områder der skal dokumenteres, for at kunne kommunikeres. Eksempelvis er Rolle et arkitekturelement (se nedenfor) fra cellen ”Roller og grupper”, der er relateret både vertikalt til en række personer, men også horisontalt til de forretningsprocesser de indgår i. Alle arkitekturelementer ”hører hjemme” i en celle.

Cellernes titel angiver hvad vi gerne vil kommunikere med netop den celle. Titlerne kan enten pege på et specifikt arkitekturprodukt eller en række arkitekturelementtyper. En celle der definerer et arkitekturprodukt er eksempelvis ”Domænemodel” som der kun vil findes en af for organistionen. Et eksempel på en celletitel der angiver en eller flere arkitekturelementtyper er Systemer. Selvom denne celle også indeholder modeller (en for hvert system) angives det navn ikke, da der netop vil findes en lang række diagrammer, og ikke en unik model.

Arkitekturelement

I det ovenstående er arkitekturelementer allerede blevet kort skitseret, men de er kort sagt de ”ting” vi vil vide noget om. Sammenligner vi rammeværket med en database, er arkitekturelementerne databasens entiteter. Den endelige liste af arkitekturelementer skulle dermed gerne præsentere alt hvad man i en EA funktion skal vide om sin organisation og dens virken.

Når det så er skrevet, må man også erkende at det ikke er muligt at definere en organisation komplet, før man har fået sin arkitektur op at stå. Der vil være ting man har overset, ting man ikke kan definere på det stade man befinder sig på og ting der først senere viser deres relevans. Derfor må man være åben overfor at revidere de eksisterende arkitekturelementer og tilføje nye når det viser sig nødvendigt. Vi ville gerne have indarbejdet S.L.A. som arkitekturelement, men måtte indse at organisationen simpelthen ikke var moden til at datamodellere de ansvarsforhold S.L.A.´en ville kræve. Det betyder naturligvis ikke at tanken er opgivet, blot at den er lagt i bero, indtil vi mener det vil være muligt.

Forholdet mellem søjler og rækker

Udover at hjælpe til at organisere organisationens viden om sig selv, hjælper søjlerne og rækkerne også organisationen til at kommunikere bedre internt. Rammeværket angiver hvordan man effektivt kommunikerer med hinanden i en organisation, efter to simple regler; man kan kun kommunikere vertikalt eller horisontalt. Prøver man at kommunikere diagonalt, vil modtageren højst sandsynligt ikke forstå konteksten for kommunikationen. Det giver ikke meningen at fortælle at virksomhedssystemet bruger tabellen Org_Ansat, altså fra cellen Systemer på det logiske niveau, diagonalt ned til cellen Data på det operationelle niveau. Man bør i

stedet fortælle at virksomhedssystemet anvender det logiske forretningsobjekt Ansat, og derved blive på det logiske niveau. Alternativt kan man fortælle at virksomhedssystemets komponent AdministrerAnsat behandler tabellen Org_Ansat, og holde informationen på et fysisk niveau.

Samme forhold gør sig gældende i designet af rammeværket. Forhold man ikke mener skal kunne opstå placeres i diagonalt placerede celler. Vi besluttede eksempelvis hurtigt at forretningsprocesser skulle relatere sig til de roller der udførte dem, og ikke til navngivne personer, da det ville skabe et uholdbart niveau af administration og vedligeholdelse. Eftersom Person-elementet er på et operationelt niveau, og forretningsprocesserne på et logisk har vi derved umuliggjort en direkte relation derimellem. Vil man så vide hvem der udfører noget specifikt, må man først tjekke hvilken rolle der udfører en forretningsproces, og så finde ud af hvilke personer der har den rolle.

Diagrammer og modeller

Som rammeværket er beskrevet indtil nu, ville det mest illustrative vi kunne få ud af det, være en liste af arkitekturelementer og deres relationer. Slet ikke uvæsentligt, men ej heller tilstrækkeligt. Med den værktøjsunderstøttelse vi arbejder med, kan vi tegne databaseunderstøttede diagrammer. Det betyder at vi kan indsætte vores arkitekturelementer på diagrammer, tegne pile, forbindelser og meget mere imellem dem, og vide at disse tegnede relationer registreres i databasen. Altså, hvis vi tegner en systemmodel, der viser at systemet Avid har en integration, angivet ved en pil, til systemet Mediearkiv, så ved den database der understøtter rammeværket det også. Spørger vi så på et senere tidspunkt databasen hvilke relationer Avid har til anden teknologi, vil den finde og præsentere denne integration.

Vi har altså valgt en række diagramtyper der bedst kan anvendes til at kommunikere forskellige historier. På disse diagramtyper bestemmer vi hvilke arkitekturelementer der kan indsættes, og hvordan de kan forbindes med hinanden. Når så en person har tegnet et diagram har vi dermed opnået to formål; diagrammet fortæller lige præcis den historie tegneren ønsker og samtidig indgår den historie i vores samlede kontekst, og kan senere fremfindes fra databasen.

Selvfølgelig findes der ingen regel uden undtagelser, og desværre er det ikke helt så simpelt som ovenstående angiver. Nogle gange er diagrammerne nemlig arkitekturelementer, i stedet for at symbolerne på diagrammet er det. Et workflow er i vores verden et arkitekturelement, men repræsenteres som et workflowdiagram,

indeholdende både symboler der ikke er arkitekturelementer (ex. aktiviteter) og symboler der selv er arkitekturelementer (ex. Rolle).

4.4 DRs Rammeværk og dets indhold

Efter at have beskrevet rammeværkets opbygning, er det nu tid til at gennemgå rammeværket i detaljer. Dette afsnit begynder med beskrivelse af søjlerne, men hovedparten bliver beskrivelsen af hver celle og de dertil hørende arkitekturelementer og diagramtyper. Det der præsenteres er rammeværkets udformning ultimo 2005, hvor der stadig er planlagt endnu en review-runde. Denne finder dog ikke sted før forventeligt juni 2006, hvorfor denne udgave bliver den, for dette speciale, gældende. Udover præsentation, vil jeg også ridse de mest relevante af de mange diskussioner og overvejelser der har været i forbindelse med definering og design. Til sidst i afsnitte vil jeg gå ind i en detaljeret beskrivelse af relationerne mellem de mange arkitekturelementer.

Rammeværkets søjler

Strategi

Søjlen strategi beskæftiger sig med de ledelsesmæssige discipliner der sætter rammer og afstikker kursen for organisationen. De strategiske celler og arkitekturelementer gennemløber hele organisationen, og er på samme tid målet for alle andre celler og reglerne for hvordan der kan ageres indenfor dem.

Organisation

Søjlen her beskæftiger sig med hvordan organisationen er struktureret med hensyn til hvem der står for hvad. Alle arkitekturelementer der refererer personer, deres ansvar eller grupperinger hører hjemme i Organisation.

Proces

Processøjlen omhandler alt hvad der gøres af folk i organisationen. Alle handlinger der indgår i en struktureret rækkefølge bør være dokumenteret som processer. Processerne på alle tre niveauer bliver målestok for hvor god en teknologiunderstøttelse vi har (ved at relatere arkitekturelementer fra processøjlen med de relevante fra systemsøjlen), hvilket vi ser som noget af det mest centrale for vores arbejde med EA.

Information

I DR arbejder vi stort set kun med information, lige fra tekst der skal på vores nyhedsportal til samtlige gamle DR udsendelser der i fremtiden vil blive digitale (og dermed information). Stort set hele produktionsapparatet er digitaliseret, og det er i denne søjle at det må dokumenteres og relateres til de processer der behandler dem, og den teknologi der understøtter informationen.

System og Infrastruktur

De to yderste søjler er begge teknologi. Systemsøjlen beskæftiger sig med softwareløsninger, mens Infrastruktur beskæftiger sig med den teknologi, hardware og software, der gør det muligt for systemerne at understøtte forretningsprocesserne. Groft sagt hører en ting hjemme i system-søjlen hvis den er med til direkte at understøtte en forretningsproces, hvorimod den hører hjemme som infrastruktur hvis den er med til at understøtte et eller flere systemer. Denne sondring er ikke den simpleste men valget faldt herpå efter mange måneders diskussion. I begyndelsen var de adskilt som software og hardware, men den definition holdte ikke længe. Hvordan skal man så kunne definere en fysisk server, der jo både består af noget metal, men også et styresystem?

På et tidspunkt blev de to søjler slået sammen, netop fordi det var for svært at skelne mellem hardware og software, og til hvilken grad der skulle være det ene eller det andet for at det lå i en bestemt søjle. Det betød dog en tilbagevenden til et problem, vi fra begyndelsen havde villet komme til livs. Når man ser mange diagrammer der skal vise hvordan et nyt system ser ud, er det ofte et stort rod af fysiske enheder, software og datastrukturer. Man viser på et diagram hvordan en fysisk server, i en bestemt kælder, kører en databaseserver og hvilke tabeller der ligger i databasen. Vi ville gerne frem til at man endte med flere, men simplere, diagrammer, der ville være nemmere for læseren at forstå.

Det endte derfor at vi gik tilbage til de to søjler, med den beskrevne definition. Denne inddeling har desuden fordelen at vi i IT-organisationen nemt kan overskue hvad der er kundevendt (altså mod andre afdelinger i DR) og hvad der er bagvedliggende grundteknologi.

Celler, diagramtyper og arkitekturelementer

I dette afsnit vil jeg gennemgå samtlige 18 celler i DRs rammeværk. Cellerne vil blive gennemgået søjlevis fra højre mod venstre. For hver celle vil de relevante arkitekturelementer og diagramtyper blive beskrevet. For læseren er der før hver

beskrivelse af en celle indsat en lille model af rammeværket med et kryds der angiver hvor vi er i rammeværket.

