telenor

R&I Research Note

R&I N 31/2007

Tjalve Aarflot

aCCoFrame: Et kontekstrammeverk for CAIM

R&I-notat **N 31/2007**

Tittel aCCoFrame: Et kontekstrammeverk for CAIM

Forfatter(e) Tjalve Aarflot

ISBN / ISSN /0809-1021
Gradering ÅPEN
Dato 2007.08.03

Sammendrag

Dette notatet beskriver aCCoFrame: et rammeverk for å legge kontekstinformasjon, fra forskjellige kilder, inn i bilder's metadata. Notatet tar kort for seg motivasjonen for et slikt system, beskriver den tekniske løsningen, samt peker på utfordringer for videreutvikling. En prototyp på en tjeneste som benytter aCCoFrame er også beskrevet.

Emneord

aCCoFrame, CAIM, bilder, kontekst, annotering, posisjonering, GPS, Argos

Title

aCCoFrame: a CAIM Context Framework

Abstract

This research note describes aCCoFrame: a framework system for adding context information, from different sources, to an image's metadata. The note discusses the motivation for the system, describes the technical solution, and lists challenges for further development. A prototype of a service which utilises aCCoFrame is also described.

Forord

Dette arbeidet er utført som en del av samarbeidsavtalen mellom Universitetet i Tromsø og Telenor R&I, og knyttet opp mot CAIM-prosjektet [1] ved UiT. Arbeidet er hovedsaklig utført under en seks ukers sommerjobb hos Telenor R&I, under veileding av Sigmund Akselsen, Randi Karlsen og Anders Schürmann.

Arbeidet har et sterkt teknisk fokus, og dokumentasjonen gjenspeiler dette.

Hovedarbeidet har bestått i å utvikle rammeverket aCCoFrame (a CAIM Context Framework), som tilbyr sammenknytning av bilder og kontekstinformasjon. Det er også utviklet en prototype, WeatherContext, som synliggjør funksjonaliteten til aCCoFrame.

Innhold

1 Innledning

Det er i dag et økende behov for effektivt å kunne søke i bildeinformasjon. Dagens teknikker for bildefremhenting (Image Retrieval), som enten er tekst-basert eller innholds-basert fremhenting, har klare begrensninger. Et ofte referert problem er ”the semantic gap problem” som oppstår fordi brukerne ønsker å søke etter bilder basert på en semantisk forståelse av bildet, mens dagens teknikker ofte tilbyr fremhenting kun basert på lav-nivå egenskaper ved bildet. Informasjonen et bilde formidler til en person er fremdeles hovedsaklig skjult for datamaskiner, og dermed også utilgjengelig for automasjon.

Dersom bildene var annotert med kontekstinformasjon, ville man oppnå mye bedre resultat fra bildefremhenting, men med dagens store bildedatabaser er manuell annotering av bildene en for tidkrevende prosess. Mange forskningsmiljøer jobber imidlertid med å automatisere innhenting av slik informasjon fra bilder, og vi kan derfor anta at det etter hvert vil bli bildedatabaser tilgjengelig der bildene er tilknyttet kontekstinformasjon.

Målet med dette arbeidet er derfor å lage et rammeverk for å bruke kontekstinformasjon som alt er lagret i et bilde som basis for innhenting av ny og relevant tilleggsinformasjon fra eksterne kilder. Videre vil den nye informasjonen som blir innhentet bli lagret sammen med bildet. Senere vil denne informasjonen blant annet kunne benyttes som kontekst for et nytt informasjonssøk.

Angrepsvinkelen som her benyttes for å forstå konteksten til et bilde er å bruke EXIF[1]-informasjonen som blir lagret i en del bildeformater. I EXIF-informasjonen[8] kan man f.eks. finne dato og tidsstempel, og i noen tilfeller også GPS[2]-informasjon. Det vanligste bildeformatet med EXIF-informasjon er i dag JPEG[3]-bilder, som er standarden for de fleste digitale kameraer.

Dette konseptet blir eksemplifisert i demonstratoren som er utviklet, der man bruker tidsstempelet lagret i EXIF-informasjonen til bildet som basis for å innhente værdata og lagre det sammen med bildet.

Rammeverket aCCoFrame[4] kan utvides til også å bruke andre EXIF-felt som basis, og gjøre søk etter annen data enn værdata.

[1] *Exchangable Image File Format*
[2] *Global Positioning System*
[3] *Joint Photographic Expert Group*
[4] *A CAIM Context Framework*

2 Bakgrunn

Etter at CAIM-prosjektet ble opprettet ved UiT, har det vært utført en god del arbeid knyttet til dette prosjektet [2][3][4], men arbeidene har stort sett vært uavhengige av hverandre. Nå var det ønske om å samles om en felles plattform for nye arbeid, slik at muligheten til samarbeid på kryss av dem øker. En felles plattform vil også øke gjenbruksverdien av de arbeidene som blir utført. Det ble i tillegg forespurt noen generelle verktøy som kunne søke etter og lagre kontekstinformasjon til bilder, ved å ta utgangspunkt i EXIF-informasjonen som ligger lagret i noen bildetyper[5].

Ut av en liten deltidsstilling noen måneder på våren 2007, som hovedsaklig gikk med til forprosjektering, ble det opprettet en seks ukers sommerjobb ved Telenor R&I sommeren 2007. Oppgaven var å designe og utvikle verktøyene ovenfor ved bruk av mellomvareplattformen Argos [5], som er et annet prosjekt ved UiT. Dermed ville man i tillegg få litt erfaring med hvordan Argos fungerte som en felles plattform for videre arbeid med CAIM-prosjekter.

Verktøyene endte til slutt opp med å bli et rammeverk, aCCoFrame[6], basert på Argos, der tjenesteytere kan tilby kontekstinformasjon til klienter gjennom en sentral proxy. Demonstratoren WeatherContext ble implementert for å vise aCCoFrame i praktisk bruk, og aCCoFrame ble gitt et webbasert grensesnitt bygd med AJAX[7] og JSP[8].

2.1 Context-Aware Image Management

”The CAIM project introduces image context-awareness as a means of supporting image-based information retrieval in distributed and mobile environments”.[1]

CAIM er et samarbeidsprosjekt ved UiB, UiT, NTNU og Telenor som ser på bilde-basert informasjonsinnhenting når man benytter kontekstgjenkjennende metoder. Det er i dag et økende behov for effektivt å kunne søke i bildeinformasjon. Dagens teknikker for bildefremhenting (Image Retrieval), som enten er tekst-basert eller innholds-basert fremhenting, har klare begrensninger. Et ofte referert problem er ”the semantic gap problem” som oppstår fordi brukerne ønsker å søke etter bilder basert på en semantisk forståelse av bildet, mens dagens teknikker ofte tilbyr fremhenting basert på lav-nivå egenskaper ved bildet. Resultatet man får etter et bildesøk er derfor som regel ikke godt nok. CAIMprosjektet undersøker disse problemstillingene.

