Utredning av ny IT-arkitektur for samlingsdatabasene utviklet og driftet i regi av MUSIT

Innhold

0. Sammendrag og konklusjon

Dagens IT-arkitektur i MUSIT ble utviklet gjennom Dokumentasjonsprosjektet og Museumsprosjektet på 1990-tallet. De teknologiske valgene som ble tatt den gang, har gitt robuste og stabile samlingsforvaltingssystem og gode verktøy for forskning, formidling og forvaltning i universitetsmuseene.

Det er gruppens anbefaling at MUSIT går fra dagens to-lags IT-arkitektur med de begrensninger det medfører, og over til en moderne og tjenesteorientert flerlagsarkitektur som skalerer bedre i forhold til de krav og behov som museene har i dag. Det bør utvikles nye, funksjonelle IKT-løsninger som i liten grad bør ta hensyn til den lange historikken som samlet sett gjør dagens løsninger mer ressurskrevende enn nødvendig.

Som et første ledd i arbeidet med å (videre)utvikle, ev. anskaffe, en eller flere nye IT-løsninger for forvaltning av museumsobjekter, anbefaler gruppen at man gjør et forarbeid med å beskrive arbeids- , informasjons- og dataflyten ved museene (virksomhetsarkitekturen). I den forbindelse bør man også se på om det er databaser eller applikasjoner som i dag vedlikeholdes i regi av MUSIT, man bør gjøre seg av med eller la andre overta. Et eksempel på et system som vedlikeholdes av andre, er Intrasis som nå brukes til feltdokumentasjon ved arkeologiske undersøkelser, og således dekker ett bestemt behov innenfor den felles løsningen som MUSIT har ansvar for. Topografisk arkiv og fotobasen, i alle fall den delen som går på det å ta bilder, manipulere bildeformat og lagre bildene, er applikasjoner som ikke nødvendigvis må være del av MUSITs portefølje av IKT-løsninger i fremtiden.

Anbefalingene til gruppen kan summeres opp i følgende hovedpunkter:

- Framtidens IT-løsning skal ikke være ett, monolittisk system som inneholder alle moduler og komponenter
- IT-løsningene skal bygges på en tjenesteorientert flerlagsarkitektur (SOA) og bør i hovedsak være Web-baserte
- Gå bort fra mange databaseløsninger med samme eller tilsvarende funksjonalitet til en mer ensartet tjenesteorientert IT-arkitektur
- Målet bør være samme type funksjonalitet – samme type IT-løsning – uavhengig av fagområde, for eksempel felles standardiserte løsninger for magasin, utlån/innlån, multimedia, fotografier, arkiv, etc. Dette kan innebære at museene må tilpasse sine eksisterende rutiner.
- En .NET-basert tilnærming anbefales framfor en Java-basert der dette er hensiktsmessig

Et estimat på tidsbruk summeres til ca. 14 årsverk. Da er det beregnet ett månedsverk ved hvert museum for å gjennomføre en virksomhetsanalyse. Virksomhetsanalysen vil dessuten trenge ressurser til en prosessleder og 2-3 møter.

Dersom hele prosessen skal gjennomføres med eksterne konsulenter, vil det kreve over NOK 22 millioner. Dersom hele prosessen gjøres med egne tilsatte, vil et budsjett på det halve rekke langt. Den beste måten å gjennomføre prosessen på, vil være en kombinasjon av disse alternativene, der MUSIT-tilsatte, andre tilsatte ved universitetenes IT-avdelinger og eksterne konsulenter gjennomfører ulike deloppgaver.

Vi har ikke tallfestet driftsutgiftene ved overgang til ny arkitektur. Det er rimelig å anta at de vil være i samme størrelsesorden som for dagens løsning.

Følgende tidsplan vil gi en gjennomføring i løpet av 4,5 år. Noen av aktivitetene må gjøres i rekkefølge, mens andre kan gjennomføres samtidig:

Aktivitet	**Årsverk**	**Halvår**								
		1	2	3	4	5	6	7	8	9
Virksomhetsanalyse ved museene	0,5	x	x	x						
Installere og drifte mellomlagsapplikasjonen	1	x	x							
Konvertere dagens Oracle-database og applikasjoner til Unicode	1	x	x							
Implementere og teste ny mellomlagsprogramvare	1,5			x	x	x				
Flytte databasenær kode ut i mellomlaget	1				x	x				
Migrere naturhistorieapplikasjonene	2						x	x	x	x
Migrere arkeologi	2						x	x	x	x
Migrere øvrige samlinger (etno./ kulturhistorie)	2						x	x	x	x
Migrere databaser som ikke er MUSIT-baser	3	x	x	x	x	x	x	x	x	x

1. Innledning

Styret for MUSIT bestemte i sitt møte den 20. september 2012 å nedsette en prosjektgruppe som skulle utrede ny IT-arkitektur for MUSIT. Prosjektgruppen fikk følgende mandat (jf. V-sak 06/3-12):

1. Det nedsettes en prosjektgruppe med følgende sammensetning:
    - to personer fra DUG
    - to personer fra USIT (en fra Seksjon for system og applikasjonsdrift (SAPP) og en fra Seksjon for utvikling av nasjonale informasjonssystemer (SUN))
    - én person fra NTNU IT
    - to personer med IT-faglig kompetanse og/eller erfaring fra museene. Disse personene bør dekke henholdsvis kultur- og naturhistorie.
2. Prosjektgruppen får følgende mandat:
    A. Beskrive dagens system og peke på utfordringer knyttet til framtidige behov og ønsker i museene
    B. Utrede ulike løsninger for hvordan MUSIT sin framtidige IT-arkitektur bør være når det gjelder:
        i. Databaser og databaseteknologi
        ii. Utviklingsrammeverk for klient-server-teknologi, dvs. utviklingsverktøy for både databasenær kode, inkl. autentisering, og brukergrensesnitt
        iii. Tilgjengeliggjøring for web, mobil og nettbrett i egne og eksterne portaler
    C. Vurdere om det finnes andre system, spesielt i Norden og Europa for øvrig, som kan dekke de behovene museene har for samlingsforvaltning og tilgjengeliggjøring
    D. Beregne hvilke investeringer og årlige ressurser de ulike løsningene vil kreve
    E. Anbefale én løsning for ny IT-arkitektur og foreslå tidsplan for implementering
3. Koordineringsgruppene fungerer som referansegrupper for prosjektgruppen, og konklusjonene og forslagene prosjektgruppen kommer fram til, legges fram for koordineringsgruppene før de oversendes styret.
4. DUG fungerer som sekretariat for prosjektgruppen som selv velger sin leder.
5. Prosjektgruppen skal ferdigstille utredningen før 1. februar 2013.

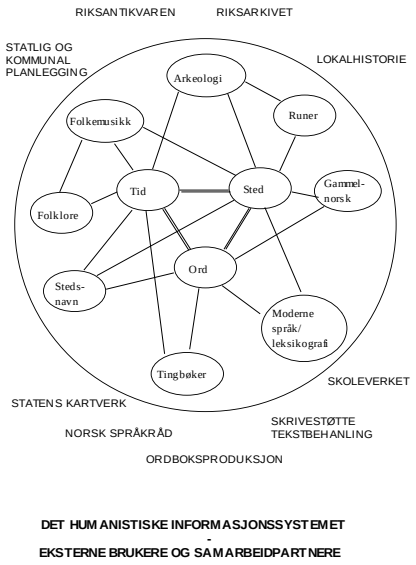
Følgende personer ble oppnevnt til prosjektgruppen som på første møte konstituerte seg selv:

- Espen Uleberg (KHM, leder)
- Einar Timdal (NHM)
- Carl-Fredrik Sørensen (NTNU IT)
- Frank Solem (SAPP/USIT)
- Geir Vangen (SUN/USIT)
- Jarle Stabell (DUG/MUSIT)
- Jarle Ebeling (DUG/MUSIT)

Prosjektgruppen har hatt to møter i prosjektperioden. I tillegg hadde Einar Timdal, Espen Uleberg og Jarle Ebeling møte med Eva Vedin og Julie Melin ved Statens Historiska Museum, og Fredrik Ronquist ved Riksmuseet i Stockholm for å få informasjon om utviklingen på tilsvarende områder innenfor svensk museumsverden. Espen Uleberg og Jarle Stabell snakket dessuten med Ulf Bodin i Stockholm som arbeider for Kutlur-IT og som har vært med å utvikle feltdokumentasjonssystemet Intrasis og samlingsforvaltningssystemet som brukes av Statens historiske museum i Stockholm.

2. Bakgrunn og rammer

Dagens IT-arkitektur i MUSIT ble utviklet gjennom Dokumentasjonsprosjektet (Dokpro) og Museumsprosjektet på 1990-tallet. Dokumentasjonsprosjektet arbeidet med forskningsdatabaser for samlingsavdelingene ved de Historisk-filosofiske fakultetene ved universitetene i Norge. Man utarbeidet løsninger for hver enkelt samling, men målet var et humanistisk informasjonssystem, hvor det skulle være mulig å søke på tvers av samlingene (se Figur 1.). Dokumentasjonsprosjektet utviklet en løsning for topografisk arkiv i samarbeid med Historisk Museum, UiB, en løsning for fotografier (mediebase) i samarbeid med Tromsø Museum – Universitetsmuseet, og løsninger for runer, numismatikk og arkeologiske gjenstander i samarbeid med IAKN, UiO. Disse løsningene er senere utvidet og tilpasset behov slik at de nå brukes ved de samarbeidende museene i MUSIT.



Figur 1. Det humanistiske informasjonssystem. (Holmen & Uleberg 1996[1])

Da Dokumentasjonsprosjektet ble avsluttet, var de kulturhistoriske museene skilt ut fra HF-fakultetene. De delene av Dokpro som stadig hørte inn under HF-fakultetene ble videreført i Enhet for Digital Dokumentasjon (EDD). De kulturhistoriske universitetsmuseene fortsatte samarbeidet i Museumsprosjektet som også inkluderte de naturhistoriske samlingene.

Før Museumsprosjektet hadde de naturhistoriske samlingene samarbeidet i UNADOK. Det nasjonale fakultetsmøte for realfag var oppdragsgiver for UNADOK som ble igangsatt som et forprosjekt i 1994. UNADOK-prosjektet involverte de botaniske, geologiske, paleontologiske og zoologiske museene/avdelingene ved universitetene i Bergen, Oslo, Tromsø og Trondheim. Ved de naturhistoriske museene var det utviklet en rekke ulike databaseløsninger for større og mindre samlinger. Mange av disse basene er utviklet og fremdeles vedlikeholdt av enkeltpersoner. Det er en stadig pågående prosess å samle og konvertere dette mangfoldet av databaser til færre og felles løsninger.

