

/**
 * Klasse for feilmeldinger ved ugyldige avhengigheter.
 *
 * @author Axel Wathne
 * @author Erlend Agøy Engum
 */
public class UgyldigeAvhengigheterException extends Exception {
    /**
     * Konstruktor som kaller superklassens konstruktor.
     */
    public UgyldigeAvhengigheterException() {
        super();
    }
    /**
     * Konstruktor som kaller superklassens konstruktor.
     *
     * @param message
     */
    public UgyldigeAvhengigheterException(String message) {
        super(message);
    }
    /**
     * Konstruktor som kaller superklassens konstruktor.
     *
     * @param message
     * @param cause
     */
    public UgyldigeAvhengigheterException(String message, Throwable cause) {
        super(message, cause);
    }
    /**
     * Konstruktor som kaller superklassens konstruktor.
     *
     * @param cause
     */
    public UgyldigeAvhengigheterException(Throwable cause) {
        super(cause);
    }
}
```

```java
package no.ntnu.idi.aposserver;

import java.io.IOException;
import java.sql.SQLException;
import java.util.ArrayList;
import org.jdom.Document;
import org.jdom.Element;
import org.jdom.JDOMException;

/**
 *
 * Dette er en styringsklasse som setter sammen funksjonalitet fra de andre
 * klassene
 *
 * @author Erlend Agøy Engum
 * @author Axel Wathne
 *
 *
 */
public class Xmlsjef {

    //Her settes addressen til mySql-serveren. Denne verdien kan for eksempel
    // heller leses inn fra Deploymentdescriptoren.
    static final String adr = "mysql://localhost/apos?user=luser&password=toastjern";
    Dbconn db;

    /**
     * Konstruktor: Oppretter en databaseforbindelse.
     */
    public Xmlsjef() {

        db = new Dbconn(adr);
        db.connect();

    }

    /**
     * @param id
     * @return Webside i html-format
     *
     * Denne metoden brukers til å hente en arbeidsprosess fra databasen,
     *
     */
    public String websideFraID(String id) throws JDOMException, IOException {

        return new JdomOgXslt().transformer((db.hentApFraID(id).toString()),
                "/Applications/eclipse/workspace/aposserver/transform.xsl");
    }

    /**
     * Metoden skal lage en streng med html-kode fra et xml-dokument: Benytter
     * AposParser til å validere xml-dokumentet. Benytter JdomOgXslt til
     * transformasjonen til html.
     *
     * @param dok
     * @return html-representasjon av xml-dokumentet.
     * @throws JDOMException
     * @throws IOException
     */
    public String validertWebsideFraStreng(String dok) throws JDOMException,
            IOException {
        System.out.println("Xmlsjef:");
        //new AposParser().documentFraStreng(dok, false);
```

```
    System.out.println("validert");
    return new JdomOgXslt().transformer(dok,
            "/Applications/eclipse/workspace/aposserver/transform.xsl");
}

/**
 * Henter en arbeidsprosess fra databasen og presenterer den for
 * nettleseren.
 *
 * @return xml-dokument
 */
public String xmlFraID(String id) {
    return db.hentApFraID(id).toString();
}

/**
 * Metode som implementerer UseCase oppdater arbeidsprosess
 *
 * 1: Parse dokumentet med eller uten validering og få tilbake Document 2:
 * Lag Arbeidsprosess (objekt). Legg inn id, eier og dok 3:
 * SetdDok(Document) 4: AvhRes.oppdaterAvhengigheter(Document) får tilbake
 * ArrayList med referanseIDer 5: Send referanseIDer og minID til
 * databasesklasse for å oppdatere 6: Legg dok i databasen.
 * insertAP(Arbeidsprosess) 7: Hente en liste med eierinformasjon for alle
 * prosesser som avhenger av den som oppdateres 8: Varsle eierne av disse
 * prosessene
 *
 * @param dok -
 *            xml-dokumentet i en streng
 * @return boolean - true hvis ingen exception ble kastet.
 */
public boolean oppdaterAp(String dok) throws JDOMException, IOException,
        SQLException, UgyldigeAvhengigheterException {
    Document apDoc = null;

    apDoc = (new AposParser()).documentFraStreng(dok, true);

    Element dokumentbeskrivelse = (apDoc.getRootElement())
            .getChild("dokumentbeskrivelse");
    String minID = (dokumentbeskrivelse.getChild("identifikator"))
            .getChildTextTrim("id")
            + "-"
            + (dokumentbeskrivelse.getChild("identifikator"))
                    .getChildTextTrim("versjon");
    String forrigeID = (dokumentbeskrivelse.getChild("identifikator"))
            .getChildTextTrim("id")
            + "-"
            + (Integer.parseInt(dokumentbeskrivelse.getChild(
                    "identifikator").getChildTextTrim("versjon")) - 1);
    String minEier = "";

    ArrayList liste = new ArrayList(dokumentbeskrivelse.getChildren());
    for (int i = 0; i < liste.size(); i++) {
        if (((Element) liste.get(i)).getName().equals("publisher"))
            minEier = ((Element) liste.get(i)).getText();
    }
    Arbeidsprosess ap = new Arbeidsprosess(minID, minEier, dok);
    ap.setDDok(apDoc);
    //System.out.println("Resultat: "+ap.getID()+" "+ap.getEierID()+"\n
    // "+ap);
    AvhRes ar = new AvhRes();

    //Registrere meg selv
```

```
        //Finner ut hvilke andre arbeidsprosesser jeg er avhengig av
        ArrayList avhengighetsIDer = ar.finnAvhengigheter(apDoc);
        //Forteller arbeidsprosessene jeg er avhengig av at jeg er avhengig av
        // dem ved å legge min ID til i deres arraylister.
        if (avhengighetsIDer != null)
            db.registrerAvhengigheter(minID, avhengighetsIDer);
        //Legge meg inn i DB
        db.settInnAP(ap);
        //Si fra til de som var avhengige av den forrige versjonen av meg at
        // jeg er blitt oppdatert
        //Hente min liste der andre har registrert seg
        ArrayList avhengighetsliste = db.henteAvhengighetsListe(forrigeID);
        //Hvis andre var avhengige av den forrige versjonen av meg
        if (avhengighetsliste != null) { //
            ArrayList avhengighetseierliste = db.hentEiere(avhengighetsliste);
            ar.formidleOppdateringsMelding(avhengighetseierliste, minID,
                    forrigeID);
        }

        return true;
    }

    /**
     * Metoden benytter AposParser til å lage et Document fra xml-dokument
     * hentet fra databasen.
     *
     * @param id
     *              Arbeidsprosessens unike id
     * @return Nytt Document fra databasen
     * @throws JDOMException
     * @throws IOException
     */
    protected Document documentFraID(String id) throws JDOMException,
            IOException {
        return new AposParser().documentFraStreng(
                db.hentApFraID(id).toString(), false);
    }

}
```

Tillegg D

Kilde- og dokumentasjons-CD