2.2 Argos

”*The purpose of Argos is to be a research platform for the development of a Personal Middleware System (as opposed to an Enterprise middleware system).”[5]*

Argos-plattformen er bygd over Java og JMX[9], og tilbyr blant annet en integrert webserver, en integrert database, og monitoreringsverktøyet Argus. Før Argos

[5] *JPEG, TIFF*
[6] *a CAIM Context Framework*
[7] *Asynchronous Javascript and XML*
[8] *Java Server Pages*
[9] *Java Management eXtensions*

ble et mer generelt rammeverk, var det tiltenkt en rolle som et personlig kontekshåndteringssystem, og prosjektet het CoMS: Context Management System[12]. Argos er derfor en naturlig kandidat å vurdere når man skal finne en felles plattform for kommende arbeider i CAIM.

2.3 Relatert arbeid

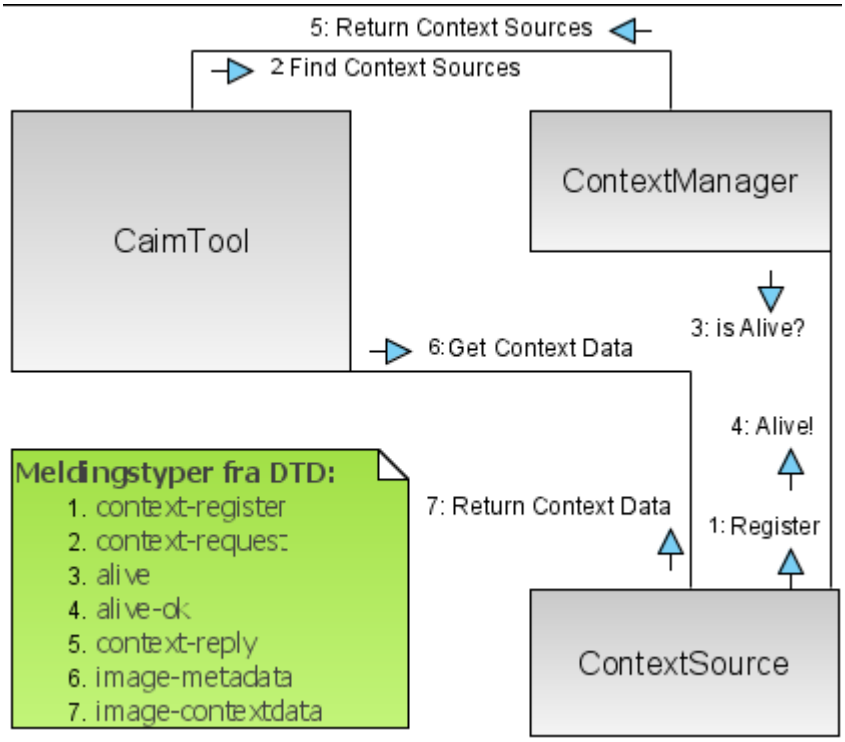
Gjennom to gjesteforelesninger ved IFI 2005 og 2006 presenterte Dr. Cathal Gurrin arbeidet som blir gjort på Centre for Digital Video Processing ved Dublin City University (DCU) [6]. Der er flere prosjekter som er interessante i forhold til CAIM, men foreløpig er det ikke inngått noe samarbeid.

3 Arkitektur

aCCoFrame-rammeverket er inndelt i tre komponenter. Disse kan kjøre i ulike Argos-kontainere og er med andre ord geografisk uavhengige av hverandre. I midten av rammeverket står ContextManager-komponenten, som fungerer som et bindeledd mellom brukere og tjenesteytere. CaimTool-komponenten kjører grensesnittet mot brukerne, og tilbyr håndtering av bilder og filer. Hver bruker vil kjøre en egen, lokal CaimTool.

Den tjenesteytende komponenten er ContextSource, som tilbyr kontekstinformasjon basert på metainformasjon fra bilder. ContextManager står dermed som en sentral proxy mellom CaimTool og ContextSource. Det kan finnest mange ulike ContextSource-komponenter som har registrert seg hos ContextManager, og mange brukere kan kjøre CaimTool samtidig.

Figur 1 gir en oversikt over komponentene i aCCoFrame. *Figur 5* gir en mer detaljert oversikt over kommunikasjonen mellom dem.



Figur 1: Arkitektur av aCCoFrame. Sammenhengen og kommunikasjonen mellom komponentene.

3.1 CaimTool

CaimTool er en selvstendig komponent som driver webgrensesnittet. Den vil som basis-funksjon alltid tilby filbehandling, bildevisning og visning av EXIF-informasjonen knyttet til et bilde. CaimTool trenger kontakt med en ContextManager for å kunne finne kontekstkilder. Dersom den har kontakt med en (eller flere) kontekstkilder vil man også kunne spørre etter kontekst for bildet, og få det lagret i bildets EXIF-header. Det kan kjøres mange CaimTool klienter samtidig. Normalt vil hver bruker kjøre sin egen CaimTool.

3.2 ContextManager

ContextManager er en serverkomponent som det bare skal finnes en instans av. Alle kontekstkilder må registrere seg hos ContextManager, og CaimTool kan spørre ContextManager etter kontekstkilder. ContextManager fungerer derfor som en proxy mellom brukere og tjenesteytere, og står for service-discovery.

3.3 ContextSource

ContextSource er komponenten som tilbyr kontekstkinformasjon til klientene. Navnet på denne komponenten kan velges fritt, etter reglene for navngiving av komponenter i Argos. En implementasjon av ContextSource-komponenten vil få et beskrivende navn, for eksempel WeatherContext. En ContextSourcekomponent vil motta bildemetadata fra CaimTool klienter, og bruke denne til å gjøre ett oppslag i en database for å finne kontekstinformasjon, som deretter blir returnert til CaimTool-klienten.

4 Utvikling

Utviklingen av aCCoFrame var i starten bare utviklingen av CaimTool (seksjon 3.1), som skulle bli et verktøy for å lese og skrive EXIF-informasjon fra bilde-filer. Etterhvert som nytten og bruken av et slikt verktøy ble diskutert mer, ble planene lagt for de komponentene som i dag inngår i aCCoFrame rammeverket. CaimTool fikk selskap av ContextManager (seksjon 3.2) og ContextSource (seksjon 3.3).