Vision og strategimodeller

Beskrivelse: Cellen omhandler organisationens øverste strategiske niveauer. For en given strategi, afdeling eller projekt opretter man et strategidiagram. Herpå kan man så indsætte de fire typer arkitekturelementer.

Arkitekturelementer:

- Mission: En missionsbeskrivelse er en kort sætning om hvorfor organisationen, afdelingen eller gruppen eksisterer og hvad dens formål er.

- Vision: En vision er en kort sætning der formulerer hvor organisationen er på vej hen, et S.L.A.gs pejlemærke for udviklingen over et længere stykke tid. En vision skal være så vag at den ikke kan måles på (”DR Ikke som de andre” som den nyligt er blevet formuleret i DR).

- Mål: Mål opsættes på baggrund af organisationens vision, og er til for at man kan måle om man er nået derhen hvor man gerne ville.

- Strategi: En strategi er en handlingsplan der fortæller hvordan organisationen skal nå henimod sin vision, som regel ved at føre til opfyldelsen af et konkret mål.

Diagramtyper: Strategidiagram. Diagrammet er ikke baseret på nogen standard, men er udviklet til arkitekturværktøjet der anvendes. Der findes ingen standarder for denne type diagrammer. Diagrammet består, foruden de fire arkitekturelementer, af forskellige måder at forbinde disse på, for derved at kunne vise tilhørsforhold mellem arkitekturelementerne.

Diskussion:

Der er mange måder at arbejde med strategi og ovenstående er blot en. Denne måde er dog efterhånden blevet en klassisk måde at inddele feltet på om end man ofte ser flere elementer, såsom kritiske succes faktorer. Vi valgte at basere os på velkendte og nemt anvendelige elementer, da vi ved at det er en broget forsamling der arbejder strategisk i DR, og de vil have en meget forskellig tilgang til arbejdet.

Denne celle er desuden en af de mest usikre på nuværende tidspunkt, da strategiarbejdet netop vil blive udført af folk i alle områder af DR, og dette oplæg er udviklet rent af IT orienterede arkitekter. Denne celle og dens indhold skal forankres på højeste niveau, hvis vi skal gøre os forhåbninger om at den kan slå igennem overalt i DR. Derfor må vi også regne med en større revision inden en sådan forankring kan blive en realitet.

Politikker og standarder

Beskrivelse: På det logiske niveau i strategisøjlen arbejder vi kun med arkitekturelementer, og ingen diagramtyper. Hvor det konceptuelle niveau gik ud på at dirigere organisationens udvikling, er vi nået ned på et mere konkret niveau, hvor det går ud på at udstikke rammerne for organisationen. Det opdeles i tre forskellige arkitekturelementer; politikker, standarder og teknologier.

Arkitekturelementer:

- Politik: En politik er en eller flere regler forankret og kommunikeret til organisationen om hvordan arbejde udføres, eller ting behandles. Der findes mange politikker i de fleste organisationer, eksempelvis personalepolitik, IT-sikkerhedspolitik, lønpolitik m.fl.

- Standard: En standard er en måde at udføre et stykke arbejde på, i modsætning til en regel derfor. Der findes mange både eksterne og interne standarder vi arbejder med i DR. Det være sig eksempelvis W3C html, som er en programmeringsstandard, eller applikationsafdelingens udviklingsmodel, der også er en standard. Herunder kunne ISO 9000 standarder også indekseres.

- Teknologi: Et væsentligt fokus for vores afdeling er at kunne overvåge og relatere hvilke teknologier der anvendes hvor i DR. Det kan dreje sig om high definition TV (HDTV), trådløse teknologier (ex 802 11b) eller enddog en type styresystem (Mac Tiger, Windows XP eller RedHat Linux). Man har i DR for nyligt vurderet om hvordan man skal konvertere sin produktion til 16:9 format og HDTV, og havde man haft teknologierne indekseret og relateret, ville man hurtigt have kunnet skabe sig et overblik over konsekvenser og udfordringer.

Diskussion: Det kan synes mærkeligt ved første øjekast at teknologier er indekseret i strategisøjlen, men ved nærmere eftertanke var det, det rigtige sted. På samme måde som med politikker og standarder er teknologier noget der anvendes stort set universelt gennem organisationen, og noget strategier og mål ofte vil relatere sig til, eksempelvis i ovennævnte eksempel med en ny strategi om at fokusere ensidigt på 16:9 optagning.

Beslutninger

Beskrivelse og diskussion: På det operationelle niveau, og dermed som strategiarbejdets mest konkrete byggeblok, finder man beslutninger. Der er utrolig mange udfordringer hvis man sætter sig som mål at alle relevante beslutninger i en organisation skal dokumenteres og relateres til de arkitekturelementer de har relevans for. Hvordan sikrer man at beslutingerne rent faktisk bliver dokumenteret, for beslutningerne kan være taget af et hvilket som helst organ eller måske egenhændigt af en kompetent leder. Det vil være umuligt at opnå.

At vi mente det ville være umuligt at opnå en komplet styring af beslutninger, betyder dog ikke at man ikke bør arbejde med dem som arkitekturelementer. I DR, vel sagtens som i mange andre organisationer, er meget af det strategiske arbejde rodfæstet i beslutninger, der aldrig er blevet løftet op til standard eller politik. Et godt eksempel herpå er en beslutning der blev taget om, at vi i fremtiden udelukkende skal have Microsoft og Linuxservere, og altså fase vores Unixservere ud. De fleste i BIT ved at det er sådan det skal være, men beslutningen er ikke forankret noget sted som strategi, politik eller standard. Derfor ser vi Beslutning som en måde at forsøge at få dokumenteret de mange strategiske valg der er gjort mht teknologi, men som aldrig er blevet forankret. Det står dog stadig som et åbent spørgsmål hvordan vi skal administrere arbejdet, og hvordan vi skal få folk til at anerkende behovet herfor.

Organisationsstruktur

Beskrivelse: På øverste niveau af organisationssøjlen, finder vi det klassiske organisationsdiagram, hvor de indtegnede organisationsenheder er arkitekturelementer. Det drejer sig her kun om afdelinger, direktører og vertikale ansvarsforhold, hvorfor diagrammet ikke alene kan beskrive en organisation som DR der er matrix-organiseret, med mange ad-hoc grupper, projektteams mm. Se figur 4.1 for et eksempel på et organisationsdiagram.

Arkitekturelementer:

- Organisationsenhed: En samling af personer og roller der enten tilsammen skal opfylde en funktion (ex lønadministration) eller har samme type kvalifikationer (Projektcentret).

Roller og grupper

Beskrivelse: På det logiske niveau har vi netop den måde DR oftest arbejder på; nemlig som et virvar af grupper og roller. Det er på dette niveau at man kan beskrive ansvar for forretningsprocesser, teknologi og de andre celler i det logiske lag. Der findes ingen diagramtyper for denne celle, men rollerne optræder til gengæld på andre cellers diagrammer. Eksempelvis bruges de på det logiske niveau i Processøjlen (på workflowdiagrammer og som aktører på Use Cases), netop for at definere ansvarsforhold.

Arkitekturelementer:

- Grupper: Grupperne er samlinger af medarbejdere der på tværs af linie-tilhørsforhold (hierarkisk placering), arbejder sammen om at gennemføre en bestemt proces, der både kan være en fast gentagende, eller en unik proces.

- Roller: I sit arbejde udfylder alle medarbejdere en eller flere roller. En udvikler kan eksempelvis være både ”udvikler”, ”systemarkitekt” og ”Faglig projektleder”. En effektiv inddeling i roller betyder at man, som tidligere beskrevet, ikke relaterer specifikke personer, men holder det på et rolle niveau. Det betyder en kæmpe effektivisering når organisationsstrukturen ændrer sig (man ændrer blot rollens relation til en organisationsenhed) eller personer skifter job (nyt personelement overtager blot sine roller).

Diskussion: Man bliver nødt til at indse at udviklingen af et komplet defineret sæt roller for en organisation med 3500 medarbejdere vil være en meget stor opgave. Derfor overvejer vi hvilken tilgang vi vil tage. Skal alle DRs medarbejdere og deres roller dokumenteres med det samme? Skal man lade rollerne definere i forbindelse med at projekter indenfor nye forretningsområder bliver gennemført? Eller bør man arbejde med et fast, men simpelt, sæt roller, eksempelvis 50 standard roller for hele organisationen? Vi har valgt en kombination af de to sidste, hvor man i nye projekter afkræves at dokumentere sine forretningsprocesser, og dermed også de roller der udfører dem, men opfordrer til at man gør det på et overordnet niveau. Det bliver dog en stor udfordring at finde det leje, hvor vi får udbyttet af at kunne overskue alle rollers ansvarsområder, og stadig kan vedligeholde samtlige definitioner af rollerne. Det vil desuden blive et forskelligt niveau for de forskellige områder i DR. I BIT har vi implementeret et kompetencesystem hvor man har defineret alle BITs roller, hvorfor denne liste er hurtig og nemt implementeret i arkitekturen. De fleste direktørområder har dog intet i den stil.

Personer

Beskrivelse: En person er enhver ansat i DR, samt eksterne der har ansvar indenfor DR. Der findes omfattende person- og brugerregistreringssystemer i DR, og det mest interessante for DRs EA er lænken fra en person til den række roller han eller hun opfylder. Der er ingen diagramtyper, og kun dette ene arkitekturelement.

Diskussion: Vores centrale EA-værktøj skal ikke pludselig overtage hvad vores Active Directory registrerer i dag, men i stedet skal et samarbejde mellem de forskellige dokumentationsbærende systemer udvikles. Jeg vil på et senere tidspunkt beskrive vores tidlige værktøjsovervejelser, og der gå i dybden med hvor det vil være gavnligt at registrere hvad.