I MUSIT har en i den første fasen lagt vekt på å videreutvikle, ferdigstille og ta i bruk løsninger. Senere er det lagt vekt på at applikasjonene skal utvikles i samarbeid mellom museene. En har også lagt mye arbeid i å få bedre sammenheng mellom eksisterende moduler,

[1] Holmen, Jon & Espen Uleberg 1996. The National Documentation Project of Norway –the Archaeological sub-project. In: Kamermans, H.; Fennema, K. (eds.), Interfacing the Past: Computer Applications and Quantitative Methods in Archaeology CAA95 vol. 1. Analecta Praehistorica Leidensia 28.

som mellom mediebasen og gjenstandsbasene. En legger også stadig mer vekt på å utvikle ny funksjonalitet som henger sammen med eksisterende løsninger istedenfor å arbeide med frittstående enkeltapplikasjoner. Det nyeste eksempelet på dette er utviklingen av en konserveringsmodul innenfor arkeologibasen.

Et internasjonalt rammeverk for utvikling av databaseløsninger for museer er CIDOC Conceptual Reference Model (CRM). CIDOC CRM er en ISO-standard for definisjoner og en formell struktur for å beskrive implisitte og eksplisitte begrep og sammenhenger i dokumentasjon av kulturarv. Det norske miljøet i EDD/Museumsprosjektet og MUSIT er ledende i Norge og viktige aktører internasjonalt når det gjelder implementering av CIDOC-CRM. MUSITs etnografibase er en av de største og mest gjennomførte CIDOC CRM basene.

Det har fram til i dag vært et bevisst valg i MUSIT å bruke alle ressursene i Drifts- og utviklingsgruppen (DUG) på å få utviklet løsninger ved bruk av dagens rammeverk/arkitektur, slik at flest mulig samlinger ved det enkelte museum kunne forvaltes av MUSIT.[2]

I de siste par årene har det vokst fram en erkjennelse av at MUSITs IT-arkitektur må fornyes. Det er behov for å hente ut og kombinere data på måter som ikke er mulig med dagens løsning. En annen grunn er praktiske hensyn som at samiske og andre tegnsett ikke støttes av den løsningen en har i dag. Det er nødvendig å tenke grundig gjennom IT-arkitekturen, for å kunne velge løsninger som tar hensyn til statens krav om helhet, samarbeid og felles tiltak når det gjelder de store IKT-investeringene som gjøres. I stortingsmelding nr. 19 (2008-2009) *Ei forvaltning for demokrati og fellesskap*[3], kommer Fornyings- og administrasjonsdepartementet med en rekke krav og anbefalinger om arkitekturprinsipper som tjenesteorientering og gjenbruk, interoperabilitet, tilgang, trygghet, åpenhet, fleksibilitet og skalerbarhet. Innføring av ny IT-arkitektur hos MUSIT er en stor oppgave som vil legge beslag på det meste av organisasjonens utviklerkapasitet i lang tid. Det er derfor nødvendig å ha best mulig grunnlag når en legger opp en strategi som skal være grunnlag for utvikling av IT-løsninger som skal velges og hvordan de nye løsningene skal implementeres.

Mandatets 2A: Beskrive dagens system og peke på utfordringer knyttet til framtidige behov og ønsker i museene

Dagens system og framtidige behov og ønsker i museene

I stortingsmelding nr. 19 sies det at statlige virksomheter skal bruke følgende prinsipp i planlegging av nye IKT-løsninger eller ved ombygging av eksisterende løsninger:

- *Tenesteorientering:* IKT-system skal byggjast opp som ei samling avgrensa delsystem som legg til rette for mest mogleg gjenbruk.
- *Interoperabilitet:* IKT-system må kunne utveksle og dele data og informasjon med andre system gjennom standardiserte grensesnitt.
- *Tilgjenge:* Elektroniske brukartenester skal vere universelt utforma, og brukarane skal kunne nytte dei utan omsyn til tid, stad og kanal.

[2] Jf. f.eks. styrevedtaket fra 2010, hvor man utsatte vedtaket om innføring av Unicode (muligheten for å lagre og vise bl.a. samiske og greske tegn), for å bruke alle ressurser til å samle baser og øke MUSIT-aktiviteten i museene. (Innføring av Unicode ville medført at man måtte bruke ressurser på å oppgradere Oracle-databasen og applikasjonene. Nesten all trafikk på Internett er i dag basert på Unicode som er anbefalt standard for offisielle nettsider i Norge.)

[3] http://www.regjeringen.no/nb/dep/fad/dok/regpubl/stmeld/2008-2009/stmeld-nr-19-2008-2009-/6.html?id=552898

- *Tryggleik:* Informasjon og tenester skal tilfredsstille krav til konfidensialitet, kvalitet og tilgjenge.
- *Openheit:* Offentlege IKT-system skal vere baserte på opne eller godkjende standardar. Systema skal ikkje setje spesielle krav til teknologi hos brukarane.
- *Fleksibilitet:* Forvaltninga skal etablere og utvikle IKT-system på ein slik måte at dei er førebudde på endringar i bruk, innhald, organisering, eigarskap og infrastruktur.
- *Skalerbarheit:* IKT-system skal vere førebudde på endringar i talet på brukarar, datamengd og livslengda til tenesta.

MUSIT og museene må ta hensyn til dette når en utvikler nye IT-løsninger som svar på universitetsmuseenes framtidige behov og ønsker. Det ligger store effektiviseringsgevinster i mulighetene for heldigital dokumentasjon. Informasjonen kan bli raskere tilgjengelig for flere brukergrupper når den er skapt og strukturert digitalt og ikke må gjennom en digitaliseringsprosess for å bli tilgjengelig.

De arkeologiske utgravningene bidrar hvert år til mye ny informasjon. En god beskrivelse av den digitale arkeologiske produksjonskjeden (DAP) kan gi store effektiviseringsgevinster.[4] Målet for DAP er digital overføring av data mellom aktørene. De felles IT-løsningene som utvikles og driftes av MUSIT/DUG omfatter nå:

- Feltdokumentasjon
- Gjenstandskatalogisering
- Intern gjenstandshåndtering
- Utlån
- Fotografering

For katalogisering er det laget to grensesnitt – ett for generell gjenstandskatalogisering og to skjemaer for registrering av massemateriale – ett for steinaldermateriale og ett for middelaldermateriale. Skjemaet for middelalder kan også brukes for jernaldermateriale.

I løpet av 2013 vil følgende moduler ferdigstilles:

- Konservering (ferdigstilles i februar 2013)
- Aksesjon (ferdigstilles 2013)

Når Aksesjonsmodulen ferdigstilles, har de arkeologiske samlingene godt dekkende IT-verktøy for alle museale og andre forvaltningsoppgaver.
Videre arbeid vil omfatte

Bedre integrering av modulene, særlig til og fra feltdokumentasjonsmodulen
Samlet lagring og tilgjengeliggjøring av innmålingsdata fra felt
Grensesnitt for mobile enheter
Løsninger for multimedia (film, lyd)

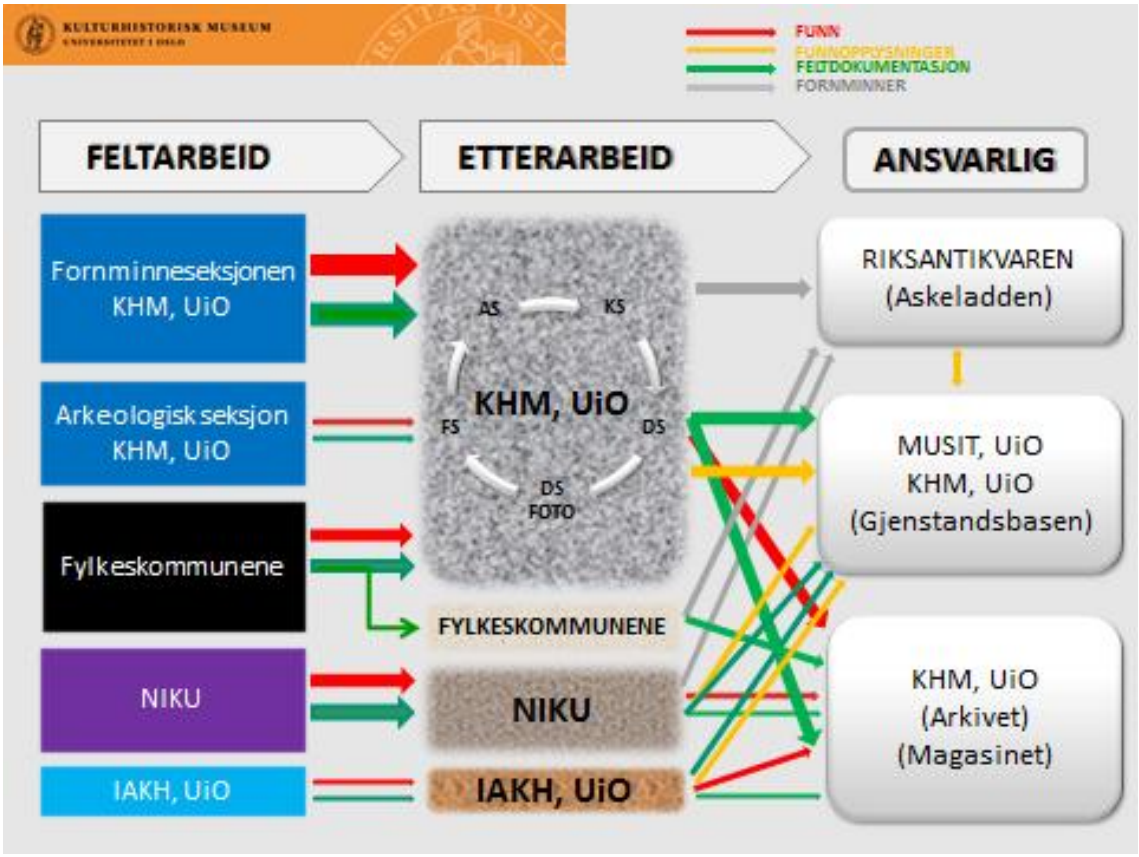
Arkeologiske utgravninger kan involvere fylkeskommuner, NIKU, Sjøfartsmuseer, universitetsmuseene, RA og universitetene. Resultater som lagres som lenkede, åpne data og eksponeres gjennom hele prosessen, krever en informasjonsarkitektur med ontologiske beskrivelser, tolkninger etc. Det er en stor utfordring for de arkeologiske samlingene at mye av resultatene fra den eksternt finansierte virksomheten, som gjenstandskataloger og

[4] http://www.slideshare.net/surikat/caanorway-2012-1015-ppna-data-terstlla-arkeologisk-information)

utgravningsrapporter, bruker lang tid på sin vei til publisering på nettet, og at det ikke er gode nok rutiner for å gjøre denne informasjonen raskt tilgjengelig.