Siden aCCoFrame bygger på Argos får det mye funksjonalitet gratis, og kodebasen er derfor behagelig liten. Men de mange ulike funksjonene som aCCoFrame knytter sammen har likevel gjort utviklingen krevende.

Utviklingen har foregått stegvis, med fokus på så adskilte moduler som mulig i hver omgang. Grovt sett kan utviklingen skisseres slik.

1. Utvikling av ExifReader og ExifWriter for å lese og skrive EXIF-informasjon fra bilder
2. Utvikling av et JSP-basert webgrensesnitt for filhåndtering, med støtte for komprimerte formater og batch-prosessering
3. Prototyping av CaimTool: integrering av steg 1 og 2, med tillegg av funksjonalitet for håndtering av bilder
4. Komponentutvikling i Argos: CaimTool, ContextManager og ContextSource
5. Systemutvikling: API (Tillegg C) og kommunikasjonen mellom de tre komponentene, inkludert automatisk validering av XML[10]-payload opp mot caim-messages-DTD[11]
6. AJAX-basert webgrensesnitt for CaimTool, med producer-consumer meldingssystem for å håndtere den asynkrone kommunikasjonen mellom CaimTool og ContextSource
7. Prototyping avWeatherContext (seksjon 5.2) som demonstrator for funksjonaliteten og mulighetene som ligger i aCCoFrame

4.1 Designvalg

aCCoFrame er, som Argos, utviklet i Java, og komponentene er i all hovedsak POJO[12].

Siden Argos tilbyr den integrerte webserveren Jetty[15] var det naturlig å bruke JSP for websidene. Mye av prosesseringen av data som vises i websidene blir gjort i funksjoner i bønnene som ligger i uit.caim.webbeans[13]-pakken. Dermed kan koden i JSP-sidene holdes renest mulig. For AJAX-funksjonalitet blir javascriptbiblioteket mootools[7] benyttet.

Kommunikasjonen mellom komponentene foregår vha. JMX Notifications, som er integrert i Argos. Selve payloaden i notifikasjonene er et XML-dokument som følger APIet beskrevet i Tillegg C.

[10] *eXtensible Markup Language*
[11] *Document Type Definition*
[12] *Plain Old Java Objects*
[13] *uit.caim.webbeans* http://caim.cs.uit.no/trac/browser/trunk/CaimTool/src/uit/caim/webbeans

4.2 Status

Fokus for utviklingen gikk etter hvert over til å få demonstratoren WeatherContext fungerende og presentabel. Dette er delvis oppnådd i og med at den er fungerende, men ikke så presentabel som den gjerne skulle vært. Det å jobbe med selve webdesignet er likevel egentlig bare belønningen etter at alt annet fungerer, og det blir forhåpentligvis mulig å pusle litt med dette i etterkant.

Ulempen med et grelt utseende grensesnitt er at det ikke tilstrekkelig formidler kompleksiteten i alt som foregår i bakgrunnen.

Et problem med systemtjenesten twoWay i Argos, som aCCoFrame benytter for å sette opp toveis kommunikasjon mellom komponentene, gjør at det foreløpig bare fungerer å ha en ContextSource-komponent kjørende. Dette er dog tilstrekkelig for demonstratoren sin del.

4.3 Gjenstående arbeid

Alt i alt var spekteret ganske bredt for hva som skulle designes og utvikles i løpet av fire uker, og det ble etter hvert nødvendig å ta en del snarveier. I de fleste tilfellene var årsaken at aCCoFrame dynamisk ble til ettersom problemstillinger dukket opp og ble forsert, og en del funksjonalitet måtte da designes på nytt for å fungere. Dette ble innimellom for tidkrevende eller medførte for stor fare for at også annen funksjonalitet ville måtte omdesignes. Derfor ble en del løsninger hardkodet inn i komponentene slik at de nå ikke tar hensyn til alle generelle tilfeller aCCoFrame helst skal dekke, men i stedet bare fungerer for demonstratoren WeatherContext.

En grov liste over gjenstående arbeid:

- forbedre dokumentasjon (JavaDoc) og annotering i nesten alt av kildekode
- løse problemene med twoWay systemtjenesten til Argos, eller sette opp denne kommunikasjonen manuelt. Ved en manuell løsning vil man derimot få en høna-eller-egget problematikk.
- forbedre designet på webgrensesnittet betraktelig
- skille de tre komponentene CaimTool, ContextManager og ContextSource skikkelig fra hverandre, slik at de faktisk kan kjøre uavhengig av hverandre. Dette er en liten og enkel jobb, men vil forsterke problemene med twoWay, så de bør løses først
- standardisere og renske opp i bruken av kontekstype gjennom hele koden, fra select-boksen på index.jsp websiden til definisjonen av dem i DTDdokumentet.
- implementere et bedre format på innholdet i image-contextdata meldingen fra ContextSource til CaimTool. Formatet på kontekstdata bør være et XML-dokument med tilhørende XSL-stilskjema for å tilby et så bredt og generalisert format for kontekstdata som mulig, slik at dette ikke legger noen begrensning på hva slags ContextSource som kan legges til. En del av håndteringen av kontekstdata er per i dag hardkodet for demonstratoren WeatherContext. Kontekstdata må også lagres i rett format i EXIF-headeren til bildet.

Før man bygger videre på aCCoFrame, kan det være lurt å ta tak i det på nytt og foreta en redesign / reimplementering. Dette er gjerne slik det er med eksperimentell programmering.

4.4 Erfaringer med Argos

Å utvikle aCCoFrame som et rammeverk basert på Argos-plattformen oppleves som positivt. Det Argos tilbyr har vært akkurat hva aCCoFrame hadde behov for, blant annet en webserver, et asynkront kommunikasjonssystem mellom komponenter, en cron[14]-lignende scheduler i form av @execute-annotasjonen, og ellers frie tøyler.

Uavhengig av dette vil det å basere fremtidige arbeid på Argos medføre at utvikling vil skje i samme miljø ved bruk av samme programmeringsspråk, og dermed øke gjenbruksverdien og samarbeidsmulighetene.

Argos er veldig lite begrensende, foruten at man må holde seg til Java-verdenen, og det burde derfor være mulig for de aller fleste å utvikle på Argos-plattformen, selv om de ikke tar i bruk den funksjonaliteten som Argos tilbyr.

Ulempen kan være at Argos for tiden ikke har noen hovedutvikler, men dette har ikke bydd på noen problemer for aCCoFrame sin del.