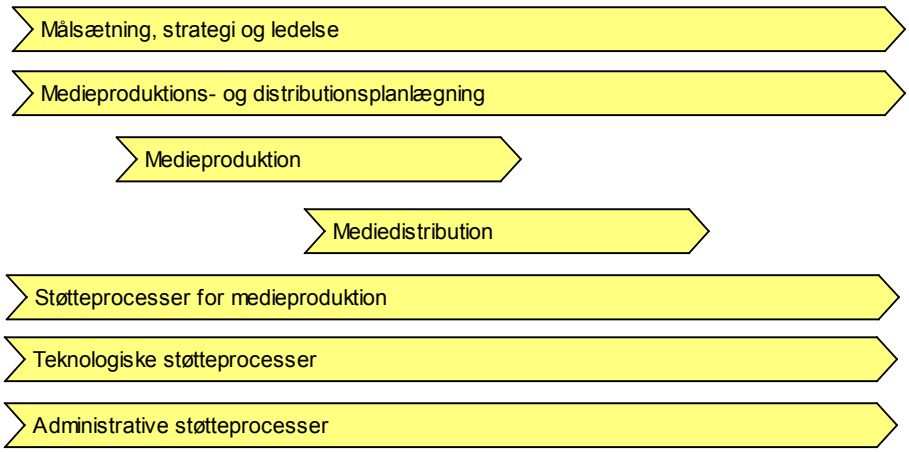
Værdikæde

Beskrivelse: For at beskrive hvordan DR arbejder på et konceptuelt niveau skal der udarbejdes en 3-lags værdikæde (ikke at misforstå med de tre niveauer), der vil stå som eneste arkitekturprodukt for denne celle. Værdikæden forankres hos topledelsen, for at sikre dennes opbakning til vores måde at beskrive vores verden. Der findes kun et arkitekturelement for cellen, nemlig en hovedproces.

Arkitekturelementer:

- Hovedproces: En hovedproces er et symbol der definerer en lang række forretningsprocessers logiske sammenhæng. Meningen er at kunne sætte alle DRs forretningsprocesser hierarkisk op, så man kan gå fra de helt overordnede hovedprocesser (ex Administration) gennem specificerede hovedprocesser (licensadministration) til et konkret overblik over det sæt workflows der tilsammen er licensadministration.

Diagramtype: Til værdikæden anvendes en diagramtype fra vores EA-værktøj, der ikke er standardiseret, kaldet et forretningsprocesdiagram.



Figur 4.4 DRs Værdikæde, øverste lag. Hver af disse hovedprocesser kan brydes ned til hvert sit nye diagram indeholdende processerne på andet lag.

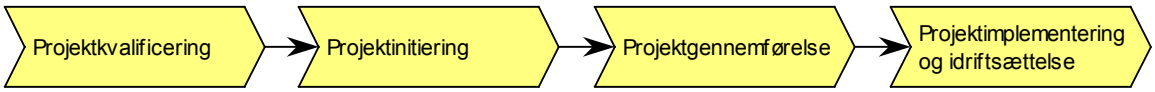
Diskussion: Der findes desværre ingen standard for hvordan forretningsprocesser diagrammeres, men der er de fleste steder en stiltiende forståelse af at en forretningsproces tegnes som en firkant trukket ud som en pil, som på figuren ovenfor. Selve diagramtypen indeholder, når det anvendes på det konceptuelle niveau, kun forretningsprocesser og den manglende standardisering er derfor i dette tilfælde til at bære. Den manglende standardisering og udvikling på området betyder desuden at det blev nødvendigt at bruge både diagramtypen og proces-symbolet på

det logiske niveau, såvel som på det konceptuelle. Symbolet er det samme om man vil visualisere en forretningsproces på et lavere niveau, som en hovedproces. På samme måde findes der ingen særegne symboler for støtteprocesser, hvorfor de også angives om hovedprocesser på det konceptuelle niveau.

Anvendelsen af samme diagramtyper og arkitekturelement-symboler på to forskellige niveauer frygter vi vil forvirre brugeren på længere sigt. For hvornår er noget en hovedproces, og hvornår er noget en forretningsproces? Vi har ikke kunnet finde nogen endegyldige argumenter for det ene eller andet, men praktiske forsøg viste at hovedprocesserne netop passede til at kunne tegnes i tre lag, hvorfor vi valgte at gøre alle diagrammerne på det konceptuelle niveau til et samlet arkitekturprodukt, forankret og godkendt i topledelsen, nemlig som DRs Værdikæde.

Forretningsprocesser

Beskrivelse: Denne celle beskriver hvordan DRs arbejde hænger sammen fra de helt operationelle arbejdsgange og hele vejen op til DRs Værdikæde. På dette niveau kan man både arbejde med workflows, use cases og forretningsprocesdiagrammer. På de nederste lag af det logiske niveau vil man have en række workflowdiagrammer. Hver af disse repræsenteres som en forretningsproces på et højere lag. Dette diagram kan så høre under en samlende forretningsproces på et højere lags forretningsdiagram, og så fremdeles indtil man når DRs Værdikæde.



Figur 4.5 Et forretningsprocesdiagram beskrivende projektarbejde i BIT. Hver af disse forretningsprocesser kan brydes ned til enten et nyt forretningsprocesdiagram eller et workflowdiagram (se figur 4.6 nedenfor).

Arkitekturelementer:

- Forretningsproces: Som beskrevet ovenfor er en forretningsproces et element der omfatter den række processer eller aktiviteter, der fortæller om udførelsen af en arbejdsopgave.

- Workflow: Et workflow, eller arbejdsgang, er en rækkefølge af aktiviteter der logisk kan bindes sammen. Workflowet angiver desuden ansvar for hver aktivitet, samt giver regler for hvordan arbejdsgangen kan variere i forskellige situationer. Et workflow er et arkitekturelement, men det repræsenteres ikke som et symbol, som de tidligere beskrevne arkitekturelementer, men er derimod et diagram i sig selv (se nedenfor).

- Use case: En use case er en systemorienteret pendant til workflowet. Use cases anvendes oftest til systemudvikling, men bliver også mere og mere almindelige indenfor forretningsudvikling. En use case angiver en handling en aktør (menneske eller system) igangsætter i et system. I modsætning til workflowet fortæller den ikke en eksakt vej igennem en arbejdsopgave, men use case diagrammet indeholder derimod samtlige handlinger der kan foregå i en bestemt kontekst. Use case er taget direkte fra UML-standarden[1].

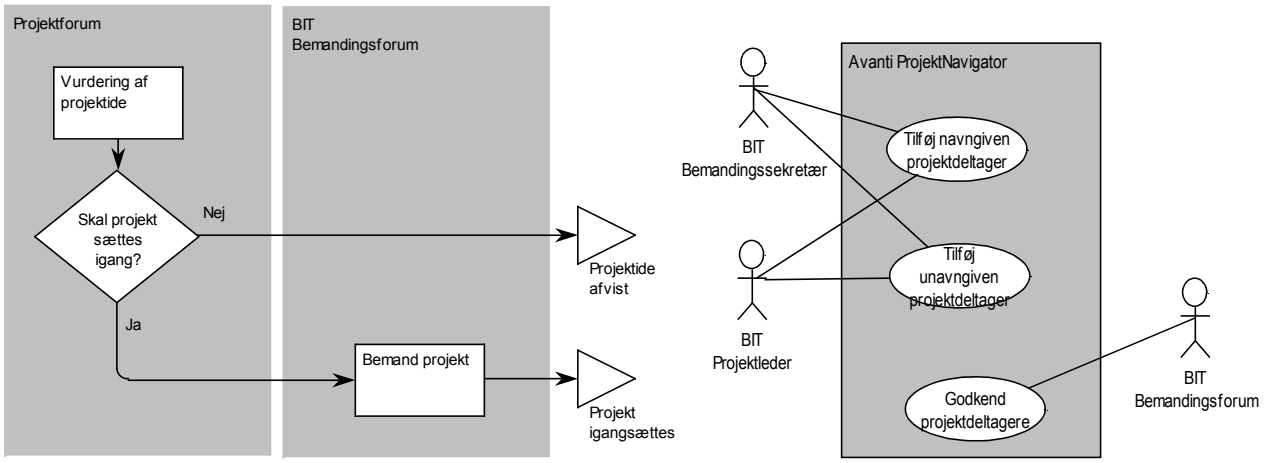
Diagramtyper:

- Forretningsprocesdiagram: Fungerer ligesom på det konceptuelle niveau, med undtagelse af at man på det logiske niveau kan være tjent med at forbinde forretningsprocesserne med pile. Det kan give et godt overblik over hvordan arbejdet i en overordnet proces hænger sammen.

- Workflowdiagram: Workflowdiagrammet er kort sagt det kanvas hvorpå de aktiviteter, roller og hændelser der tilsammen udgør et workflow, tegnes.

- Use case diagram: Et use case diagram indeholder et givent antal use cases der hænger sammen i en større kontekst. Som regel tages der udgangspunkt i et system, og så tegner man alle de aktører og use cases ind, som har med det system at gøre. Drejer det sig om et større system eller sammenhæng, bør der oprettes flere diagrammer. Som tommelfingerregel bør der ikke være flere end 8-10 use cases på et diagram, hvis det stadig skal være læsbart.

[1] Unified Modelling Language



Figur 4.6 Workflowdiagram og Use Case diagram. Workflowdiagrammet til venstre er en nedbrydning af forretningsprocessen Projektinitiering på figur 4.5. Til højre ses en use case baseret på aktiviteten Bemand Projekt på workflowdiagrammet.

Arbejdsgange:

Beskrivelse: På det operationelle niveau af processøjlen drejer det sig primært om hvordan folk rent faktisk arbejder, i modsætning til hvordan de bør arbejde på det logiske niveau. Derfor er det meningen at man også anvender use cases og workflows lig det logiske niveau, men som analyse på det operationelle niveau. Der kan ofte være stor forskel på hvordan man bør arbejde, og hvordan tingene rent faktisk gøres. Vores IT-understøttelse skal naturligvis understøtte arbejdet som det bør gøres.