Figur 2. Arbeidsoppgaver i museene (DFD=Digital Feltdokumentasjon)
(Figuren er utarbeidet av M. Matsumoto)

En annen del av DAP er arbeidsflyten inne på museet. Her ligger det et stort potensiale i å få bedre sammenheng mellom funksjonalitet i datasystemene og arbeidet med konservering, objektstudier, utstillinger osv. Det er flere gjenstandsgrupper som har ulike berøringspunkter. Mynter kan være blant gjenstandene fra en arkeologisk utgravning, men må også beskrives i en myntbase for at alle relevante opplysninger om myntene skal være med. Naturvitenskapelige prøver fra utgravningene skal analyseres og resultatene skal være tilgjengelige for arkeologene sammen med resten av opplysningene om funnene. Både human-osteologisk og animal-osteologisk materiale kan på samme måte både være arkeologiske funn og et materiale som studeres av naturvitere. Dette er eksempler på at en trenger en IT-arkitektur som gjør det mulig å sette sammen ulike opplysninger om objektene avhengig av hvilke personer og faggrupper som vil lese og beskrive dem og uavhengig av hvor informasjonen er lagret. Dette vil være mulig i en såkalt SOA-løsning (se *2Bi Tjenesteorientert arkitektur*) nedenfor).



Figur 3. DAP – Digital Arkeologisk Produksjonskjede.
Informasjons- og arbeidsflyt i forbindelse med utgravinger
(Figuren er utarbeidet av M. Matsumoto)

Innen naturhistorie har MUSIT-databasen primært vært utviklet for registrering av etikettert samlingsmateriale. Våre fremtidige behov og ønsker omfatter et dataverktøy som i større grad er knyttet til *arbeidsprosessene* i museene, alt fra feltregisteringer via arkivering til analyser og bruk. Det er meget sannsynlig at arbeidsstøtten skal deles opp i moduler, men det er grunnleggende viktig at modulene kommuniserer slik at det hele oppfattes som en helhetlig løsning.

Noen slike tenkte moduler er:

- Håndtering av feltdata tilknyttet materiale og/eller rene observasjoner registrert på GPS/PC/nettbrett/mobiltelefon/kamera
- Vasking og import av databaser/regneark fra interne og eksterne registratorer
- DNA-arbeid (støtte for arbeidsflyt på lab og for lagring av data). Spesielt støtte for rutinemessig DNA-barkoding
- Annet analyse-arbeid (morfologiske/anatomiske målinger, kjemiske analyser)
- Lagring av ulike typer multimedia tilknyttet objektene
- Forvaltning (er under utvikling): plassering/lån/preparering

SOA-løsningen skal gjenspeile behovene til forskere, forvaltere, studenter og allmenhet. Det er derfor behov for en gjennomgang av arbeidsrutinene ved museene slik at en kan bygge løsninger som er gode verktøy for flere av de arbeidsoppgavene som gjøres. Samtidig er dette en god anledning for museene til å tenke gjennom hvordan arbeidet gjennomføres og gå vekk

fra arbeidsmåter som er lite hensiktsmessige. Svært ofte er det slik at innføring av nye datasystemer fører til effektivisering. Ved nærmere ettersyn skyldes effektiviseringen bedre rutiner minst like mye som de nye datasystemene. For at dette skal skje, må en ha en nøye gjennomgang av rutinene og brukerne må oppleve de nye løsningene som arbeidsbesparende. De nye løsningene må være så gode at de avløser gamle rutiner og ikke blir et ekstra system som brukerne må forholde seg til.

Interessentanalyse
For å gi et godt svar på hva som er framtidige behov og ønsker i museene, bør det gjennomføres en interessentanalyse. Det er mange interessentgrupper med ulike mål og ønsker om hvordan registrering og tilgjengeliggjøring av informasjonen i universitetsmuseene felles samlingsløsninger skal være. Universitetsmuseene har behov knyttet til sin kjernevirksomhet; Forskning, Forvaltning og formidling. Kulturminneforvaltningen, RA, NIKU, sjøfartsmuseene og fylkeskommunene har interesser i tilgang til kulturhistoriske data. Tilsvarende har DN, fylkesmannen og Artsdatabanken interesser i tilgang til naturhistoriske data. Kommuner og fylkeskommuner trenger tilgang. Allmennheten trenger også tilgang, både til tilrettelagte data og til rådata. I arbeidet med denne rapporten har vi valgt å ta med noen viktige interessegrupper og hvilke forventninger de kan ha til løsningen museene velger:

Universitetsmuseene
Økt endringsevne.
Effektivitet - systemer som bidrar til gjenbruk av data og gjør data lettere tilgjengelig for forskning, samlingsforvaltning og formidling
Større grad av tilgjengelighet på tvers av faggrenser

Universitetsinstitutt
Lettere tilgang for forskere og studenter.
Raskere tilgang til forskningsresultater

Kulturminneforvaltningen/naturforvaltningen
Effektivitet - systemer som letter gjenbruk av og tilgang til data på tvers av institusjonene.

MUSIT (DUG)
Redusere personavhengighet i forvaltning av løsninger
Skape mer robuste systemer
Minske tid til systemvedlikehold og brukerstøtte og på den måten frigjøre tid til utviklingsoppgaver.

Noen generelle, sentrale mål felles for flere interessenter
Raskere tilgang til forskningsresultater/oppdaterte data
Mer effektiv gjenbruk av data

Risikoanalyse
Ved valg av ny IT-arkitektur må en bli bevisst de aktuelle risikofaktorene. Den største risikoen for universitetsmuseenes IKT-løsninger ligger i å utsette overgangen til ny IT-arkitektur. Det er nesten ingen andre som bruker rammeverket Delphi slik at det er svært vanskelig å rekruttere nye medarbeidere/utviklere. Dessuten er den nåværende IT-arkitekturen bygd på et klient-tjener mønster, noe som gjør at både klienter og tjenere fremstår som tykke og dermed er vanskelig å videreutvikle og forvalte.

- Underdimensjonering av årsverkene som kreves slik at gjennomføringen blir forsinket.
- Gjennomføringen krever så store ressurser i DUG at det ikke blir noe tid til konvertering av andre baser ved museene, og disse basene vil ta lenger tid å konvertere fordi programvaren de trenger ikke er like lett tilgjengelig.
- Det er et sikkerhetshull at brukere har direkte tilgang til Oracle-serveren gjennom sine Delphi-grensesnitt. Økte krav til sikkerhet kan føre til at vi blir pålagt å bytte løsning i løpet av kort tid.
- Det å knytte databasenær kode for tett opp til Oracle, gjør at man ikke enkelt kan bytte ut Oracle med andre databaser. Det bør utvikles mer generisk databasenær kode som ikke er avhengig av én databaseplattform.

Beskrivelse av dagens og framtidens IT-arkitektur

I dag består MUSITs IT-arkitektur av fire hovedkomponenter:[5]

- Oracle-databaser
- Databasenær programkode utviklet i PL/SQL, en Oracle-dialekt av SQL[6]
- En Oracle-klient på alle datamaskiner som benytter applikasjoner
- Applikasjoner (brukergrensesnitt) utviklet i programmeringsrammeverket Delphi

Valget av disse komponentene var velbegrunnedepå 1990-tallet. De har vist seg meget robuste og har gjort det mulig å drifte stadig flere og mer innholdsrike samlinger (databaser).

Med noen få, men viktige unntak, har arbeids- og dataflyten i museene endret seg fra oppstarten av Dokumentasjonsprosjektet på nittitallet til i dag. I Dokumentasjonsprosjektet, UNADOK og Museumsprosjektet var fokus massedigitalisering av analogt materiale, for eksempel kataloger og herbarieark. I dag lagres og manipuleres data digitalt fra første stund, ved innhenting i felt, til (endelig) oppbevaring i magasin. I felt brukes mobile enheter ved registrering, inkludert fotografering og koordinatfesting, og i magasinet benyttes strek- og QR-koder til å holde orden på hvor objektene befinner seg, enten de er lagret i magasinet, utlånt, på utstilling e.l.

Det har vært en rivende IT-teknologisk utvikling siden 1990-tallet, hvor Internett var i sin barndom, og kun kunne vise enkle, statiske HTML-sider, og hvor smartmobiler og nettbrett kun var en drøm. Tross den rivende utviklingen, er det først i disse dager at applikasjoner (grensesnitt) av den kompleksitet vi snakker om i MUSIT-sammenheng, kan utvikles for og "kjøres" i en nettleser eller på mobil og nettbrett.

Dette er bra for MUSIT. Det betyr nemlig at vi kan ta steget direkte fra gammel til ny teknologi, og trenger ikke å forholde oss til forskjellige mellomløsninger som har vært utviklet opp gjennom årene, hvor man har etterstrebet friheten fra maskinvare og operativsystem som en ren Internett-basert applikasjon gir.

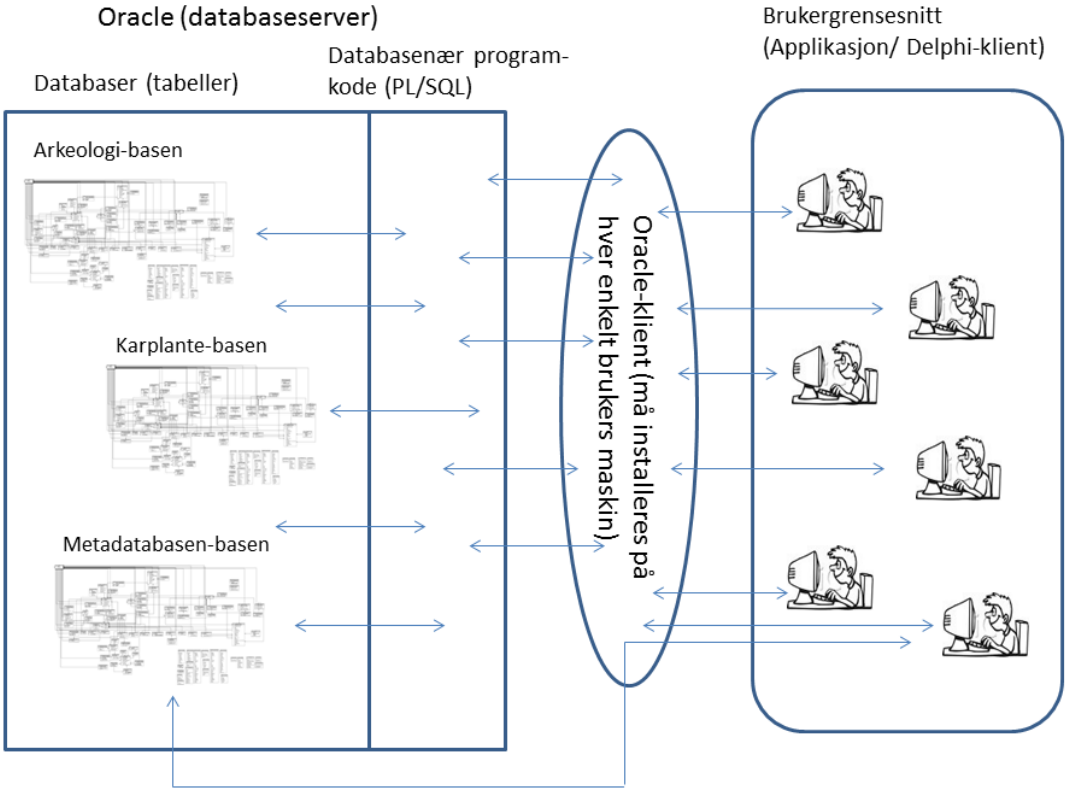
Samtidig som arkitekturen og utviklingsrammeverkene DUG benytter per i dag har vist seg robuste, skaper de også problemer. Rammeverket Delphi, f.eks., er det (nesten) ingen som benytter lenger, slik at vi har problemer med å rekruttere nye medarbeidere/ utviklere. Delphi-programmer går dessuten bare på Windows PCer, ikke Mac, Linux, nettbrett, mobiler osv. MUSIT trenger å utvikle (web)programmer som støtter alle plattformer.