[14] *cron* http://en.wikipedia.org/wiki/Cron

5 Brukersenarioer

aCCoFrame har som mål å være et generelt og utvidbart rammeverk, med et enkelt grensesnitt for å søke etter kontekst for bilder og å håndtere bilder med kontekst. Utgangspunktet er at man har digitale bilder som ikke er spesielt annotert på noen måte, men som har et minimum av EXIF-informasjon lagret i selve bildefilen. De aller fleste JPEG-bilder vil ha EXIF-informasjon dersom de er lagret av et moderne bildebehandlingsprogram eller tatt med et digitalt kamera. Tanken er at de fleste digitale bilder per i dag har denne informasjonen tilgjengelig, men kun denne. Utfordringen er derfor å lage et system som kan utnytte denne informasjonen til å sette bildet inn i kontekst, uten å gå veien gjennom avanserte bildegjenkjenningsalgoritmer eller nitidig manuell annotering av bildene.

Under følger et eksempel som bygger litt på tidligere arbeid som er gjort ved Telenor R&I der bildesøk kobles sammen med turistinformasjon [16]. Videre følger et eksempel der det innhentes værdata. Sistnevnte er det utviklet en prototyp for, WeatherContext, som en del av arbeidet med aCCoFrame sommeren 2007.

5.1 Turistinformasjon

Turistnæringen er et eksempel på en bransje som kan nyttiggjøre seg av denne typen tjenester.

Kontekstkilden kan kjøre på en sentral maskin, eller være fordelt utover hos lokale turistbyrå, der hvert byrå har ansvaret for å oppdatere informasjonen i denne. Dersom kontekstkilden kjører på en sentral maskin kan den likevel slå opp informasjonen i databaser lokalt for hvert turistbyrå. Det finnes med andre ord flere måter å organisere strukturen til systemet på.

Kontekstkilden kan bruke posisjonsinformasjon fra EXIF-headeren til bildet, som GPS-Latitude og GPS-Longitude. GPS er i dag ganske nøyaktig (1-3 meter[15]), og man kan derfor ha temmelig finkornet informasjon og skille på nivå med bygninger og attraksjoner. Informasjon om disse attraksjonene må være lagret sammen med posisjonen de har. Informasjonen kan også være lagret med tidsstempel der man kan hente ut informasjon relevant til tidspunktet bildet ble tatt på (for eksempel dagens program for en kino).

Videre er det to muligheter: Turistene kan selv kjøre CaimTool på sin datamaskin, eller aksessere en sentralt kjørt CaimTool over internett, og laste inn bildene fra sitt digitale kamera. Ved å koble seg opp mot ContextManager som turistbyrået kjører, kan de nå få hentet informasjon om hva de har tatt bilde av, og lagre det i bildefilen. De vil få velge hvilke bilder de ønsker informasjon om (ikke alle bilder er relevante å søke opp informasjon om), og de vil også kunne velge hvor vidt de vil lagre den informasjonen som blir funnet sammen med bildet.

Det andre alternativet er å bruke webgrensesnittet fra en mobiltelefon. Man vil da normalt ikke kunne bla gjennom bildefilene i minnet på mobiltelefonen, og må bruke grensesnittet til å laste opp de bildene man ønsker informasjon om først.

[15] http://en.wikipedia.org/wiki/Global_Positioning_System#Accuracy_and_error_sources

Scenarioet forutsetter at det digitale kameraet har en GPS-enhet som lagrer GPS-informasjon sammen med bildene som blir tatt.

En av utfordringene i et slik scenario, er at det er vanskelig å få god treffsikkerhet på hva man faktisk har tatt bilde av. GPS-enheten vil normalt ikke registrere kompassretning, og man kan derfor få informasjon om noe som er i nærheten, men faktisk befinner seg bak turisten og dermed ikke er med på bildet. En løsning på dette er å kombinere tjenesten med funksjonalitet lignende den i SmartKikkert (se seksjon 6.3) der også retning og blenderåpningen blir tatt med i beregningen, og et oppslag blir gjort i en POI[16]-database for å anslå hva turisten faktisk har tatt bilde av.

5.2 Værdata - WeatherContext

For å demonstrere funksjonaliteten i aCCoFrame-rammeverket, er det laget en prototyp på en kontekstkilde som henter informasjon om været fra statistikk som IFI ved UiT har ført siden 1994[17]. I denne statistikken finnes bare informasjon om været i Tromsø, så bare tidsstempelet til bildet er tatt i bruk for å slå opp værdata. I praksis burde man også brukt GPS informasjon for å få værdata på det stedet og tidspunktet bildet faktisk ble tatt, men da trenger man tilgang til en database der denne informasjonen er lagret.

Tjenesten passer naturligvis best til bilder tatt utendørs, som landskapsbilder og naturbilder.

En mulig videre bruk av tjenesten er som ledd i en analyse av været på bilder: bildene kan nå først få lagret informasjon om de faktiske værforhold i motivet på bildet. Dette kan så brukes som fasit for å finne en bildegjenkjenningsalgoritme som kan bruke fargehistogram til å gjenkjenne været på bildet.

[16] *Point Of Interest*
[17] *Værtjenesten ved IFI* http://weather.cs.uit.no/

6 Muligheter fremover

Verktøyene som er utviklet under dette arbeidet - CaimTool, ContextManager og ContextSource - er ment som en nødvendig basis for videre arbeid med Context-Aware Image Management (CAIM).

6.1 Nye kontekstkilder (ContextSource)

Å lage en ny kontekstkilde forutsetter ikke nødvendigvis at man må opprette en stor database med kontekstinformasjon. Man kan i stedet bruke eksisterende kilder fra internett, som Wikipedia eller Google Maps.

6.1.1 GeoContext

GeoContext kan bruke API-et til Google Maps til å slå opp posisjonen fra EXIF-headeren i et bilde, og få tilbake mulige stedsnavn. På denne måten kan man få annotert bildene med geografiske navn på region, by m.m.

6.1.2 WikiContext

En kontekstkilde kan bruke tidsstempelet fra EXIF-informasjonen til å slå opp informasjon fra Wikipedia, som har en egen side for datoer der det har skjedd noe av betydning.

6.2 ImagePools + SIFT

Ved hjelp av CaimTool kan man lettvint legge til kontekstinformasjon i EXIFheaderen til bilder, og kjøre batch-jobber der dette blir gjort for mange bilder samtidig. En ImagePool er en bildedatabase som har lagret metadataen fra EXIF-headeren for hvert bilde. Man kan så gjøre spørringer til ImagePool der man bruker kontekst som input. Eksempel på slik kontekstinformasjon er posisjon (lagret av en GPS enhet), vindstyrke (lagret av WeatherContext beskrevet i seksjon 5.2) eller kontekst lagret av en annen kontekstkilde, som for eksempel geografisk navn fra GeoContext.