Der findes altså arkitekturelementerne workflow og use case, samt diagramtyperne workflowdiagram og use case diagram. Se beskrivelsen under Forretningsprocesser.

Diskussion: Det kan synes mærkeligt at vi pludselig opdeler niveauerne efter om noget er analyse eller design, men det giver umiddelbart god mening når man sammenligner med de andre celler i nærheden. Når man analyserer vil man ofte være nødt til at definere at ”Hanne trykker på Gem i licenssystemet, hvilket sikrer at de nye data sendes videre til komponenten OpretKunde”. Det er til gengæld alt for specifikt når vi i stedet vil designe vores processer, der burde det hedde ”Licensbehandleren gemmer den nye kunde, der sendes videre til virksomhedssystemet”. Her arbejder vi med roller istedet for personer (licensbehandler/Hanne), forretningsobjekter i stedet

for data (Kunde/ny data) og systemer i stedet for komponenter (virksomhedssysten/OpretKunde), altså et logisk niveau i stedet for et fysisk/operationelt.

I rammeværket defineres dog ikke disse analyseelementer. Ophøjes analysen til noget man vil arbejde efter, og understøtte med teknologi, hører arbejdsgangene hjemme som forretningsprocesser på det logiske niveau.

Informationsarkitektur

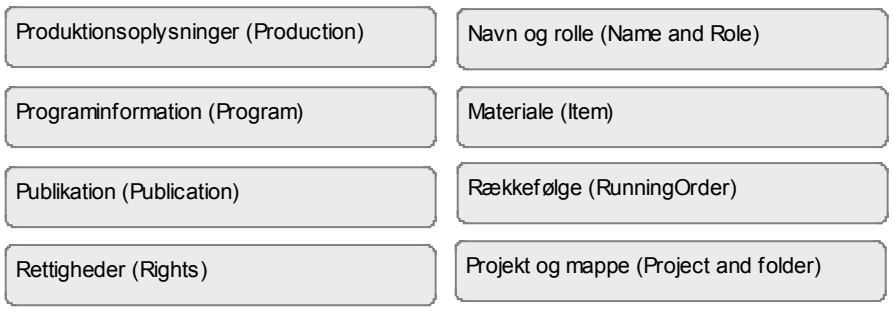
Beskrivelse: I en virksomhed som DR er næsten samtlige produkter der udvikles medierelaterede og de fleste digitale. Det har medført et omfattende fokus på metadata om vores produktioner, der er mundet ud i en imponerende detaljeret metadatamodel[1]. Ud fra denne er det muligt at overskue samtlige forretningsobjekter i DR, sålænge de er produktionsorienterede. Det samme findes desværre ikke for vores administrative områder (løn, økonomi, licensadministration, personale etc.). Metadatamodellen i sin anvendelige form (xml-specifikationer af forretningsobjekter) hører hjemme på det logiske niveau, men for at kunne holde styr på modellen er der tilmed designet et sæt overordnede informationsområder; eks: Programinformation, Rettigheder og Rækkefølge. På dette niveau er det meningen at der skal udvikles en samlet forankret DR Informationsarkitektur, på samme måde som med DRs Værdikæde. Informationsarkitekturen kan dermed på sigt sættes op mod værdikæden og domænemodellen (mere om denne under Systemsøjlen), og disse tre modeller skabe et konceptuelt overblik over DRs arbejdsprocesser, den information de arbejder med og den IT der understøtter uden at blive konkret med hvem der gør hvad, og hvilket system der er valgt til understøttelsen.

Arkitekturelementer:

- Informationsemne: Et informationsemne er en ramme for forretningsobjekterne på det logiske niveau, og dermed en måde for os at sætte vores forretningsobjekter ind i en logisk hierarkisk struktur. Det viser sin effektivitet på det logiske niveau, når man skal finde frem til et bestemt forretningsobjekt blandt mange.

[1] DRs Metadatamodel www.dr.dk/metadata

Diagramtype: Et Datamodeldiagram anvendes både her på det konceptuelle niveau og på det logiske. Her anvendes dog kun arkitekturelementet Informationsobjekt.



Datamodeldiagram over DRs medievendte informationsområder.

Diskussion: På mange områder står vi med udfordringer som her; nemlig at en del af forretningen er voldsomt godt dokumenteret (som metadatamodellen) mens resten er mangelfuld eller slet ikke eksisterende. Det har medført at vi ikke kan søge mod en komplet arkitektur fra dag et, men i stedet ser den største værdi i at finde de områder hvor der er umiddelbar gevinst ved at få dem ind i rammeværket (som med metadatamodellen). Den gamle kliche om at det er de sidste 20% af udviklingen der kræver 80% af arbejdet kan jo også vendes om til at man får 80% af gevinsten ved 20% af indsatsen. Det har gennemgående været en ledesnor for os. Det er utroligt nemt at blive fanget i detaljen, men også meget farligt for arbejdet.

Forretningsobjekter

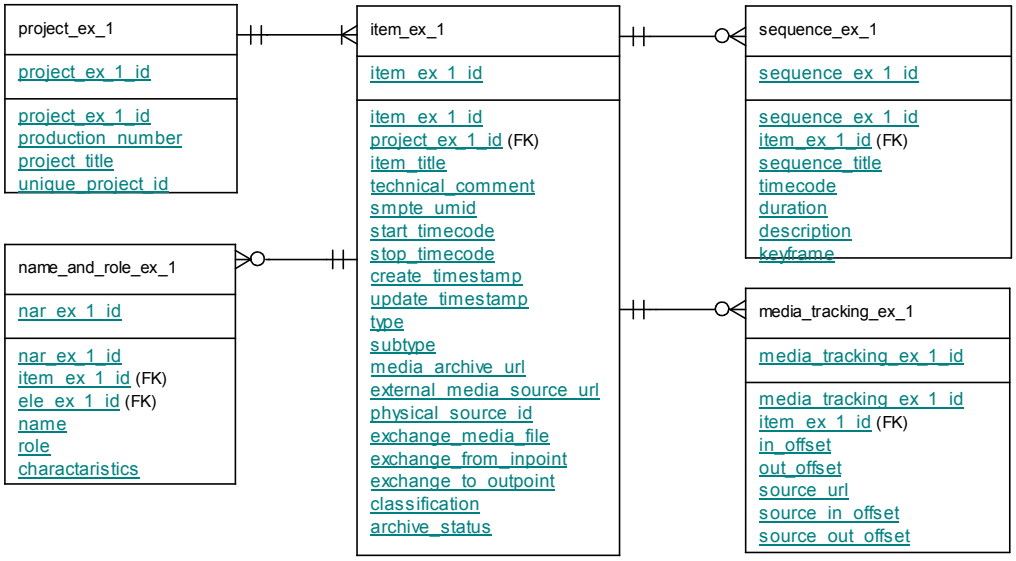
Beskrivelse: Ligesom i processøjlen, adskiller det logiske niveau sig fra det operationelle ved at præsentere virkeligheden som den burde være, ikke som den nødvendigvis er. Det er her udviklere i DR går hen når de har fået til opgave at lave en integrtion til et nyt system, der har brug for et sæt data, for at finde ud af om der findes et forretningsobjekt der opfylder behovet. Gør der ikke det, må et nyt designes og implementeres i rammeværket, som del af DRs metadatamodel. Forretningsobjekter repræsenterer data som det bør samles, pakkes og sendes.

Arkitekturelementer:

- Forretningsobjekt: En samling metadata om et produkt, eksempelvis varighed, rækkefølge og titel for en udsendelse. Formålet med forretningsobjekterne på dette

niveau er at skabe overskuelighed over vores data og ikke mindst de mange integrationer der findes i DR. Man henter ikke længere bare fra andre systemer, men designer i stedet ordnede integrationer, der udveksler forretningsobjekter.

Diagramtyper: Datamodeldiagram, hvor man foruden adgang til informationsemner, kan tegne sine forretningsobjekter ind og deres indbyrdes relationer; hvilke andre objekter de arver fra og hvilke de har relationer til.



Figur 4.7 Datamodeldiagram over ”post-production” forretningsobjekter. Hver tabel angiver et forretningsobjekt der kan blive en del af en integration.

Data

Beskrivelse: Information på et operationelt niveau er naturligvis data. Denne celle repræsenterer hvordan den information der kommunikeres mellem systemer i DR reelt ser ud, i modsætning til forretningsobjekterne der angiver hvordan vi bør strukturere vores data. I den perfekte verden ville Data- og Forretningsobjektcellen være helt ens, men det er urealistisk at sætte som mål, af to årsager. For det første har vi endnu ikke en komplet model over vores forretningsobjekter og for det andet har vi stadig flere hundrede integrationer mellem systemer og mindre applikationer, der fungerer uden nogen form for standardisering.

Diskussion: Så længe man ikke har et detaljeret overblik over ens systemer og integrationer, vil det være meget svært at opretholde et overblik over ens samlede

dataarkitektur, som den ser ud i og på tværs af DRs mere end 400 forskellige systemer og applikationer. Derfor vurderedes det at vi som arkitekter burde fokusere på det logiske niveau, og designe en ønskværdig informationsarkitektur vi kan udvikle os hen imod. Grænsen ned til datalaget vil vi så stille krav til om at overholde det logiske design. Det har den store fordel at vi ikke skal i gang med at forcere et dataoverblik over systemer der naturligt vil blive udfaset, eller udviklet ind i vores service-orienterede arkitektur og DRs integrationsplatform, men kun arbejder med de data der er vigtige for forretningen. Her mister vi altså bevidst det totale overblik, for at sigte mod et niveau vi rent faktisk tror på kan lade sig gøre.