Det å måtte installere en Oracle-klient på hver enkelt brukers datamaskin, skaper unødvendig mye problemer for DUG, brukerne og brukernes lokale IT-støtte. Dette burde

[5] I tillegg til hovedkomponentene som er beskrevet består systemet (arkitekturen) av programvare(-moduler) som håndterer (laster opp/ konverterer/ viser) bilder, PDF-filer og etiketter.

[6] SQL er en standardisert måte å søke i relasjonsdatabasetabeller på. Alle relasjonsdatabaser inkl. Access, MySQL osv. benytter en variant av SQL.

ikke være nødvendig. Videre er det et sikkerhetshull at brukere har direkte tilgang til Oracle-serveren gjennom sine Delphi-grensesnitt. Kun én maskin (en server) bør ha kontakt med Oracle-maskinen/ serveren. Det å knytte databasenær kode for tett opp til Oracle, gjør at man ikke enkelt kan bytte ut Oracle med andre databaser. Det bør utvikles mer generisk databasenær kode som ikke er avhengig av én databaseplattform. De to figurene nedenfor illustrerer hvordan arkitekturen ser ut i dag og hvordan den kan se ut etter en omlegging.

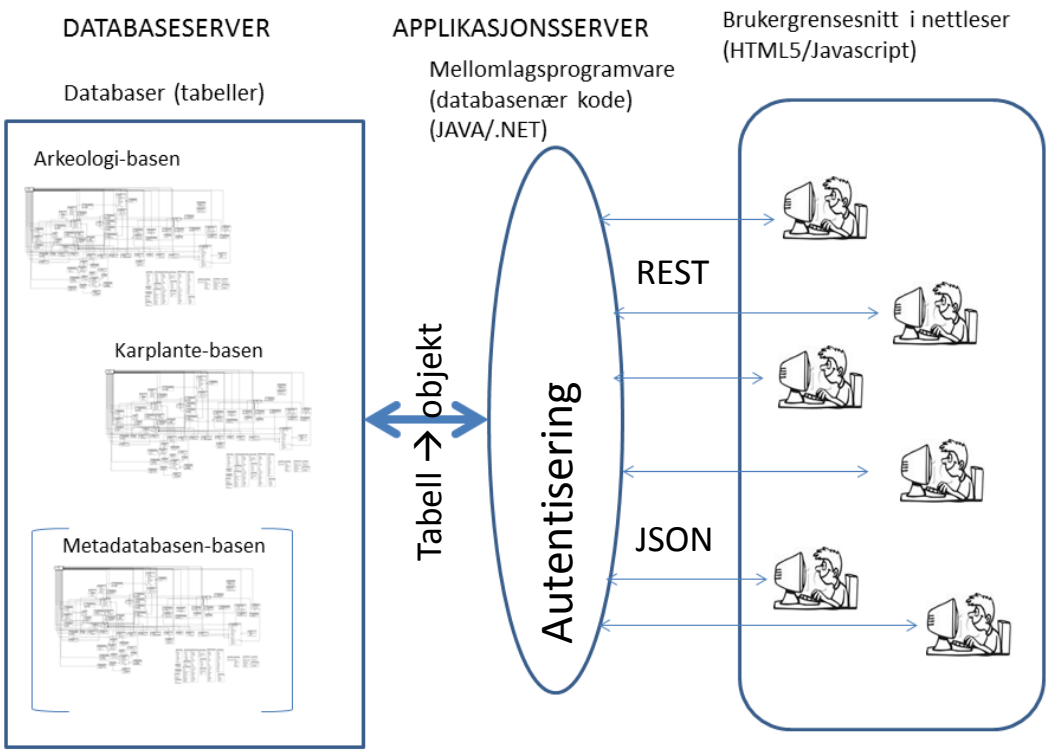


Figur 4. Dagens arkitektur

Overgangen fra figur 4 til figur 5 er ment å vise at utviklingen av et nytt system bør basere seg på at brukerne benytter nettleserapplikasjoner i sin kommunikasjon med samlingene og at denne kommunikasjonen går gjennom et mellomlag som holder styr på bl.a. hvilke rettigheter den enkelte bruker har i databasen (autorisering). Rettighetene knyttes til brukerens universitetskonto, slik at man kan bruke sitt vanlige brukernavn og passord ved pålogging via Feide, Felles Elektronisk IDEntitet, Kunnskapsdepartementets valgte løsning for sikker identifisering i utdanningssektoren.

I og med at kommunikasjonen mellom applikasjonsserveren (mellomlaget) og brukerne vil basere seg på standardformat (JSON/ XML) og standardprotokoller (REST/ SOAP), vil man kunne bruke den samme utdatastrømmen til å høste data fra databasene og kople de sammen med data fra andre kilder. På denne måten frigjør man dataene og gjør det mulig for andre, inkludert kommersielle aktører, å lage applikasjoner med basis i dataene i databasene.[7]

[7] Som i dag vil konservatorer/ forskere ha kontroll med hvilke data/ objekter som gjøres tilgjengelig for hvilke brukergrupper i og med at det er i databasen man styrer dette.



Figur 5. Framtidig arkitektur

Denne måten å håndtere inn- og utdatastrømmer på, via standardiserte protokoller, kalles noen ganger CRUD (create, read/ retrieve, update, delete), det vil si alle operasjonene som trengs for å lagre (create), hente ut (read/ retrieve), oppdatere (update) og slette (delete) data. Dette er som kjent basis-funksjonene som benyttes i den digitale samlingsforvaltningen. Et viktig poeng i denne sammenheng er utviklingen av gode rettighets- og kontrollmekanismer som sikrer at data kvalitetssikres og logges på en forsvarlig måte.

I forbindelse med at man utvikler en åpnere informasjonsarkitektur, er det viktig å trekke frem at det må arbeides mer med informasjonsarkitekturen for den type informasjon som skal registreres, behandles, lagres og hentes. Domenemodeller (helst felles), vil være viktig for å støtte gjenbruk og viderebruk av informasjon, gjerne i helt andre kontekster enn det den ble samlet til. En informasjonsarkitektur skiller seg fra en dataarkitektur ved at fokuset ikke ligger i databasen, men i representasjon og mening av informasjonselementer.

Mandatets 2B: Utrede ulike løsninger for MUSITs framtidige IT-arkitektur når det gjelder databaser og databaseteknologi

Mandatets 2Bi: Tjenesteorientert arkitektur

Dagens samlingsforvaltningssystem, inkludert MUSITs, består som oftest av to komponenter, en database og en applikasjon (klient) som muliggjør og setter rammer for brukerens interaksjon med innholdet i databasen. All informasjon brukeren trenger er lagret i databasen, og alle operasjoner og funksjoner som trengs for å oppdatere og redigere dataene styres av applikasjonen. Satt på spissen kan man si at hvis databasen ikke inneholder den informasjonen brukeren har bruk for, får han/ hun ikke tak i den, og hvis applikasjonen ikke har de funksjoner brukeren trenger for å hente ut og manipulere dataene på en hensiktsmessig måte, får ikke brukeren gjort det.

Et alternativ til en slik organisering av data og programmer er en tjenesteorientert arkitektur (SOA-arkitektur)[8], hvor man etterstreber en løsere kopling mellom data og operasjoner på data (tjenester) bygd på veldefinerte protokoller og format. Ved å beskrive og utvikle veldefinerte tjenester, gjør man seg mer uavhengig av hvor dataene er lagret og hvem som lagrer dem, og man gir andre lettere tilgang til sine data og derved også andre muligheten til å tilby tjenester basert på ens egne data. Norge digitalt og Yr er velkjente tjenestetilbydere som har tilrettelagt og beskrevet tilgang til sine data på en slik måte at det er enkelt for andre institusjoner å bruke dem i sine applikasjoner.

I tillegg til å tilby online-tjenester som gjør at man alltid har tilgang til de mest oppdaterte data, tilbyr for eksempel Norge digitalt også tjenester hvor man kan laste ned hele datasett for å kunne inkorporere disse i egne data eller lagre dem i egne databaser.

En tjenesteorientert arkitektur innebærer også at en bør gå ut over det å hente ut data fra en datakilde, til å se hvordan dataflyten og arbeidsflyten i organisasjonen er, slik at en får gode systemer som støtter det arbeidet som skal utføres. All innføring av datasystemer innebærer en omlegging av arbeidsrutiner. I de årene som er gått siden universitetsmuseenes felles databasesystemer ble først utviklet og tatt i bruk, har det også skjedd en endring i arbeidsrutinene ved museene. Ett eksempel er arbeidet med bilder. Museene leverer bilder til en rekke brukere. De som skal bruke bilder i ikke-kommersielle sammenhenger – til f.eks. studentoppgaver, faglige artikler, ulike typer formidling – har gjerne fått bildene gratis eller til en pris som neppe dekket museets omkostninger, fordi det hører med til museenes arbeidsoppgaver å formidle informasjon om samlingene. Nå bruker en tid på å registrere og legge fotografier inn i en base, og brukerne får tilgang til digitale versjoner av bildene. Det er en gjensidig påvirkning her – datasystemene lages ut fra de behov og rutiner museene har, og museenes behov og rutiner endres ved innføring av nye datasystemer. Når en nå skal se på IT-infrastrukturen, bør en se hvordan en helhetlig IT-løsning kan støtte museenes hovedoppgaver – forskning, forvaltning og formidling – slik at rutinearbeid blir mindre krevende og slik at forskning og det kreative arbeidet får mer plass. Viktige element er gjenbruk av data og gjenbruk av IT-løsninger på tvers av institusjoner.