Videre kan det opprettes et distribuert system av ImagePools og en sentral PoolManager server, etter samme modell som ContextSource og ContextManager. CaimTool kan utvides til å støtte ImagePools, og man får nå muligheten til å opprette en ImagePool fra sine lokale bilder, og se / søke gjennom bilder fra andre ImagePools som er tilgjengelige. Man kan kombinere utvalget fra ImagePools og ContextSources, noe som vil være en god utvidelse av funksjonaliteten til CaimTool.

SIFT[18] er en bildegjenkjennigsalgoritme som er god på å sammenligne et bilde mot kjente bilder i en database. Hovedproblemet til SIFT er at den er tung og treg[18], og det vil som oftest ta for lang tid å få svar fra algoritmen. Den er likevel interessant, siden den har en veldig god treffsikkerhet sammenlignet med andre algoritmer.

Om de kjente bildene SIFT skal bruke som sammenligningsgrunnlag var lagt inn i en (eller flere) ImagePools, kunne man bruke kontekstinformasjonen der

[18] *Scale-Invariant Feature Transform*

sammen med kontekstinformasjonen fra bildet man har tatt, til å begrense utvalget av bilder SIFT må sammenligne mot. For hvert bilde man kan fjerne fra utvalget vil responstiden fra algoritmen synke lineært, og man kunne nærme seg en kjøretid som var praktisk anvendbar.

6.3 Mobil bruk: SmartKikkert

Argos-rammeverket er ikke rettet mot mobil bruk, og det finnes ingen lettversjon av Argos som kan kjøre på mobiltelefoner. Siden aCCoFrame er bygget på Argos, kan man ikke uten videre ta det i bruk på mobiltelefoner. Den foreløpige veien er å bruke webgrensesnittet som CaimTool tilbyr, men dette kan for eksempel ikke integreres automatisk mot kameraet på telefonen, og man må først finne bildet i minnet på telefonen, laste det opp til CaimTool, bruke grensesnittet til å finne kontekstinformasjon, for til slutt å laste ned igjen bildet med den integrerte kontekstinformasjonen. En slik prosedyre er kanskje noe vi godkjenner fra en stasjonær pc, men det samsvarer ikke med forventningene til mobil integrasjon.

SmartKikkert[19] er en prototype på en mobil applikasjon som integrerer kameraet i mobiltelefonen med en GPS/kompass-enhet. Basert på kameraparametere, kompassretningen og gps-posisjon sender den så en forespørsel til en server som finner POIer[19] innenfor det arealet som er motivet i bildet. Man kan deretter velge en POI og få opp informasjon om hvert punkt.

SmartKikkert kunne derfor være et godt utgangspunkt for en mobil komponent til aCCoFrame. Den integrerer seg med kamera, henter ut metainformasjon, og kommuniserer med eksterne servere. Funksjonaliteten som finner POI kan reimplementeres som en kontekstkilde og inngå som en del av aCCoFrame, og SmartKikkert kan da kommunisere med aCCoFrame på samme måte som CaimTool gjør det.

For å få lagret kontekstinformasjonen i bildet må det implementeres en ExifWriter for JavaME [20] som SmartKikkert kan benytte.

6.4 Aggregering

Internett er i dag den største kilden til informasjon, og potensielt sett kan det implementeres mange ulike kontekstkilder som henter informasjon fra internett. Hovedproblemet med denne informasjonen er at den ikke er semantisk bygget opp. Derfor er man stort sett begrenset til søk etter frie tekststrenger, og det er vanskelig å vurdere kvaliteten på informasjonen man finner. Ved bruk av aCCoFrame har man i utgangspunktet bilder som ikke er annotert med tekststrenger egnet for søk på internett, og muligheten til å tappe av denne store informasjonskilden er derfor liten.

Spesialkomponenter som GeoContext og WikiContext beskrevet ovenfor (seksjon 6.1) har begrenset funksjonalitet, men om man hadde mange slike komponenter kunne man aggregere resultatene fra dem, og bruke dette som input til ett nytt søk. Som eksempel kunne man fra GPS-koordinater først få navn på geografiske lokasjoner, og deretter bruke disse som input i et tekstlig søk hos f.eks. Google eller Wikipedia.

aCCoFrame kan derfor utvides med en komponent, eller en systemtjeneste i Argos, som gir mulighet til å aggregere informasjon fra flere komponenter, og bruke dette som input igjen til en ny komponent. Med komponent tenkes det

[19] *Point Of Interest*

her da spesielt på ContextSource komponenter i aCCoFrame. Etter hvert som tilbudet av kontekstkilder i aCCoFrame utvides vil nytten av aggregering bli enda mer aktuell.

Det vil sannsynligvis være nyttig med etterkontroll av informasjonen som blir hentet fra internett. Aggregeringskomponenten bør derfor tilby et eget grensesnitt der den aggregerte informasjonen blir vist før man velger ut den informasjonen man vil skal lagres sammen med bildet. Det kunne også være en mulighet å prøve å 'lære opp' aggregeringskomponenten ved å gi tilbakemelding på hvor relevant informasjonen som ble funnet er.

Referanser

[1] Context-Aware Image Management (CAIM http://caim.uib.no)

[2] Arne-Henrik Tøndersen. 2006. Image retrieval based on location and time. Masteroppgave i informatikk. Mai 2006. Institutt for Informatikk, Det matematisk-naturvitenskapelige fakultet, Universitetet i Tromsø.

[3] Kurt Jøran Nyland. 2006. Bildesamlinger og kontekst. Masteroppgave i informatikk. Mai 2006. Institutt for Informatikk, Det matematisk-naturvitenskapelige fakultet, Universitetet i Tromsø.

[4] Anne Staurland Aarbakke. 2007. *M2S and CAIR: Image based information retrieval in mobile environments.* Masteroppgave i informatikk. Mai 2007. Institutt for Informatikk, Det matematisk-naturvitenskapelige fakultet, Universitetet i Tromsø.