Domænemodel

Beskrivelse: Ligesom DRs Værdikæde og DRs Informationsarkitektur, udmunder denne celle sig i et fast og forankret arkitekturprodukt; DRs Domænemodel. Modellen indeholder en lang række funktionsområder som DRs systemer skal kunne understøtte. Målet for modellen er at understøtte DRs Værdikæde, således at alle processernes behov for teknologiunderstøttelse er repræseneret ved et funktionsområde, eller rettere et domæne, i modellen her. På samme måde som domænemodellen skal opfylde DRs Værdikæde, skal alle DRs systemer indekseres op imod domænemodellen. Det skal altså ende med at vi har en komplet model, og et overblik over hvilke systemer vi har der kan opfylde hvert domæne. I en så teknologisk kompliceret virksomhed som DR er der til næsten hvert domæne flere systemer der kan anvendes, og bliver anvendt. Et eksempel herpå er domænet Videoredigering, hvor såvel Avid som vores to Dalet-systemversioner anvendes, og der findes en række andre systemer i anvendelse, der kan bruges til videoredigering, men ikke bliver det. Denne sammenhæng ser vi som et stærkt argument når vi i fremtiden vil begynde at måle på vores effektive udnyttelse af vores teknologi. Hvorfor skal vi betale penge for 5 forskellige systemer der kan videoredigere? I fremtiden vil det blive en af vores opgave at lægge op til rationaliseringer af teknologi, så vi ender med færrest mulige systempakker.

Arkitekturelementer:

- Domæne: Et domæne skal forstås som en række nært beslægtede opgaver som et system er designet til at løse eller arbejde med. Et system vil oftest arbejde

med flere end et domæne. Denne definition kommer fra det engelske udtryk ”application domain” (oversat ”anvendelsesområde”).

Figur 4.8 Domænemodellen brudt ned til de domæner der har med DRs produktion at gøre.

Diskussion: Domænemodellen var i lang tid til diskussion. Mange kunne ikke se det fornuftige i den, andre forstod radikalt andre ting ved ordet domæne og det var svært at identificere den samlede gevinst ved modellen. Det var først da DRs Værdikæde og DRs Informationsarkitektur var defineret som arkitekturprodukter, at det pludseligt blev klart hvor effektivt et våben domænemodellen kunne blive, når vi skal kunne måle vores teknologianvendelse og tage strategiske beslutninger herom.

Systemer

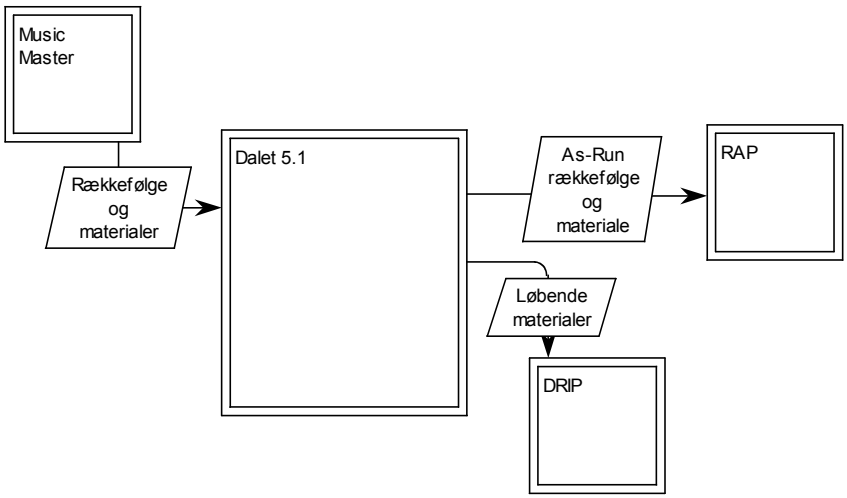
Beskrivelse: Cellen omfatter alle DRs systemer, hvor system forstås som en samling softwarekomponenter der tilsammen og i sammenhæng opfylder en kundes forretningsbehov. Et system afgrænses ved at være hele den løsning der servicerer kunden, understøttet af en samlet infrastrukturservice (se nedenfor). Når en S.L.A. indgås i DR mellem en organisationsenhed og BIT, er systemet det der afgrænser omfanget af aftalen.

Arkitekturelementer:

- System: Et system er en løs samling af komponenter og integrationer der i en sammenhæng opfylder en kundes behov, som defineret i en S.L.A. mellem kunden og BIT. På det logiske niveau beskæftiger man sig udelukkende med systemet, ikke de dele det består af.

- Informationsflow: Information bliver mindre og mindre bundet til proprietære systemer og lukkede databaser, og med udviklingen henimod en SOA virkelighed, kommer information til at eksistere mere og mere i en konstant udveksling. I sin fysiske udformning er informationsudveksling naturligvis integrationer (enten mellem to systemer (ex webservices) eller gennem en integrationsplatform), men det skal også kunne designes på et logisk niveau, hvordan udvekslingen bør se ud. Dertil anvendes arkitekturelementet Informationsflow.

Diagramtyper: På det logiske niveau er meningen at vise et system i den kontekst det eksisterer. Altså vil vi diagrammere systemet på et diagram sammen med de andre systemer det udveksler og deler information med. Der skal eksistere et diagram for hvert system, der har systemet i fokus. Et system kan altså godt findes på mange diagrammer, men er kun fokus for et. På de andre kan det optræde som en del af et andet systems kontekst.



Figur 4.9 Systemdiagram for Dalet 5.1. Med Dalet i centrum vises på diagrammet de systemer Dalet integrerer med og hvilken information der sendes og modtages. De tre informationsflows kunne så linke til et datamodeldiagram som figur 4.7, mens de tre systemer der ikke er i fokus, hver vil have et andet diagram hvor de er det.

Diskussion: Definitionen på system som beskrevet ovenfor, strider imod tidligere tiders definition, hvor grænsen mellem hvad der er et system overfor hvad der var

programmer, komponenter eller applikationer var, og er, meget sløret. Vi søgte en definition der klart kunne adskille systemer, og det logiske niveau, fra komponenter på det operationelle. Samtidig var vi utilfredse med den måde BIT kommunikerer med sine kunder på, hvor der ingen konsensus er omkring hvilke ord man bruger til at definere hvad, hvilket nemt leder til misforståelser. Det ledte frem til at vi ville anvende termer på det logiske niveau, som tager udgangspunkt i en, for vores kunder, logisk terminologi. En kunde skal ikke bekymre sig om hvorvidt hans system indeholder et eller tyve forskellige indkøbte systemer, eller hvordan de indbyrdes arbejder sammen. Kunden skal bekymre sig om hvorvidt hans forretningsprocesser er understøttet af det samlede system BIT har leveret, derfor denne definition af System.

Denne skelnen betyder også at man tager ansvar for teknologivalg helt væk fra kunden, hvilket igen betyder at BIT på længere sigt nemmere kan effektivisere og rationalisere vores teknologi. Hvis kunden ikke har ret til at vælge om han vil have Microsoft Office eller Open Office, har teknologiorganisationen meget bedre muligheder for at tilpasse sig og kan potentielt spare organisationen mange penge. Denne tankegang er dog slet ikke uproblematisk. I DR, som i vel nok de fleste andre virksomheder, har man været vandt til at forretningen krævede hvad de ville af IT-afdelingen, og så måtte man i IT følge med, hvilket oftest har ledt til for få ressourcer til for meget arbejde og udbredt utilfredshed med IT-afdelingen. Nu forsøger vi at vende billedet, så vi stiller krav til forretningen (ingen medbestemmelse omkring teknologivalg) for derved at kunne opnå forretningens overordnede mål om effektivisering.

Endnu et tiltag for at forbedre forholdet mellem forretningsenhederne og BIT, er at man er begyndt at indgå S.L.A.. Det skal klart defineres hvem der har ansvar for hvad, og i hvilken grad, inden ny teknologi indkøbes eller udvikles. Denne målsætning spiller godt ind ift vores arkitektur, da en afklaret ansvarsfordeling hjælper os til at kunne afgrænse systemer fra hinanden, såvel som at få identificeret forretningsejere, fra forretningen, og systemejere fra BIT.

Når nu systemer er defineret som de er, hvad er så de samlinger af software man indkøber, såsom Microsoft Office, Avid redigeringssuite eller Oracle Application? Er de ikke systemer? Umiddelbart var vores holdning da vi stillede os selv dette spørgsmål at det var de jo selvfølgelig! Men det giver faktisk god mening at sige at eksempelvis Microsoft Office er en pakke af komponenter der indgår i en række systemer. Dermed undlader vi at tænke på Office som et samlet system, hvilket, når man tænker på EA som relationen mellem forretning og teknologi, giver rigtig god mening. Det giver os

dog en udfordring når man tænker økonomisk, for Office er naturligvis en pakke, når der skal forhandles licenser, og indgår faktisk i en større forhandling omkring al Microsoft, lige fra Windows til vores serverpark. Hvordan denne side af sagen på sigt vil blive understøttet af arkitekturen, er der dog ikke taget stilling til endnu.

Systemkomponenter

Beskrivelse: På det operationelle niveau i systemsøjlen, beskæftiger vi os med hvordan vores samlede mængde software arbejder sammen. Komponenter er et vidt begreb der kan dække over alt fra webservice, over applikation til databaseserver. Ethvert system vil have et diagram under sig der viser hvordan de forskellige komponenter spiller sammen, for at systemet kan fungere. Med baggrund i vores ønske om at bruge åbne standarder så vidt muligt til at repræsentere vores arkitektur, anvendes UML til al diagrammering i denne celle.

Arkitekturelementer:

- Komponent: De dele et system er bygget op af, eller som stiller sig selv til rådighed for andre systemer eller komponenter.

- Integration: For at definere præcis hvilke forretningsobjekter, eller rettere; hvilke fysiske instantieringer af de logiske forretningsobjekter, der udveksles mellem komponenter, fokuserer vi på arkitekturelementet Integration. En integration på et UML Component Diagram skal altid relatere et logisk forretningsobjekt. Gør det ikke det, overholdes DRs metadatamodel ikke, og man må revidere.