Mandatets 2Bii: Utviklingsrammeverk for klient-server-teknologi, dvs. utviklingsverktøy for både databasenær kode, inkl. autentisering og brukergrensesnitt

På serversiden

Som tidligere beskrevet i forbindelse med diskusjonen av figur 5, vil sluttbrukerapplikasjonene (klientene) og andre, eksterne system aksessere våre tjenester via

[8] http://en.wikipedia.org/wiki/Service-oriented_architecture

REST[9]-API-er. Når det gjelder datautvekslingsformat for disse tjenestene, synes i dag JSON som klart mest utbredt slik at vi trolig kan ignorere XML som dataformat for REST-tjenestene dersom XML-støtte gir oss mye merarbeid. Det er imidlertid viktig at semantikken blir ivaretatt gjennom skjemaarbeid og ontologier. Utvikling av ontologier muliggjør automatikk for generering av XML-baserte skjema og meldinger. JSON foretrekkes i hovedsak fordi det medfører mindre behov for beregningsressurser, både på klienter og tjenere, dvs. at IT-tjenestene blir raskere.

På serversiden må vi ha en teknologi som gjør det lett å implementere REST-API-er, samt også å håndtere andres REST-API-er. Enn så lenge kan det også hende vi må takle SOAP-API-er, men SOAP blir mindre og mindre brukt og det er trolig unødvendig for oss å implementere nye SOAP-tjenester. SOAP kan imidlertid være nødvendig når meldingene som sendes har behov for økt sikkerhet mot innsyn og/eller endring.

I den nye arkitekturen vil det være mindre logikk (færre linjer kode) i klientene (sluttbrukerapplikasjonene) enn det var før, for eksempel vil det ikke lenger være mulig å gjøre direkte SQL-spørringer (oppslag) mot databasen fra klientene. Dette betyr at denne logikken må havne på serversiden, og at den da vil bli mer omfattende enn tidligere. (Vi har etterstrebet å ha så mye logikk som mulig på serversiden tidligere også, da i form av PL/SQL-kode, men vi har ikke vært konsekvente her.)

Konsekvensen av dette er at vi fra serverkoden må ha enkel tilgang til databasen, og gjerne også et system som sjekker at tabellnavn og andre database-objekter som det refereres til fra koden verifiseres av en kompilator (ikke ”dumme/opake SQL strenger”), samt et generelt kraftig programmeringsspråk som er lett å bruke og som gjør det lett å implementere og konsumere webtjenester. I dag har vi mye kode skrevet i Oracle PL/SQL. PL/SQL har én spesielt god egenskap, nemlig at SQL-uttrykk blir syntaks-sjekket under kompilering. For eksempel om vi har følgende spørring i koden:

```
SELECT GJENSTAND.MUSEUMSNR FROM GJENSTAND, GJENSTAND_PAA_FOTO, FOTOARK
WHERE
 GJENSTAND.GJENSTAND_ID = GJENSTAND_PAA_FOTO.GJENSTAND_ID
 AND GJENSTAND_PAA_FOTO.FOTOARK_ID = FOTOARK.FOTOARK_ID AND
FOTOARK.FOTOARK_ID = l_fotoark_id
```

Her sjekker PL/SQL-kompilatoren om vi har skrevet navn på tabeller og kolonner riktig. Redigeringsverktøy hjelper også til med å foreslå tabeller og kolonner, slik at man ikke behøver å huske eller slå opp dette. I tillegg til at det hjelper til med å skrive spørringen korrekt, så er dette også svært verdifullt idet noen gjør endringer i selve databasen; for eksempel om noen fjerner en av kolonnene i spørringen ovenfor, vil man få beskjed ved neste kompilering at denne kolonnen ikke (lenger) eksisterer.

Med ”dum/opak SQL streng” menes kode à la følgende:

```
ResultSet rs = stmt.executeQuery("
SELECT GJENSTAND.MUSEUMSNR FROM GJENSTAND, GJENSTAND_PAA_FOTO, FOTOARK
WHERE
 GJENSTAND.GJENSTAND_ID = GJENSTAND_PAA_FOTO.GJENSTAND_ID
 AND GJENSTAND_PAA_FOTO.FOTOARK_ID = FOTOARK.FOTOARK_ID AND
FOTOARK.FOTOARK_ID = ” + l_fotoark_id);
```

Her sendes SQL-spørringen direkte til databasen uten noen som helst sjekking eller tolking under kompilering, man skriver spørringene ”i blinde”. Dette gjør at man må bruke mer tid på

[9] http://en.wikipedia.org/wiki/Representational_state_transfer

testing, ev. ha et veldig effektivt testregime, samt at man er mye mer sårbar for sikkerhetsproblemer av type SQL Injections.[10]

Man mister også muligheten til å "komponere" spørringer i koden, dvs. man mister et sentralt abstraksjonsverktøy (ved "dumme/opake SQL-strenger" kan man ikke bygge opp spørringer fra mindre bestanddeler på en hensiktsmessig måte).

Serveralternativ 1: Java-plattformen (Java Virtual Machine (JVM))

Et vanlig valg er å benytte teknologi à la JBoss Seam.[11] Utviklerne i DS (= DUG/MUSIT) har ingen erfaring med denne type teknologi og er litt engstelige for at dette er tung teknologi å jobbe med. (Det synes som om andre her på USIT med praktisk erfaring med dette også er av samme oppfatning.)

En del av disse teknologiene generer de endelige html-sidene (brukergrensesnittene) på serveren. Dette er muligens feil vei å gå. Spesielt for vår gruppe som har lang erfaring med å lage tykke klienter er det trolig mest hensiktsmessig å lage html5-"apps" i form av såkalte "Single Page Applications", dette gir generelt også den beste html-baserte brukeropplevelsen. Vi vil uansett måtte ha egnede webtjenester for tredjepart som måtte ønske å lage "apps" baser på våre data, og da er det kanskje fordelaktig at vi selv blir tvunget til å bruke de samme tjenestene for våre egne html-"apps".

På serversiden vil det ikke genereres statiske html-sider (views), kun data som klienter kan vise frem som de måtte ønske.

DS ser på Java som et litt utdatert språk, mens JVM-en er god. De mest populære alternativene på JVM-en synes å være Clojure og Scala.

Lovende Scala-biblioteker for å snakke med databaser uten å gå via "dumme SQL-Strenger":

1. Squeryl (http://squeryl.org/)
2. Slick (http://slick.typesafe.com/)

I de neste avsnittene vil Java betegne plattformen (JVM-en etc), ikke Java som programmeringsspråk.

Serveralternativ 2: .NET

Alle utviklerne i DS har lang erfaring med Windows-programmering. Noen har også erfaring med programmeringsspråkene C# og/eller F# og/eller VisualBasic. Derfor vil det trolig være en litt mindre terskel for DS å ta i bruk en .NET-plattform på serversiden enn en JVM-variant. På den annen side er det stort sett klientprogrammering vi er vant med på Windows, ikke serverprogrammering, så det er ikke opplagt at dette er et spesielt vektig argument.

På .NET-siden vil det være naturlig å velge WEB-API[12] eller tilsvarende. Potensielle .NET-biblioteker for å snakke med databaser uten å gå via "dumme/opake SQL-Strenger":

1. Linq http://en.wikipedia.org/wiki/Language_Integrated_Query
2. Idealised Linq (forskingsstadie) http://homepages.inf.ed.ac.uk/jcheney/linq/essence-linq.pdf (F# har sin egen dialekt av Linq, som ligger nært 'Idealised Linq')

[10] http://en.wikipedia.org/wiki/SQL_injection
[11] http://www.seamframework.org/
[12] http://www.asp.net/web-api

Serveralternativ 3: JavaScript

NodeJS[13] synes å være ganske populær. NodeJS tar i bruk Googles velrennomerte JavaScript-motor (V8) for å kjøre JavaScript-kode utenfor nettleseren, kombinert med et bibliotek for å gi tilgang til filer og andre ressurser på den lokale maskinen, ressurser som i en vanlig JavaScript (i en nettleser)-kontekst er utilgjengelig fra JavaScript. NodeJS er populær fordi man kan gjenbruke sine JavaScript-kunnskaper på serversiden og bibliotekene som følger med er pragmatiske og lette å komme i gang med. NodeJS som serverteknologi er kontroversiell, det er en relativt utbredt oppfatning i mer akademiske programmeringskretser at NodeJS som serverteknologi har gjenoppfunnet et hjul som andre for lengst har forkastet, fordi det finnes langt bedre hjul der ute. NodeJS gjør det mulig å lage og kjøre kryssplattformkode uten å være avhengig av en nettleser, så det kan tenkes NodeJS på sikt vil være viktigere på klientsiden enn som serverteknologi.

Serveralternativ: Drøfting

NodeJS er trolig et for stort sjansespill på det nåværende tidspunkt. Det er for eksempel ikke opplagt at det per i dag finnes noen tilstrekkelig robuste måter å snakke med Oracle på fra NodeJS.

Når det gjelder å velge mellom Java og .NET, så er det et valg om programmeringsspråk og verktøy. Begge plattformene er anerkjent som svært robuste.

På Java-siden er trolig Clojure og Scala de mest naturlige alternativene. På .NET-siden er det C# og F#. Clojure er det språket som ligner minst på hva utviklerne i DS har erfaring med. (Riktignok har to i DS tatt et kurs på IFI som bruker Scheme, som er et språk i samme språkfamilie som Clojure, men ble ikke ”frelst”)

Hvis vi ser bort fra Clojure så synes det å være en utbredt oppfatning at om man sorterer C#, F#, Java og Scala etter kompleksitet/funksjonalitet så får vi følgende:

Java < C# < F# < Scala

Det vil si at Java er det mest ”primitive” språket, mens Scala er det mest komplekse/funksjonsrike.

I dette legges det at det tar lenger tid å lære seg all funksjonalitet i Scala enn for eksempel F#. Men det betyr ikke at kode som er enkel i for eksempel C# blir kompleks i F# og Scala, snarere tvert imot. Eksempel på en generell funksjon som snur rekkefølgen i et par: C# (tilsvarende i Java):

```
public static Tuple<U,T> Swap<T,U>(Tuple<U, T> t)
{
 return (new Tuple<U, T>(t.item2, t.Item1));
}
```

F#:

```
let swap(x,y) = (y,x)
```

Disse er helt ekvivalente, det blir samme (abstrakte) assemblerkode av disse.

I Java-verdenen er mange på jakt etter arvtageren etter Java (språket). Clojure og Scala er trolig de sterkeste kandidatene per i dag. For første gang kom disse to språkene i ”Adopt”-kategorien til Thoughtworks Technology Rader[14] i 2012. (F# ligger der bare på ”Assess”).