[5] Argos http://argos.cs.uit.no

[6] Centre for Digital Video Processing, Dublin City University (CDVP http://www.cdvp.dcu.ie/)

[7] Mootools, a compact, modular, Object-Oriented JavaScript framework (mootools http://mootools.net)

[8] Exchangable Image File Format (EXIF http://www.exif.org/Exif2-2.PDF)

[9] Asynchronous Javascript and XML (AJAX http://en.wikipedia.org/wiki/Ajax_(programming)

[10] Java Server Pages (JSP http://java.sun.com/products/jsp/)

[11] Java Managable eXtensions (JMX http://java.sun.com/javase/technologies/core/mntr-mgmt/javamanagement/)

[12] APMS (nå Argos) Wiki (apms-wiki http://ablab10.cs.uit.no:8008/apms-wiki/index.php/Main_Page#History)

[13] eXtensibel Markup Language (XML http://www.w3.org/XML/)

[14] Document Type Definition (DTD http://www.w3.org/TR/REC-xml/#dt-doctype)

[15] Jetty - en open-source webserver implementert i Java (Jetty http://www.mortbay.org/)

[16] Schürmann, A, Aarbakke, A S, Egeland, E, Evjemo, B and Akselsen, S. 2006. *Let a picture initiate the dialog! A mobile multimedia service for tourists.* Fornebu, Telenor Research & Innovation. Report R&I N 33/2006.

[17] Joint Photographic Experts Group (JPEG http://www.jpeg.org/)

[18] Thomas Bakken. *An evaluation of the SIFT algorithm for CBIR*. R&I Forskningsnotat N 30/2007.

[19] Thomas Bakken. *Smart Binocular – a context aware mobile application*. R&I Forskningsnotat N 29/2007. Bedrift KONFIDENSIELT.

[20] Java Plattform, Micro Edition (JavaME http://java.sun.com/javame/index.jsp)

[21] Subversion, a Version Control System (Subversion http://subversion.tigris.org)

[22] The Trac Project (Trac http://trac.edgewall.org)

[23] ProgrammeringsManual for Argos (manual.pdf http://argos.cs.uit.no/doc/Programming%20manual.pdf)

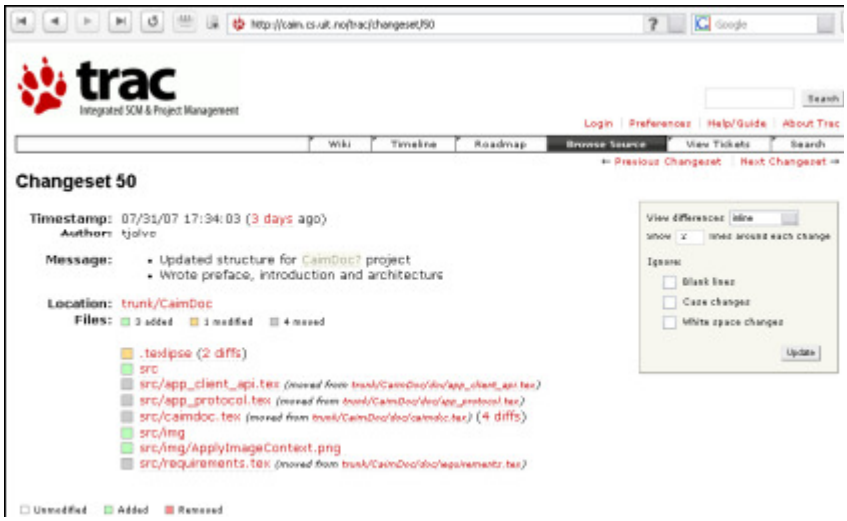
[24] eXtensible Hypertext Markup Language (XHTML http://www.w3.org/TR/xhtml11/)

[25] Cascading Style Sheet (CSS http://www.w3.org/Style/CSS/)

[26] Extensible Stylesheet Language (XSL http://www.w3.org/TR/xsl/)

Tillegg A Utviklingsmiljø for CAIM

Det er satt opp en server for CAIM-prosjektet på http://caim.cs.uit.no/trac. Denne serveren kjører to tjenester for de som deltar i CAIM: Subversion og Trac.



Figur 2: Trac for CAIM-prosjektet

CAIM Subversion: svn://caim.cs.uit.no

CAIM Trac: http://caim.cs.uit.no/trac

Subversion [21] er et VCS[20] som holder rede på alle versjoner av et dokument, for eksempel kildekode for et prosjekt. Det støtter også funksjoner for å spleise to ulike versjoner av et dokument sammen (dersom for eksempel to personer har gjort ulike endringer på et dokument samtidig).

Trac [22] er et webgrensesnitt mot subversion, og gir god oversikt over de ulike revisjonene av dokumentene til et prosjekt. I tillegg tilbyr det blant annet en Wiki og et Ticket-system der man kan rapportere feil m.m.

[20] *Version Control System*

Tillegg B Programmeringsmanual

For å ta aCCoFrame i bruk, trenger man en sentral server som fungerer som en proxy mellom brukerne og kontekstkilden(e), og denne kjører ContextManagerkomponenten. I tillegg må det finnes kontekstkilder som er registrerte hos ContextManager. En kontekstkilde er en komponent av typen ContextSource som kan finne informasjon basert på tid, dato, gps-posisjon eller en kombinasjon av disse.

Man har valget mellom å bygge opp sin egen database med informasjon registrert på tid, dato og gps-posisjon, eller å lage en information-retrieval algoritme som søker etter informasjon på internett. Denne kunne for eksempel søke på kjente nettsteder som Wikipedia eller Google Maps.

Å utvikle tjenester for aCCoFrame skiller seg ikke så mye fra å utvikle tjenester for Argos. En god start er derfor å følge manualene for Argos, og få en følelse av begrepene konteiner og komponent. aCCoFrame utnytter på langt nær alt som Argos tilbyr; nedenfor følger derfor en forklaring av hvordan Argos brukes av aCCoFrame.

B.1 Bruk av Argos

All kode for aCCoFrame er delt opp i komponenter som blir kjørt i en konteiner i Argos. Komponentene kommuniserer med hverandre ved hjelp av Notifications, som er en del av JMX.

1. Notifications fungerer slik at en komponent kan 'abbonnere' på Notifications fra en annen komponene. Når en komponent sender ut Notification, blir den mottatt av alle abbonnentene. Når en komponent mottar en Notification, blir den sendt til den metoden i komponenten som er annotert som @NotificationHandler. Bruken av Notifications er mer utførlig forklart i manualen til Argos [23].
2. CaimTool og ContextManager-komponentene sender og behandler bare en Notification dersom den er av en type som er definert i NotificationType-klassen[21]. Man kan selvsagt benytte egendefinerte Notifications så lenge de ikke sendes til en av disse to komponentene.
3. En ContextSource må benytte @Component("!!TwoWay") slik at CaimTool klienter kan sette opp to-veis kommunikasjon når de etterspør kontekst for bilder.
4. CaimTool, ContextManager og ContextSource må følge API for kommunikasjon slik det er spesifisert nedenfor (Tillegg C).