Diagramtyper: For EA er det eneste rigtigt interessante diagram på dette niveau Componentdiagrammer der viser hvordan de forskellige komponenter arbejder sammen. For udviklere, og andre der skal kende detaljerne, er de andre typer UML-diagrammer dog også interessant, hvorfor de også bliver en del af rammeværket og denne celle. Jeg vil dog ikke beskæftige mig yderligere med andre UML-diagrammer end Component og Use Case (se tidligere).

Diskussion: Aldrig en regel uden undtagelser. Selvom vi har som mål at bruge standardiserede diagrammer og symboler så vidt muligt, var vi nødt til at gøre en undtagelse her. Kender man UML vil man vide at der ikke findes et Integrationssymbol, hvilket vi mener er essentielt, for at kunne dokumentere en SOA.

Så selvom UML var det eneste område hvor der fandtes en fuldstændig fast standard, så må vi erkende at den ikke var specifik nok for os.

Infrastrukturarkitektur

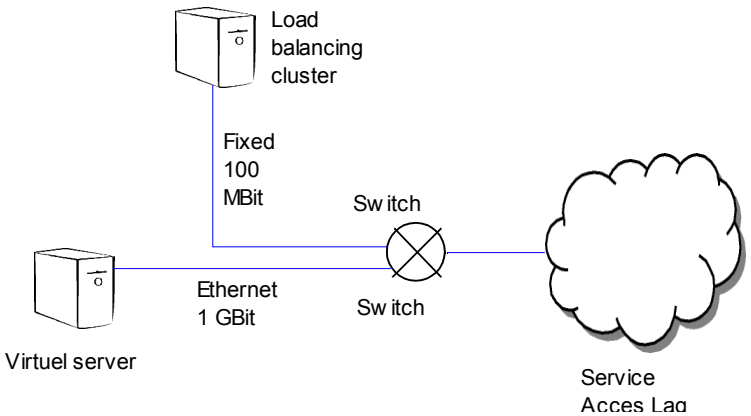
Beskrivelse: Under denne celle beskæftiger vi os med hvilke typer infrastruktur der anvendes i DR. Et overordnet kort over DR fortæller hvor vi eksempelvis anvender trådløse netværk, hvordan DR-Byen er kablet og hvilket broadcastudstyr vi anvender. På dette niveau er vi ikke interesserede i om et trådløst netværk er 802 11b eller 802 11g, men blot at det findes, og på samme måde kan vi repræsentere vores serverpark, dog uden at komme ind på om det er Microsoft eller Linuxservere. Hvordan denne celle skal dokumenteres og hvilke arkitekturelementer vi vil anvende, er endnu ikke afklaret.

Infrastrukturservices

Beskrivelse: En infrastruktur service er en samling infrastrukturenheder der understøtter et system, som defineret i en S.L.A.. For at understøtte et system vil man altid have en række krav i stil med; det skal køre på et Fail Over Server Cluster, der skal være minimum 1Gbit forbindelse mellem komponent A og B etc. Disse krav omsættes til en platformuafhængig løsning, som bliver driftkravet fra kunden til BIT.

Arkitekturelementer:

- Infrastrukturkomponenttype: Ligesom systemkomponenter kan infrastrukturkomponenter være mange forskellige ting; servere, netværk, switches osv. På et Infrastrukturdiagram angiver man disse forskellige infrastrukturkomponenttyper hvilket angiver hvordan en reel løsning skal se ud.



Figur 4.10 Infrastrukturdiagram der viser hvad et givent system kræver af infrastrukturen for at fungere. I dette tilfælde en virtuel server på en 1GBit forbindelse til vores Service Acces Lag, og et Load Balancing Cluster med en 100MBit forbindelse til netværket.

Diagramtyper: Infrastrukturdiagram

Diskussion: Som jeg diskuterede tidligere omkring systemer, er det i fremtiden heller ikke meningen at en kunde skal have kendskab til eller nogen medbestemmelsesret omkring den infrastruktur der understøtter hans system. Indtil nu har man ofte aftalt med kunden at han skulle købe en server til sit nye system, men det fører konsekvent til at kunden mener at have eneret over det stykke hardware han har betalt for. Det er selvfølgelig mere rationelt at lade kunden betale for at blive serviceret, for så kan teknologiorganisationen selv bestemme hvordan infrastrukturen skal opbygges. Derfor skitserer man løsningen på et abstrakt niveau, uden enhedsbetegnelse på infrastrukturkomponenterne. Kunden ser altså at hans system kommer til at køre på en virtuel server der har mindst 100Mbit forbindelse til sin database. Hvilken virtuel server der er tale om og hvilket netværk der anvendes skal kunden ikke bekymre sig om.

Infrastrukturenheder

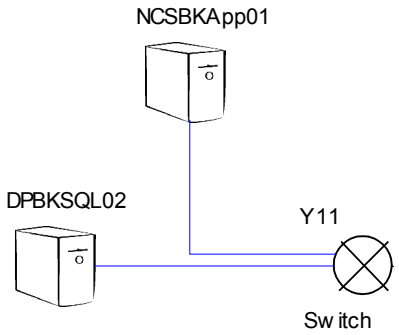
Beskrivelse: På det operationelle niveau opstiller man for hver Infrastrukturservice et diagram over hvilke enheder der reelt gør arbejdet. I eksemplet ovenfor kan vi her angive at det er virtuel server X311 der kører kundens system, og den er koblet til vores primære WAN via switch y11. Dermed kan vi skabe et samlet databaseunderstøttet overblik over vores infrastruktur, blot ved at modellere hvordan vores systemer er understøttet. Når vores diagrammer alle er tegnet ind i vores

databaseunderstøttede EA-værktøj, kan vi nemlig spørge database om eks; Hvilke servere har vi på vores primære WAN? eller hvilke servere anvender switch y11. På den måde vil det blive nemmere for os at udvikle vores infrastruktur i fremtiden.

Arkitekturelementer:

- Infrastrukturkomponent: Alle enkeltdele i vores infrastruktur er infrastrukturkomponenter.

Diagramtyper: Infrastrukturdiagram. Det samme som på det logiske niveau, men her indsættes infrastrukturkomponenter i stedet for infrastrukturkomponenttyper.



Figur 4.11 Infrastrukturdiagram med specifikke enheder i modsætning til figur 4.10 ovenfor. Læg mærke til at service acces laget ikke er medtaget her. Vi vil fra andre diagrammer videre at switch Y11 er koblet på service acces laget, og derfor tegner vi det ikke ind her.

4.5 Den dynamiske dokumentationsmodel

At kalde rammeværket dynamisk henviser ikke til at vi kan eller skal ændre det løbende. Det henviser derimod til at indholdet i rammeværket hænger dynamisk sammen, så man ud fra ens dokumentationsarbejde får langt mere end hvad den enkelte har tastet ind. Fundamentet for at rammeværket kan være dynamisk er naturligvis at der ligger en database under, hvor alle de arkitekturelementer og diagrammer vi dokumenterer gemmes. Det betyder, at sætter vi et arkitekturelement ind på to forskellige diagrammer, er det det samme objekt i databasen. Har vi indsat systemet Avid på flere forskellige diagrammer, kan vi senere ændre karakteristika ved systemet, og det vil slå igennem på alle tre diagrammer. Ændres navnet eksempelvis, vil det dynamisk blive ændret på alle diagrammerne. Eftersom alle arkitekturelementerne er objekter i databasen, kan man også fortælle mere om elementet. For Avid systemet kan vi eksempelvis indsætte en lænke til systemets S.L.A. (et eksternt dokument), skrive en beskrivelse af systemet, angive ejere og ansvar, samt relatere det til andre relevante arkitekturelementer.

At relatere mellem arkitekturelementer og diagrammer giver os mulighed for at få et rigtig godt overblik over vores arkitektur. Et system kan vi eksempelvis relatere på følgende måder;

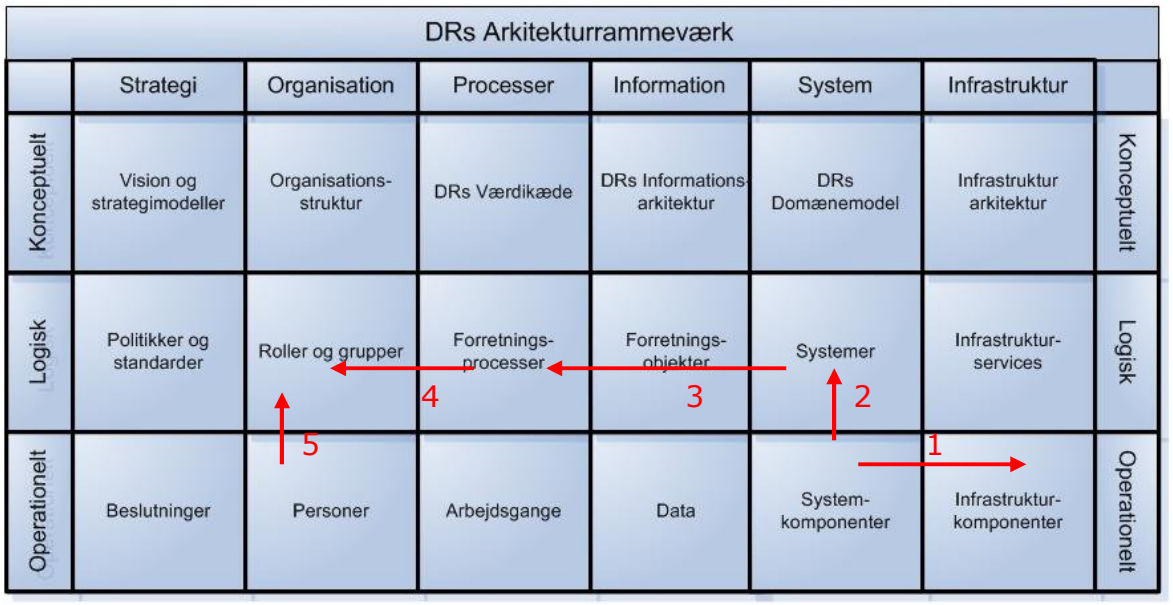
- ”Bryde ned til” et UML Component diagram (fra logisk til operationelt niveau).
- ”Opfylde” et eller flere domæner (fra logisk til konceptuelt niveau).
- ”Kræve” en infrastrukturløsning (fra cellen Systemer til Infrastrukturservice).
- ”Udveksle” information med et andet system (fra cellen Systemer til Forretningsobjekter).