[13] http://nodejs.org/

[14] http://www.thoughtworks.com/articles/technology-radar-october-2012

Scala og Clojure skaper større bølger i Java-verdenen enn F# gjør i .NET-verdenen. To momenter som til dels kan forklare dette er at
Scala (og Clojure) er en langt sterkere/bedre konkurrent til Java enn det F# er til C# (p.g.a. rekkefølgen nevnt ovenfor)

Scala og F# er hybride språk, begge blander ObjektOrientert- og funksjonell programmering. Mange mener at dette er neste fase nå som OO ikke lenger har samme magiske status det har hatt de siste 20 årene. C# har kommet et stykke på vei mot å inkorporere dette, men sliter endel p.g.a. sitt opphav som et rent OO-språk. (Java sliter enda mer enn C#.)

Funksjonell programmering er dessuten svært egnet til å jobbe med data, spesielt til å velge ut, filtrere og på annen måte transformere data. (SQL Select kan sees på som et primitivt, ikke spesielt godt, funksjonelt språk). Parsing er også et relevant felt hvor funksjonell programmering har fortrinn. (Spesielt på naturhistoriesiden har DS en god del parseringskode/-behov.)

Det kan tenkes Scala er litt for komplekst[15] for denne konteksten, at F# kanskje er et mer egnet språk for DS, spesielt ettersom to i DS har endel F#-erfaring, mens ingen har Scala-erfaring.

Hvis DS satser på F# kan man trivielt blande inn C# (og VB), om noen av utviklerne skulle foretrekke det.

Både Scala og F# er Open Source. (Kildekoden for F# har Apache 2.0-lisens.) F# har svært god støtte for å konsumere web-tjenester, se for eksempel denne videoen: ”F# 3.0 and the Future of Information-Rich Functional Programming”[16].

Det bør også nevnes at C# 5 (med introdusering av async/await nøkkelord) og F# med sin Async workflow markedsledende støtte for å programmere asynkron kode på en enkel måte.

Når det gjelder ORM-rammeverk, [17] rammeverk for å gjøre det enklere å snakke med en relasjonsdatabase fra et objektorientert språk, så kan det tenkes det er mest hensiktsmessig å starte opp uten, og i stedet ha teknologi som utfører syntaks-sjekking av SQL-nær kode under kompilering, og heller utvikle egne biblioteker underveis.

Uansett teknologivalg på serversiden bør det legges inn åpning for at det vil, og må, komme endringer i teknologi i årene fremover, og at man ikke skal anta en slik stabilitet som man har hatt de siste årene. Dette betyr at man bør jobbe med teknologiplattformen slik at den er åpen for endringer og tilpasninger fremover i tid. Viktig også å ha med andre aspekter som kostnader, interoperabilitet, sikkerhet, etc. når man gjør et teknologivalg.

På klienten

På klientsiden har det heldigvis blitt rimelig opplagt hva man bør velge for å gjøre det lett for flest mulig brukere å få tilgang til våre databaser, nemlig HTML5, eller rettere sagt trekløveret HTML, CSS og JavaScript.

Dette er ikke nødvendigvis noen spesielt god arkitektur, men den er utbredt og populær fordi det er det eneste realistiske alternativet uten å gå veien om å lage ”native apps” for et utvalgt antall spesifikke klientplattformer. JavaScript var opprinnelig et hastverksarbeid og inneholder endel designfeil og svakheter. Man har i dag klart å forstå de innebygde feilene (boken ”JavaScript: The Good Parts” [18] var viktig i så måte). JavaScript har ingen eksplisitt innebygget støtte for å håndtere litt større prosjekter, for eksempel mangler det en

[15] http://blog.goodstuff.im/yes-virginia-scala-is-hard
[16] http://vimeo.com/57692834
[17] http://en.wikipedia.org/wiki/Object-relational_mapping
[18] Douglas Crockford: ”JavaScript: The Good Parts”

modulmekanisme, språket er dynamisk typet, samt at objektorienteringen er prototypebasert, ikke klassebasert. Utviklerne i DS har utelukkende erfaring med statisk typet og klassebasert objektorientering.

Av denne grunn tror gruppen man er best tjent med å ta i bruk et språk som konverteres/ kompileres til JavaScript, noe som gir oss bedre verktøystøtte og bedre mekanismer for litt større prosjekter enn dagens JavaScript gjør.

Det beste alternativet for oss per i dag synes å være TypeScript[19]. Dette er en konservativ utvidelse av JavaScript, all JavaScript er TypeScript, og det meste av det som TypeScript har utover standard JavaScript (for eksempel klasser og moduler) er basert på foreslått syntaks og semantikk for neste versjon av JavaScript (EcmaScript 6).

Det er Microsoft som står bak TypeScript, men det er et Open Source prosjekt (Apache 2.0-lisens). En av personene involvert med TypeScript er Anders Hejlsberg, mannen bak C# og som også var sterkt involvert med Delphi på 90-tallet. TypeScript vil være lett gjenkjennelig for vår gruppe med Delphi-programmerere og vi vil ikke miste "Intellisense" og endel annen verktøystøtte som vi har blitt vant/bortskjemt med opp gjennom årene, støtte som er langt fra så god med standard JavaScript. (Det er mye lettere å lage god verktøystøtte for TypeScript enn for JavaScript fordi TypeScript tillater statisk typing.)
Fordi TypeScript kompilerer til standard JavaScript kan man i fremtiden, om/når det skulle vise seg å dukke opp bedre alternativer enn TypeScript, ta med seg den genererte JavaScriptkoden videre.

TypeScript vil være langt mer produktivt å bruke for vår gruppe enn standard JavaScript, og vi kan ta i bruk kommende JavaScript-funksjonalitet som for eksempel klasser allerede nå, i stedet for å måtte vente noen år med å kunne ta i bruk dette.

Klientutviklingsrammeverk

Det er en skog av rammeverk man kan velge mellom på JavaScript-siden, men et som synes å utmerke seg er AngularJS[20]. Dette er et Open Source prosjekt i regi av Google (MIT-lisens). AngularJS er trolig det rammeverket per i dag som best speiler hvordan man vil utvikle våre typer webklientapplikasjoner i fremtiden. Det lar oss blant annet bruke ideer fra Web Components[21] allerede nå.

"Native Apps"

I noen tilfeller kan det kanskje være aktuelt å lage "native apps", spesielt i de tilfeller hvor man behøver å integrere sterkere mot hardware og spesielle operativsystem enn det som er mulig med en nettleser-app", dvs. høyst spesialiserte apps. For slike tilfeller vil trolig Xamarins[22] produkter være mest hensiktmessig for vår gruppe å bruke (spesielt om vi faller ned på .NET på serversiden). Xamarin selger spesialiserte produkter basert på Mono[23] som gjør det mulig å utvikle apps med standard .NET-teknologi som oversettes til native apps på iOS og Android . Det vil si man kan gjenbruke .NET-kunnskaper for å lage iPhone/iPad/Android apps samt bruke en allerede kjent utviklingsomgivelse som Visual Studio Xamarin (og "the community"[24] generelt) jobber dessuten med å gjøre F# til et reelt valg til C# også for native iOS/Android apps.

[19] http://www.typescriptlang.org/
[20] http://angularjs.org/
[21] https://dvcs.w3.org/hg/webcomponents/raw-file/tip/explainer/index.html
[22] http://xamarin.com/
[23] http://www.mono-project.com
[24] http://fsharp.org/

Databaseplattform
Per i dag ser DS ikke noen større besparelser i å velge en annen databaseplattform enn Oracle. Dette avhenger dog av utviklingsrammeverket på applikasjonsserversiden. Uansett vil vi i framtiden stå friere i valget av databaseløsning i og med at vi foreslår å flytte kode fra databaseserveren til applikasjonsserveren (mellomlaget).

"Nav" – globale ID-er
Vi ønsker at alle objekter (inkludert steder, hendelser, aktører, utgravinger, fotografier etc.) som er relevante i MUSIT-universet får en global unik ID. Dette gjør det lettere for DS å lage generiske systemer, i tillegg til at det gjør det lettere å sy sammen data fra forskjellige kilder.

Gitt en slik global ID bør systemet vårt internt holde rede på hvor man får tak i mer informasjon om dette objektet, for eksempel hvilke REST-endepunkter (og/eller websider) som finnes med potensielt mer informasjon om dette objektet.

Eksempel: For et objekt registrert i Askeladden, representert med en global, unik ID registrert i vårt system, bør det kunne være mulig for en klient som ikke aner noe om Askeladden å kunne hente ut Askeladden-data via vårt REST-API.

Mandatets 2Biii: Utrede ulike løsninger for MUSITs framtidige IT-arkitektur når det gjelder tilgjengeliggjøring for web, mobil og nettbrett i egne og eksterne portaler

Både rapporten om Nasjonalt digitalt universitetsmuseum (NDU)[25] og universitetsmuseenes utredning om DigitalViten, viser at det må tilføres nye midler dersom en skal satse mer på tilgjengeliggjøring av data, samtidig som museene arbeider med digital formidling av egne samlinger.

MUSITs styre nedsatte en portaliseringsgruppe som hadde ett møte sist høst. Diskusjonen i denne gruppa viste at universitetsmuseene har behov for felles presentasjoner på nett. Kulturhistorie har portaler under www.unimus.no for arkeologi, numismatika, etnografi, nyere kulturhistorie og fotografier. Naturhistorie har så langt presentert sine samlinger gjennom eksterne portaler som Artsdatabanken og GBiF. I tillegg kommer Encyclopedia of Life (eol.org) som er en ny internasjonal satsing som vil gjøre materiale fra de naturhistoriske samlingene tilgjengelig. Enkelte samlinger, som geologi, kommer ikke med i noen av disse.

De eksterne portalene dekker ikke alle behov universitetsmuseene har for tilgjengeliggjøring. Det er derfor et ønske om egne portaler hvor både naturhistoriske og kulturhistoriske data er tilgjengelig. Hensikten med egne nettsider må være å tilby noe mer enn det som gjøres ved å stille materialet til rådighet for eksterne aktører. Det kan blant annet være å tydeliggjøre fordelene og mulighetene som åpner seg når kulturhistoriske og naturhistoriske samlinger gjøres tilgjengelig sammen.

Stadig flere institusjoner legger ut sine data som åpne data, dvs. at de gjøres tilgjengelig i et format som er enkelt å ta inn i andre nettpresentasjoner. Når DUG tilrettelegger data for Norvegiana/Europeana blir de gjort tilgjengelige som åpne data. Det blir også vanlig at museer legger ut sine billedsamlinger til fri nedlasting og viderebruk. Dette er i tråd med EUs INSPIRE-direktiv, der et vesentlig poeng er at informasjon som er samlet inn med offentlige midler skal være fritt tilgjengelig for alle for å fremme videre verdiskaping i samfunnet.