B.2 Web-interface

Argos tilbyr komponenter muligheten til å kjøre en webserver. Dersom komponenten er konfigurert med dependency *!!Jetty6,* vil Jetty automatisk bli startet opp av Argos når komponenten lastes, og alt som ligger i rotmappen

[21] *NotificationType*
http://caim.cs.uit.no/trac/browser/trunk/CaimTool/src/uit/caim/NotificationType.java

'web' til komponenten vil bli kopiert til Jetty sin public-html mappe. Les mer om konfigurering av komponenter i programmeringsmanualen til Argos [23].

CaimTool-komponenten står for grensesnittet til aCCoFrame, og dette er et webinterface servert av den innebygde Jetty-serveren i Argos. Websidene er skrevet i JSP[22] og produserer XHTML[23]-output stilet av CSS[24]. For å holde JSP-filene renest mulig, benytter de seg av klassene i uit.caim.webbeans der mye av prosesseringen av input og output til websidene foregår. Videre er det en del AJAX-funksjonalitet på websidene som bruker biblioteket mootools.js[6].

Det meste av stilen kan forandres på ved å forandre på stilarket style.css[25], men en del CSS er også definert direkte i JSP-filene, så vær oppmerksom på dette. Når man ber om kontekstinformasjon for bilder, vil denne komme formatert som XML fra kontekstkilden. Kontekstkilden har mulighet til å sende ved et XSL[26]-stilark som forteller hvordan informasjonen skal presenteres. Uten et slikt XSL-stilark vil informasjonen bli plassert i en tabell, der det for hvert bilde kan være opptil ni felt med informasjon. Stilen på denne tabellen er inkludert i headeren til contextResult.jsp[27] og kan endres der.

B.3 Eksempel: Legge til ny ContextSource

En ContextSource tilbyr informasjon basert på metadata fra et bilde. Typen metadata må være kjent av CaimTool. Kjente typer er definert i ContextType-klassen[28], og er foreløpig ulike typer dato- og gps-felt fra EXIF-headeren. Dersom man vil benytte annen metadata i kontekstkilden, må man også oppdatere ContextType-klassen.

Meldingstypene er definert i caim-message.DTD. Når man sender en Notification, må man spesifisere meldingstypen til å være fra NotificationType-klassen. Typene der svarer til de ulike meldingene i caim-message.DTD.

Når kontekstkilden registrerer seg hos ContextManager, må den sende en ContextRegister melding. I denne må det spesifiseres hvilken metadata kontekstkilden trenger fra bildet for å kunne søke etter kontekstinformasjon. For en kontekstkilde som for eksempel tilbyr værdata, vil det da være naturlig å be om et dato-felt og gps-posisjon fra bildet.

Kontekstkilden må også håndtere ulike innkommende Notifications. Dette gjør man ved å annotere en metode som @NotificationHandler, se Figur 3.

En ContextSource benytter metoden annotert med @Execute til å:

a) sette opp en to-veis kommunikasjon mot ContextManager ved hjelp av !!twoWayNotifications systemtjenesten i Argos, og

b) registrere seg som en ContextSource hos ContextManager. Det finnes alternative måter å gjøre dette på.

Kravene for at ContextSource er aktiv i systemet og kan benyttes av CaimTool, er følgende:

---

[22] *Java Server Pages*
[23] *eXtensible Hypertext Markup Language*
[24] *Cascading Style Sheet*
[25] *style.css* http://caim.cs.uit.no/trac/browser/trunk/CaimTool/web/style/style.css
[26] *Extensible Stylesheet Language*
[27] *contextResult.jsp* http://caim.cs.uit.no/trac/browser/trunk/CaimTool/web/contextResult.jsp
[28] *ContextType*
http://caim.cs.uit.no/trac/browser/trunk/CaimTool/src/uit/caim/context/ContextType.java

1. ContextSource må sette opp kommunikasjon mot ContextManager slik at den får sendt en NEWCS notification. Denne må inneholde en melding som følger spesifikasjonene for et context-register XML dokument.
2. Den må lytte på Notifications fra ContextManager slik at den mottar og kan svare på meldinger derfra.
3. Den må ta inn @Component("!!TwoWay") slik at CaimTool klienter kan sette opp to-veis kommunikasjon ved behov.

```
@Impact(Impact.ACTION)
@RemoveInstrumentation
@NotificationHandler
public void handleNotification(Notification not) throws IOException {
    System.out.println("WeatherContext: Got notification: "
            + not.getMessage() + " "
            + not.getType());

    nt = NotificationType.valueOf(not.getType());
    switch (nt) {
    case CONTEXTDATA: {
        Notification rep = new Notification(nt.CONTEXTPARAMETERS.toString(),
                "",0,"data");
        proxy.send(rep);
    }
    case CSOK: {
        if (not.getMessage().equals("ok")) registered = true;
    }
    case ALIVE: {
        Notification alive = new Notification(nt.ALIVE.toString(),
                "",0,hostname.toString());
        proxy.send(alive);
    }
    case CONTEXTREQUEST: {
        getContextData(not.getMessage());
    }
    default:
        break;
    }
}
```

Figur 3: *@Noti ctionHandler metoden for WeatherContext. En switch-case blir gjort på verdien til enumeratoren NotificationType for å håndtere de ulike meldingene. En ContextSource må håndtere de innkommende meldingene CONTEXDATA, CONTEXREQUEST, CSOK og ALIVE.*

Tillegg C API

API for aCCoFramework er hovedsaklig definert ut i fra et DTD-dokument (se *Figur 4*) som beskriver de ulike meldingene komponentene kan / må sende seg imellom. Under følger en kort beskrivelse av disse, samt sekvensdiagrammer som viser flyten av meldinger i ulike situasjoner.

C.1 Meldinger

Nedenfor følger en tekstlig beskrivelse av de viktigste meldingstypen i aCCoFrame. De er formelt beskrevet i caim-message.DTD, se *Figur 4*. Figur 1 viser en del av meldingsflyten mellom komponentene.

context-register: sendes av en kontekstkilde til ContextManager for å registrere seg i systemet. Den inneholder navnet på kontekstkilden, adressen til konteineren kontekstkilden kjører i, en tekstlig beskrivelse av kontekstkilden, og en eller flere kontekst-typer fra ContextType-klassen.

context-request: sendes av CaimTool til ContextManager i det en bruker velger kontekst-type fra grensesnittet og klikker på knappen ”Find Context-Sources,” og inneholder de kontekst-typene brukeren valgte fra grensesnittet.

context-response: sendes av ContextManager til CaimTool som svar på contextrequest, og inneholder navn, adresse og beskrivelse (som kontekstkilden registrerte hos ContextManager).