På samme måde kan andre arkitekturelementer relatere sig til systemet:

- En proces ”understøttes af” et system (fra cellen Forretningsprocesser til Systemer.”

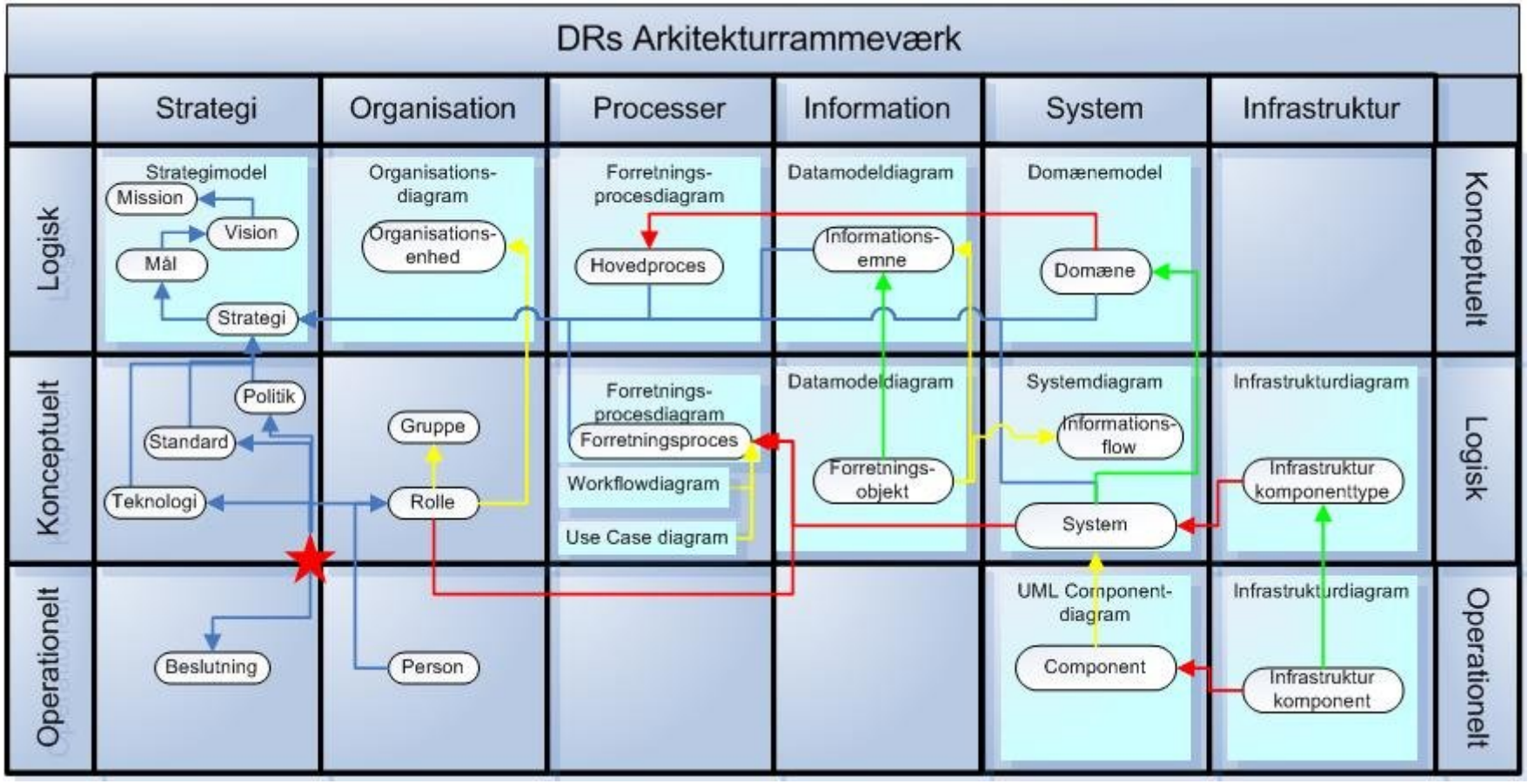
Denne måde at relatere arkitekturelementerne til hinanden gælder hele rammeværket igennem, og betyder at man bevæge sig fra en celle til enhver anden. Vil man vide hvilke personers arbejde er afhængig af om et givent netværk fungerer, kan man lave følgende forespørgsel (se figuren herunder for en visuel eksemplificering):

- Fortæl hvilke systemkomponenter (cellen Systemkomponenter) der kører på en server der er koblet på netværket (Infrastrukturenheder) [1]. Angiv hvilke systemer (Systemer) disse systemkomponenter er en del af [2]. Fortæl hvilke forretningsprocesser (Forretningsprocesser) der understøttes af disse systemer [3]. Hvilke roller er involveret i disse forretningsprocesser [4], og hvilke personer udfører disse roller [5].



Figur 4.12 Eksempel på forespørgsel gennem rammeværket.

Oftest vil forespørgsler være langt simplere, men dette tænkte eksempel viser i hvert fald mulighederne. Det følgende diagram, igen af rammeværket, indeholder samtlige relationer der kan eksistere i rammeværket. Det er værd at lægge mærke til at samtlige arkitekturelementer er relateret til mindst et andet. Det er sammenhængen der er vigtig i arkitekturen, ikke spredt dokumentation.



Figur 4.13 DRs Rammeværk og Dokumentationsmodel. Denne figur viser rammeværket med samtlige mulige relationer der kan skabes mellem arkitekturelementerne og diagramtyperne. Arkitekturelementerne er angivet som afrundede rektangler og diagramtyperne er grønne og uden kant.

Blå relation: Det første element er påkrævet for at det andet element er opfyldt eller overholdt.

Gul relation: Det første element er en del af det andet element.

Grøn relation: Det første element er kategoriseret som eller af det andet elements type.

Rød relation: Det første element understøtter det andet, der ikke kan fungere uden.

I teorien til dette speciale beskrev jeg JP Morgenthals ideer om Enterprise Information Integration, og det er her relevant at vurdere hans tanker i forhold til modellen herover. Rammeværket i baggrunden af modellen er en taksonomisk struktur hvor alle arkitekturelementerne har en, og kun en, celle de hører hjemme i. Dokumentationsmodellen, der er alt hvad der findes i og mellem cellerne, er derimod ontologisk, udfra den betragtning at det er sammenhængene der retfærdiggør modellen. På dette stadie af udviklingen findes der fire forskellige typer relationer, men det vil givetvis blive et større antal ved endelig udgivelse. Disse relationer er, som både figur 4.13 og 4.12 viser, den metadata der giver vores data mening.

Den samlede datamodel kan ikke siges at være normaliseret i nogen væsentlig grad. At prøve og katalogisere en virksomhed, dens kultur, medarbejderne, teknologien og ikke mindst uhåndterbare emner som virksomhedens ”retning” og ”ambition” er nok umuligt hvis man vil kræve en normaliseret datamodel. Vi tror dog også nok at modellen vil være anvendelig alligevel, men man må naturligvis være opmærksom på hvor i modellen der er gjort undtagelser.

I datamodellen ser man en rød stjerne, hvorfra der er pile til Politik, Standard, Teknologi, Beslutning og Rolle. Det angiver at alle andre arkitekturelementer kan relatere sig til dem, på kryds og tværs, samt endda diagonalt. En rolle (logisk) kan jo godt være ansvarlig for en infrastrukturkomponent (operationel) eller et domæne (konceptuel), på samme måde som at domænet og infrastrukturkomponenten kan overholde en politik. Rent tegneteknisk er stjernen anvendt for at undgå at indtegne samtlige af disse relationer.

Et enkelt sted i rammeværket har vi desuden set bort fra, den ellers ret håndfaste regel om, at ingen diagonale relationer må eksistere. Vi kunne ikke komme uden om at det skal være muligt at relatere en forretningsproces til en strategi. Det vil ikke altid være nok at relatere på det konceptuelle niveau, fra Hovedproces til Strategi, da det kan være vigtigt at vide præcis hvilken proces der understøtter en strategi. Holdte man det rent på det konceptuelle niveau, ville man kun vide at en eller flere af forretningsprocesserne under en hovedproces er med til at understøtte en strategi.

5. Analyse

Ideen om dynamisk dokumentation er et forsøg på at foregribe hvad der skal til for at understøtte EA i årene fremover, hvor serviceorientering vil blive en mere og mere integreret del af virksomheden. EA har betydet meget for mange virksomheder, og er stille og roligt på vej til at blive en fast del af enhver større virksomhed. Serviceorienteringen af virksomheden kommer dog til at stille endnu større krav til den viden der i virksomheden findes om ens forretningsprocesser og strategier i forhold til den teknologi der understøtter organisationen, hvorfor ideen til dynamisk dokumentation opstod.