Portaliseringsgruppa trakk også fram behovet for koordinatfesting av data. I en tid hvor kart blir stadig mer tilgjengelig og integrert i stadig flere mobile applikasjoner, vil det heve verdien og den mulige nytten av samlingene om så mye som mulig blir best mulig

[25] http://www.regjeringen.no/upload/KD/Vedlegg/UH/Rapporter_og_planer/Rapport_NDU-utvalget-090529.pdf

koordinatfestet. Koordinatfesting som del av den digitale arkeologiske feltdokumentasjonen er allerede på vei til å bli en del av hverdagen. Alle feltdata trenger en koordinatfesting for å gjøres tilgjengelig for de ulike brukergruppene på en god måte. Universitetsmuseenes samlinger bør også være tilgjengelige i Norge Digitalt, og da bør MUSIT kunne levere gode WMS-løsninger. Dette forutsetter mer tid og kompetanse på GIS innenfor MUSIT.

Dersom MUSIT skal være synlig på websidene, må en ha nødvendig kompetanse på nett. En minimumsløsning er å legge ut stadig mer som åpne data på en slik måte at alle universitetsmuseenes data er tilgjengelig fra samme adresse. Når universitetsmuseene legger ut sine data med en Creative Commons-lisens, vil det være enkelt for andre å bruke og gjenbruke tekst og bilder som er gjort tilgjengelig som åpne data. Et synlig nærvær på nettet vil være en styrke for universitetsmuseene.

Mandatets 2C: Vurdering av andre samlingsforvaltningssystem

I punkt 2C i mandatet, bes prosjektgruppen "Vurdere om det finnes andre system, spesielt i Norden og Europa for øvrig, som kan dekke de behovene museene har for samlingsforvaltning og tilgjengeliggjøring". Ut fra sin kjennskap til eksisterende databaseløsninger har gruppen konsentrert seg om å kartlegge hvilke system naboinstitusjoner i Sverige benytter.

Siden det ikke er lagt noen premisser eller kriterier for hva som skal vurderes, f.eks. at eventuelle system må kunne håndtere både kultur- og naturhistorisk materiale/ objekter, at systemene må kunne utveksle data med Artsdatabanken, GBIF og Europeana, at IT-organisasjoner ved norske universitet må ha kompetanse på teknologiene som er brukt osv., vil det måtte bli opp til styret å vurdere disse forholdene som en del av totalvurderingen av utredningen.

Kulturhistorie

Det er to miljøer for utvikling av store databasesystemer for kulturhistoriske museer i Norge: MUSIT og KulturIT. Kultur-ITs PRIMUS er et system for forvaltning av museumssamlinger primært rettet mot nyere kulturhistorie og kunstindustri. MUSITs databaser er rettet mot områder som arkeologi, etnografi og naturhistorie og vektlegger å understøtte forskning. Andre store museer i verden, som British Museum, har egne databaseløsninger som ikke er kommersielle. PRIMUS brukes også i Sverige og skal nå utvides med en modul for arkeologi. Det er Ulf Bodin som er ansvarlig for dette utviklingsarbeidet.

Bodin er arkeolog men arbeidet med utvikling av Intrasis til rundt 2000, designet deretter databasen som stadig er i bruk ved Statens historiska museum, og arbeider nå for Kultur-IT med å se hvilken retning databaseutvikling skal ta framover. Samtalen med Ulf Bodin bekreftet at en beveger seg vekk fra monolittiske databaser over til tjenesteorientert struktur.

Han ser for seg et kultur-nav – et sted for å utveksle informasjon. Kulturnavet skal være et sted hvor flere kan legge sine autoritetslister. Etter hvert som autoritetslistene fornyes vil navet kunne oppbevare flere versjoner av listene. Det vil være mulig å se hvem som har skapt autoritetslista, hvem som har gjort endringer, og når disse endringene er gjort. Systemet bør fra starten av være tilrettelagt for "crowdsourcing".

Objekter i kulturnavet skal kunne identifiseres ved unike ID'er Allerede ved den arkeologiske utgravningen kan en tildele hver enhet (gjenstand, prøve, …) en unik ID og la denne ID'en følge objektet slik at alle senere referanser til objektet kan gjøres ved denne. Slike globalt unike ID'er hjelper til å binde sammen informasjon. Så lenge det meste av ting (alt fra gjenstander til utgravninger til konkrete hendelser, steder, aktører etc.) har en global ID som helt uavhengige systemer kan finne frem til/ aksessere, så kan disse helt uavhengige databasene/ systemene alle snakke_om/ referere_til de samme objektene.

I prinsippet kan da hvem som helst bygge vilkårlig mange satellitt-databaser som gjennom felles ID'er kan referere de samme objektene og dermed indirekte snakke sammen. Kultur-navet blir skjelettet, mens et vilkårlig antall satellitt-baser/ kilder legger kjøtt på beinet.

Et kulturnav kan dermed også være et sted for å samle informasjonen fra utgravninger. En ville finne informasjon om gjenstander, kontekster og undersøkelser på samme sted. Museene ville ha ansvar for informasjon om gjenstandene, de som har gjennomført utgravninger kunne levere kontekstinformasjonen og resultater fra undersøkelsene. Bodin understreker at denne informasjonen egner seg svært godt som Linked Open Data (LOD).

Statens Historiska Museum (SHM) i Stockholm har hatt samme egenutviklete database-løsning siden 2002 og er nå i ferd med å sende ut anbud på ny IT-løsning for museet som kan forvalte arkeologi og numismatika. Museet skal utarbeide en kravspesifikasjon for det systemet de ønsker. De har få IT-tilsatte og derfor er alle IT-løsninger levert av eksterne. De skal ikke lenger ha et egenutviklet system. Et viktig aspekt ved det nye systemet er at brukergrensesnittet skal være web-basert. Dessuten vil de legge vekt på at det skal være et eget informasjonlag som flere system kan hente ut data fra. På den måten blir løsningen mer fleksibel. De vil med andre ord over på en SOA-løsning (Service Oriented Architecture).

Det ville vært ønskelig med ett system for både gjenstandsopplysninger og saksbehandling, men det de vil innhente anbud på nå er et system for gjenstandene.
SHM har digitalisert om lag 25 % av gjenstandene i samlingen som anslagsvis kan utgjøre 10 millioner objekter fordelt på 2-2,5 millioner gjenstandsposter. De regner med å bruke rundt ett år på å migrere eksisterende dataløsninger til et nytt system.

SHM bruker mye tid på å hente inn data i systemene sine. Det henger sammen med at det er mange aktører som produserer data i svensk arkeologi, og det er ikke alle disse som leverer data i formater som lar seg hente inn på en enkel måte. Kontekstinformasjonen, som ligger i den digitale arkeologiske feltdokumentasjonen, i et GIS, er ikke direkte knyttet til gjenstandene ved museet. I Sverige finnes rutiner for at gjenstandsdatabaser og GIS-systemer samles ved samme institusjon. SHM har bare gjenstandene og forventer derfor heller ikke at den IT-løsningen de skal anskaffe, skal håndtere mer enn det. De ser at det er behov for en større, nasjonal satsing for å få en løsning på felles lagring og enklere tilgang til feltdokumentasjonen.

I Norge kunne en tenke seg at et kulturnav kunne brukes for å utveksle informasjon fra fylkeskommunenes arkeologiske registreringer og museenes utgravninger. MUSITs felles stedsregister over alle matrikkelgårder i Norge er også et eksempel på en autoritetsliste som kunne gjøres tilgjengelig for flere brukere.

Ideen om et kulturnav er i samklang med den norske regjeringens planer om at alle offentlige institusjoner skal gjøre sine data tilgjengelig. Direktoratet for forvaltning og IKT har opprettet nettstedet data.norge.no som en portal for åpne offentlige data i Norge. Denne nettsiden skal være for "datasett som inneholder alt fra kraftpriser til flydata"[26] Så lenge en bruker globalt unike ID'er og åpner opp tilgangen for å dele data, er det mange muligheter for mange ulike aktørers deling og høsting av data.

De felles løsningene for arkeologi bør baseres på en flerlagsstruktur. Det bør være et åpent system hvor hvilke program som håndterer de ulike delene av informasjonen er mindre viktig enn at IT-løsningen kan hente og levere data i forhold til flere ulike kilder. Persistente ID'er som gjør at ulike typer data kan omtales og kombineres på mange måter blir en viktig forutsetning.

Våre samtaler i Sverige bekrefter det inntrykket vi har at det ikke er noen andre samlingsforvaltningssystem som enkelt kan erstatte MUSIT-basene i dag.

[26] http://data.norge.no/data

Naturhistorie

På 2000-tallet vurderte NHM og DINA[27]-prosjektet i Sverige flere samlingsforvaltningssystem som ledd i en søken etter ett system som kunne brukes for alle samlingsdatabasene ved museene. To system pekte seg ut, nemlig Specify og KE EMu.[28] Ved en nærmere sammenligning av de to systemene (se Wiig et al 2006[29] og Stengård 2008) ble KE EMu foretrukket framfor Specify. Det var dog ikke full enighet om valget ved NHM, noe som kommer til uttrykk i Wiig et al (2006: 23): "Ett av utvalgets medlemmer mener at for entomologene framstår KE EMu lite fristende".

På initiativ fra NHM, ble KE EMu testet ut i MUSIT i 2009.[30] Flertallet av gruppen som var med på uttestingen konkluderte med at "KE EMu ikke vil egne seg til bruk ved de naturhistoriske universitetsmuseene i Norge." Et mindretall mente det motsatte. Styret for MUSIT valgte ikke å gå videre med å innføre KE EMu i MUSIT.

DINA-prosjektet foretok en tilsvarende snuoperasjon i 2009/2010 og etter en nøye sammenligning av forskjellige system, primært CollectionSpace og Specify, falt valget på Specify.[31] Åpen kildekode og flerlagsarkitiektur var viktige moment ved valget. Naturhistoriska riksmuseet (NRM), som er ledende i DINA-prosjektet, benytter nå Specify for objekter og Morphbank for mediefiler (bilder osv.). DINA-prosjektet har siden blitt utvidet til å omfatte institusjoner i Danmark og Estland og Specify er innført ved flere av institusjonene. De vil nå sette opp et konsortium for videreutvikling hvor også institusjoner i Berlin og Paris har uttrykt sin interesse for å delta.

Specify er et system basert på åpen kildekode. Det arbeidet en gjør ved DINA-prosjektet er å utvikle et web-basert brukergrensesnitt inn mot tabellstrukturene i Specify. Dette systemet blir også bygget som en flerlags tjenesteorientert arkitektur (SOA).