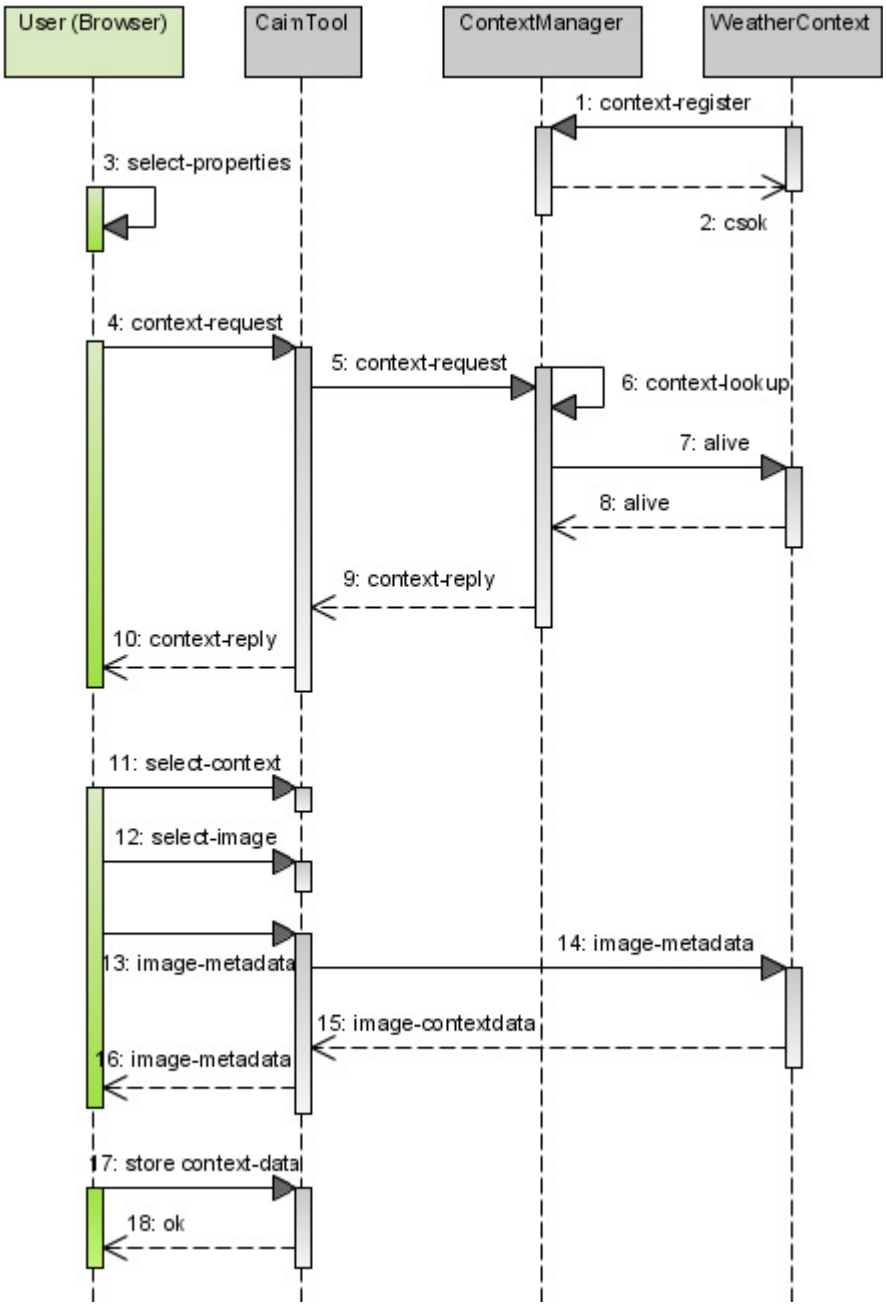
image-metadata: sendes av CaimTool til en kontekstkilde, og innholder ett eller flere 'image' element, som hver har navnet på bildet som identifikator, og en eller flere kontekst-typer som skal benyttes av kontekstkilden for å finne kontekstinformasjon.

image-contextdata: sendes av en kontekstkilde til CaimTool, og inneholder url til et XSL-skjema som benyttes av web-grensesnittet for å forstå og presentere kontekstinformasjonen, og ett eller flere 'data' element, som hver har navnet på bildet som identifikator, og et base64-encoda XML-dokument som inneholder de faktiske kontekst-dataene, og som validerer mot det medfølgende XSL-skjemaet.

```
<!-- DTD for CAIM-messages. Usually sent as JMX Notifications.
See http://caim.cs.uit.no -->
<!ELEMENT caim-message (context-register|context-request|
        context-response|image-metadata|image-contextdata)>

<!-- register a new context source -->
<!ELEMENT context-register (address, description, exifparam+)>

<!-- request context sources based on the parameters given -->
<!ELEMENT context-request (name,image+)>

<!   response on a context request   >
<!ELEMENT context-response (address*)>

<!-- image in context-request -->
<!ELEMENT image (exifparam+)>

<!-- URL -->
<!ELEMENT address (#PCDATA)>
<!-- any textual information the context source wants to provide -->
<!ELEMENT description (#PCDATA)>
<!-- DATE, TIME, GPS, TAGWORD -->
<!ELEMENT exifparam (#PCDATA)>

<!-- image metadata required by the context-source -->
<!ELEMENT image-metadata (date?, time?, gps?, tagword*)>

<!-- image context-data from a context source -->
<!ELEMENT image-contextdata (dtd, data)>
<!-- URL to an DTD describing the context-data XML -->
<!ELEMENT dtd (#PCDATA)>
<!-- Contents interpreted as Base-64 encoded. -->
<!ELEMENT data (#PCDATA)>
<!-- data must decode to a XML document valid to the DTD -->

<!ELEMENT date (#PCDATA)>
<!ELEMENT time (#PCDATA)>
<!ELEMENT gps (#PCDATA)>
<!ELEMENT tagword (#PCDATA)>
```

Figur 4: caim-message.dtd: Document Type Definition for alle XMLmeldinger i Notifications mellom komponentene.

C.2 Sekvensdiagram



Figur 5: Sekvensdiagram for aCCoFrame. Kommunikasjonen mellom User og CaimTool blir håndtert av AJAX og Jetty-webserveren, og inngår ikke som notifications-meldinger i systemet, men brukeren gjør handlinger i grensesnittet som initierer slike meldinger til/fra CaimTool (4, 10, 13, 16). En del hendelser skjer bare i grensesnittet mellom User og CaimTool (11, 12, 17, 18). Noen hendingssekvenser er interne (3, 6). Hendelsene som svarer til faktiske Notifications sendt mellom komponentene i aCCoFram er 1, 2, 5, 7, 8, 9, 14 og 15.