Utviklingen av kjernen i Specify skjer ved the Biodiversity Research Center, The University of Kansas, USA. Hvert deltakerland i DINA har sine egne nasjonale installasjoner. I hvert land vil en institusjon være vert og de andre vil arbeide over web mot databasen hos vertsinstitusjonen. Hvert land er dermed avhengig av å ha et eget, lokalt driftsmiljø.

I det videre arbeidet i DINA-prosjektet vil en komme fram til en klarere modularisering og ansvarsfordeling mellom partene, slik at hver partner skal ha ansvar for utvikling av en bestemt del av totalløsningen. NRM fortsetter med utvikling av webgrensesnittet. Den estiske partner har planer om å utvikle en taksonomimodul. Taksonomimodulen ligger lokalt i Specify-løsningen. Dette er valgt fordi NRM ikke har skrivetilgang til det svenske nasjonale taksonregisteret, Dyntaxa ved den svenske ArtDatabanken.

Vi antar at behovet for struktur og funksjonalitet i databasene ved de naturhistoriske universitetssamlingene i Norden (og verden for øvrig) er ganske sammenfallende. Norske samlinger har riktignok en utstrakt bruk av geografiske koordinater på formen UTM (i MGRS-notasjon) og har et uttrykt ønske om å benytte et nasjonalt taksonregister (Artsnavnebase hos Artsdatabanken). Det kan synes som at UTM/MGRS-koordinater er dårlig støttet av Specify i dag, men dette er en global standard som vi vil tro mange vil ha interesse av å få implementert i Specify. Nordens flora og fauna består i stor grad av felles

---

[27] DINA = Digital Information System for Natural History Collections:
A collection management system developed in an international consortium using open source components

[28] Wiig et al (2005), "Valg av programvareløsning for samlingsdatabaser ved Naturhistorisk museum, Universitetet i Oslo" og Stengård, Eva (2008), "A national system for digitizing biological specimens – DINA migration phase".

[29] Wiig et al (2006), "Vurdering av Specify og KE EMu for bruk som samlingsdatabaser ved Naturhistorisk museum, UiO".

[30] Wiig et al (2009), "Evaluering av samlingsforvaltningssystemet KE EMu for bruk ved universitetsmuseene i Norge".

[31] Se Eriksson, Torsten (2010), "Gemensamt system för samlingsdatabaser".

arter og en orientering mot et samarbeid om et felles nordisk taksonregister blir av oss sett på som positivt. Det er nok tekniske problemer å løse før Specify kan benytte et eksternt taksonregister, men vi anser modellen der våre relativt få taksonomiske eksperter (som ikke alltid sitter ved de naturhistoriske samlingene) samarbeider om et felles norsk/nordisk taksonregister som riktig. Dette registeret må være lesbart for alle datasystemer som trenger validerte taksonnnavn i Norge/Norden – dette er langt fler enn universitetsmuseenes.

De naturhistoriske samlingene har behov for dataløsninger som støtter hele prosessen materialet gjennomgår fra feltarbeid til arkivering og senere analyse/bruk. MUSIT-basene er bygget primært for registrering av etikettert materiale i samlingene og har behov for store utvidelser, spesielt i retning feltregistrering og støtte for analyse av materialet. Behovet har vært så stort at f.eks. NHM har utviklet et eksternt verktøy for DNA-arbeidet som dessverre kommuniserer dårlig med samlingsdatabasen og til dels dupliserer dens innhold. Med tanke på de begrensede ressurser vi har til utvikling av vårt eget datasystem i MUSIT, fremstår det å inngå samarbeid med parallelle utviklingsprosjekter i våre naboland som fristende. Gruppen anbefaler at MUSIT vurderer et framtidig samarbeid med DINA-prosjektet. Uansett bør MUSIT undersøke om man kan få observatørstatus i konsortiet, slik at man blir innkalt til, og kan delta, på møter.

Mandatets 2D: Hvilke investeringer og årlige ressurser kreves

Investeringene som må til i form at utviklingsarbeid for å komme fra Figur 4 til Figur 5 vil ha følgende investeringsramme i antall årsverk:

Aktivitet	**Årsverk**	**Kommentar**
Installere og drifte mellomlagsapplikasjonen	1,0	Hvor stor denne investeringen blir, avhenger noe av om man velger en JAVA eller .NET-løsning
Konvertere dagens Oracle-database og applikasjoner til Unicode	1,0	
Implementere og teste ny mellomlagsprogramvare	1,5	
Flytte databasenær kode ut i mellomlaget	1,0	
Migrere naturhistorieapplikasjonene	2,0	
Migrere arkeologi	2,0	
Migrere øvrige samlinger (etno./ kulturhistorie)	2,0	
Totalt	**10,5**	

Når det gjelder migrering av øvrige løsninger tas det for gitt at vi har fått til én løsning for etnografi/ nyere kulturhistorie inkludert et arkiv for beskrivelse av kulturhistoriske foto. Det legges også til grunn at mynt er flyttet til arkeologi, at resten av de numismatiske objektene, medaljer, sedler osv., er flyttet, eller kan flyttes, over i den migrerte etnografiløsningen. Videre er det i tabellen over ikke tatt høyde for at vi har flere databaser og applikasjoner for naturhistorie enn det vi har per 31.12.2013.

Gruppen anbefaler dessuten av MUSIT faser ut utvikling og drift av topografisk arkiv og at man finner en ny løsning for hvordan man forvalter digitale foto og andre medietyper.[32]

Hvis vi tar utgangspunkt i dagens bemanning i DUG med en utviklerkapasitet på 2,5 årsverk per år vil det ta ca. 4,2 år å gjennomføre overgangen for de samlingene MUSIT

[32] Dette kan for eksempel være noe som BIBSYS kan ha kompetanse på og interesse av å tilby løsning for. Det piloteres for tiden en løsning for en databank for læringsobjekter som på sikt også skal ivareta alle typer digitale informasjonsobjekter. Informasjonsobjektene skal gjøres søkbare ved etablering av taksonomier, metadata, rettighetsdata, etc.

forvalter per i dag. Dette er dog et optimistisk anslag fordi man da ikke har tatt høyde for permisjoner eller at noen slutter osv. Man bør derfor uansett se på om man skal få i stand en rammeavtale med et firma som kan bistå utviklingsarbeidet, eller se om det finnes ressurser innenfor universitetenes IT-avdelinger.

Avslutningsvis bør man merke seg at vi her ikke har innkalkulert hvor lang tid og hvilke ressurser man ser for seg at det kreves å beskrive dataflyten i museene for de forskjellige samlingene.

Vi vil ikke angi konkrete prisoverslag for arbeidet, men med en timepris på NOK 1500 kan en komme fram til et totalbudsjett på nærmere 20-25 millioner dersom arbeidet skal gjøres av eksterne leverandører.

Mandatets 2E: Anbefale én løsning for ny IT-arkitektur og foreslå tidsplan for implementering

Det er vår anbefaling av MUSIT beveger seg bort fra dagens to-lags arkitektur med de begrensninger det medfører og over mot en tjenesteorientert, flerlagsarkitektur. MUSIT har nå anledning til å sette strek og komme fram til et nytt, funksjonelt system som ikke tar samme hensyn til den lange historikken som gjør dagens løsninger mer ressurskrevende enn nødvendig.

Gruppen anbefaler derfor på det sterkeste at man gjør et arbeid for å beskrive arbeids- og dataflyten ved museene (virksomhetsarkitekturen) som et første ledd i arbeidet med å (videre)utvikle, ev. anskaffe, en eller flere nye IT-løsninger for forvaltning av museumsobjekter. I den forbindelse bør man også se på om det er databaser og applikasjoner man i dag vedlikeholder i regi av MUSIT som man bør gjøre seg av med eller la andre overta. Intrasis som nå brukes til feltdokumentasjon ved arkeologiske undersøkelser er et eksempel på å hente inn programvare for å dekke ett bestemt behov innenfor den felles løsningen som MUSIT har ansvar for. Topografisk arkiv og fotobasen, i alle fall den delen som går på det å ta bilder, manipulere bildeformat og lagre bildene, er applikasjoner som ikke nødvendigvis vil være del av MUSITs portefølje av databaser og applikasjoner i framtiden.

Anbefalingen kan summeres opp i følgende hovedpunkter

- Det skal ikke være ett system som inneholder alle komponenter
- Det skal være en flerlagsarkitektur (SOA)
- En skal vekk fra en skog av ulike databaseløsninger som har samme/tilsvarende funksjonalitet
- Målet skal være én funksjonalitet - én IT-løsning - uavhengig av fagområde. (Felles løsninger for magasin, utlån/innlån, multimedia, fotografier, arkiv) Dette kan innebære at museene må tilpasse sine eksisterende rutiner.
- Ut fra bl.a. utviklernes kompetanse velges en .NET-løsning framfor en Java-løsning
- Et estimat på tidsbruk kan summeres til 14 årsverk. Da er det satt beregnet ett månedsverk ved hvert museum for å gjennomføre en virksomhetsanalyse. Virksomhetsanalyse vil dessuten trenge ressurser til en prosessleder og 2-3 møter.
- Dersom hele prosessen skal gjennomføres med eksterne konsulenter vil det kreve over NOK 22 millioner. Dersom hele prosessen gjøres med egne tilsatte vil et budsjett på 11 millioner rekke langt. Den beste måten å gjennomføre prosessen på vil være en kombinasjon av disse alternativene, der MUSIT-tilsatte, andre tilsatte ved universitetenes IT-avdelinger og eksterne konsulenter gjennomfører ulike deloppgaver.
- Vi har ikke tallfestet driftsutgiftene ved overgang til ny arkitektur. Det er rimelig å anta at de vil være i samme størrelsesorden som for dagens løsning.

Følgende tidsplan vil gi en gjennomføring i løpet av 4,5 år. Noen av aktivitetene må gjøres i rekkefølge, men andre kan gjennomføres samtidig:

Aktivitet	**Årsverk**	**Halvår**								
		1	2	3	4	5	6	7	8	9
Virksomhetsanalyse ved museene	0,5	x	x	x						
Installere og drifte mellomlagsapplikasjonen	1	x	x							
Konvertere dagens Oracle-database og applikasjoner til Unicode	1	x	x							
Implementere og teste ny mellomlagsprogramvare	1,5			x	x	x				
Flytte databasenær kode ut i mellomlaget	1				x	x				
Migrere naturhistorieapplikasjonene	2						x	x	x	x
Migrere arkeologi	2						x	x	x	x
Migrere øvrige samlinger (etno./ kulturhistorie)	2						x	x	x	x
Migrere databaser som ikke er MUSIT-baser	3	x	x	x	x	x	x	x	x	x