I disse år snakker man meget om service-orienteret arkitektur og virksomheder er på vej mod at basere sig mere og mere på løse komponenter, end på brede leverandørspecifikke systemer. En del af tanken om serviceorienteret arkitektur er at tingene skal hænge sammen hvorfor man ofte snakker om ”information bus” eller integrationsplatform, for at forsøge at overskue og effektivisere ens systeminfrastruktur. Tanken om dynamisk dokumentation er en videretænkning af dette. Det er i denne forfatters øjne ikke nok at integrere hele virksomhedens teknologi, man er nødt til at have det grundigt dokumenteret for at kunne høste de vigtige langsigtede frugter af dette hårde arbejde. Dynamisk dokumentation er ikke kun beregnet til at understøtte teknologidokumentation, men derimod er hele konceptet at den fremtidige dokumentation er hele virksomhedens grundlæggende sammenhæng, lige fra virksomhedens strategi og ned til enkelte services der understøtter bestemte arbejdsgange. Dynamisk dokumentation er altså en måde at kombinere tanken om serviceorienteret arkitektur og EA, hvorfor termen *serviceorientering* eller *den serviceorienterede virksomhed*, vil blive anvendt i resten af dette skriv (Schekkerman).

Den teori der ligger til grund for EA-bevægelsen i disse år er imponerende omfavnende, men ikke desto mindre er den mangelfuld når det kommer til at tænke i det datagrundlag man skal drive sin arkitektur ud fra. Teoretikere snakker om de mange gevinster ved EA, men kommer meget sjældent ind på hvad der skal til for at høste. At have styr på sin forretning er en sætning der ofte høres, men begynder man at tænke over hvad det rent faktisk kræver af datagrundlag, så bliver man nok skuffet hvis man forventer at finde værktøjer indenfor EA-litteraturen. De tre gennemgående teoretiske værker dette speciale tager udgangspunkt i; Carbone, Bernard og Wagter, understøtter ikke udarbejdelsen af et grundlæggende databaseindekseret videngrundlag. For dem er arkitekturprodukter ofte noget der udvikles når der er behov; eksempelvis må en arkitekt bruge tid på at skrive et dokument når ledelsen

eller et projekt kræver det. Den tilgang vil på sigt give masser af arbejde til EA-teamet, men ikke den fulde værdi for forretningen generelt. Det er min holdning at der skal eksistere et grundigt forarbejdet, forankret og fuldt ud integreret datagrundlag, som arkitekter, eller brugerne selv, nemt kan anvende. Det hjælper både til at aflaste arkitekterne, men vigtigst af alt, sikrer det forretningen en sammenhængende viden om forretningen, der kan kommunikeres og genfindes på alle niveauer.

Vælger man at tro på at fremtiden bliver serviceorienteret, og applikationer bliver services der tilbyder funktionalitet til enhver arbejdsgang der har brug for den, er man også nødt til at kigge på konsekvenserne deraf, eksempelvis;

- Skal services understøtte forretningsprocesser, er man nødt til at have styr på disse forretningsprocesser.
- Antallet af alle slags services[1] vil være langt over antallet af nuværende systemer (og det er allerede svært at holde styr på dem).
- Skal virksomheden kunne sætte mål i en sådan situation, må den kunne se konsekvenserne af ethvert retningsskifte.

For at understøtte denne virkelighed vil det ikke være nok at arkitekterne udarbejder et par rapporter, eller har en liste over virksomhedens applikationer. Tværtimod skal arkitekterne kræve en dokumentation der binder hele virksomheden sammen, og være bannerførere for metoderne og rammeværket bag. For at understøtte forretningen optimalt gennem teknologi bliver det simpelthen fundamentalt at man har et videngrundlag der er;

1. Opdateret
2. Relateret
3. Kommunikeret

At viden skal være opdateret kommer nok ikke som en overraskelse for nogen, men hvor mange oplever rent faktisk en virksomhed hvor man umiddelbart kan stole på den viden man finder? Tror man på at ringbindet er blevet opdateret med de seneste ISO-ændringer? Huskede udvikleren at opdatere listen over interfaces sidst han udvidede? Er det stadig de samme mål virksomheden har, som dengang de hængte den flotte plakat op?

[1] En service kan ses som en applikation, komponent eller enhver form for infrastruktur der stilles til rådighed; et netværk, en server eller en række forskellige typer sammenstillet til en service der opfylder kundens behov.

Har man til gengæld en opdateret dokumentation, som folk har lært at stole på, kan utrolig mange folk i hele organisationen spare tid og ressourcer på at dobbeltjekke validiteten af den information man arbejder ud fra. Virksomheden vil dermed minimere omkostningerne derved og sågar blive mere omstillingsparate.

At dokumentation skal være relateret er nok det mest innovative i ideen om dynamisk dokumentation. Selvom man rent faktisk altid fik opdateret ens ringbind, og det var en del af udviklerens aflevering at opdatere kodedokumentationen og direktionen faktisk kom rundt og satte nye plakater op når de havde opsat nye mål, så er forretningen stadig ikke forbundet. Man kan ikke spørge ringbindet hvilke konsekvenser det vil have, hvis den strategi ringbindet understøtter ændres eller forsvinder, eller om det vil betyde at udvikleren så skal lave et nyt interface. Et sådant spørgsmål ville kræve gennemlæsning af hele ringbindet, kontakt til personer der kan lede en videre til information om strategien og så sikkert et maraton for at finde frem til den udvikler der lige ved noget om de interfaces. Ovenstående eksempel vil selvfølgelig variere efter kvaliteten af virksomhedens dokumentation.

Den tredje parameter omhandler vigtigheden af at kunne kommunikere ens arkitektur ud til de relevante interessenter. Det skal være nemt for både sekretæren der skal se en detalje om en af sine arbejdsgange, infrastruktureksperten der skal sætte en ny server op eller bestyrelsesmedlemmet at få information på det niveau de behøver. Det betyder at EA-teamet har et stort arbejde i at designe arkitekturens kommunikationskanaler og -formater, så man tilgodeser både ”informationsfodring”, hvor kolleger gives relevant information, og ”informationstilgængelighed” hvor ens kolleger selvstændigt kan finde den information de behøver.

Er det vigtigt for ledelsen at kunne overskue konsekvenser af deres beslutninger (og det kunne nedsætningen af et EA-team tyde på), er man nødt til at udarbejde et samlet databaseindekseret grundlag der opfylder de tre parametre. Skulle det ikke være yderst presserende for virksomheden nu, så ville det efter min opfattelse blive det i løbet af få år. Tanken om serviceorientering er begyndt at rulle og det er nødvendigt at enterprise architecture udvikles til at understøtte fremtiden.

Jeg har valgt ikke at forsøge at lave en cost/benefit analyse af værdien af en dynamisk dokumentation, da en sådan vil være for utroværdig at arbejde videre på. Der er bred enighed om vanskeligheden af at sætte kroner og øre på værdien af EA, og det samme gør sig i stor udstrækning gældende for værdien af dokumentation. At prøve at vurdere de to faktorer op mod en udregning af hvad serviceorientering vil betyde økonomisk for en forretning, sat overfor de reelle omkostninger derved, mener jeg ville være frugtesløst. Der er for mange ukendte faktorer til at sætte tal på.

Værdien af dynamisk dokumentation må ligestilles værdien af EA og generelt af dokumentation, og beslutningen om at indføre dynamisk dokumentation, vil grundlæggende være en trossag for den enkelte virksomhed. En trossag et højt EA-modenhedsniveau dog vil kunne gøre mere praktisk.

Analysen vil i stedet for kroner og øre, prøve at vurdere hvad det vil kræve at indføre dynamisk dokumentation og komme med nogle generelle anbefalinger til hvordan det kan implementeres. Casen vil blive vurderet i forhold til analysen og belyse specifikke udfordringer, fejltrin og trin på vejen mod en velfungerende dynamisk dokumentation.

De følgende fire underkapitler repræsenterer de analyseemner jeg har identificeret som de vigtigste for at på sigt kunne implementere en dynamisk dokumentationsmodel. De to første emner omhandler metodik for dokumentation, hvor først anvendelsen af rammeværk og arkitekturelementer analyseres og jeg vurderer om i hvor stor udstrækning det er muligt at anvende "in-a-box" løsninger. Findes der et så generelt rammeværk at det kan passe alle virksomheder, og har det så også de passende arkitekturelementer defineret under sig, eller er man nødt til at tilrette eksisterende rammeværk, eller måske opfinde sit helt eget? Det andet analyseemne omhandler den måde man dokumenterer på visuelt. Det er min klare overbevisning at standardiserede diagramtyper er vejen frem for enhver virksomhed, da det kan hjælpe medarbejderne til at fokusere på indholdet af diagrammerne, frem for som i dag, at skulle bruge tid på først at forstå forfatterens diagrammeringsform. Jeg vil her undersøge Jane Carbone, Scott Bernard og Wagter et al´s synspunkter på diagrammering, og forsøge at opsætte en model for hvordan en virksomhed skal gå til systematiseringen af deres modellering.

Det tredje analyseemne er en diskussion af fordele og ulemper ved den dokumentationsmodel jeg med dette speciale agiterer for, overfor den klassiske dokumentbaserede dokumentation.

Det fjerde analyseemne omhandler de mere organisatoriske aspekter af indførelsen af en så omfattende forandring. Hvad skal der til for at medarbejderne accepterer forandringen og hvorledes skal lederne agere for at den dynamiske dokumentationsmodel kan blive en succes.

Disse fire analyseemner skal tilsammen ligge grund til at jeg i næste kapitel kan give en række anbefalinger til hvordan man kunne opbygge en dynamisk dokumentation. Jeg vil løbende relatere det teoretiske til det mere praktiske, som oplevet i casen. Man har aldrig i DR taget en speciel beslutning om at forfølge dokumentationsindsatsen på den måde jeg agiterer for i dette speciale. Men arbejdet i

DR er hvad der har bragt mig frem til ideen om dynamisk dokumentation, og der ligger mange interessante sammenligninger og erfaringer i det materiale.

Inden vi gør i kødet på de fire analyseemner, er det dog nødvendigt at kigge lidt dybere på hvad serviceorientering egentligt er, og hvordan det har påvirket casen.

Serviceorientering i virksomheder

For at forstå hvorfor der må stilles nye krav til EA rammeværkerne i takt med udbredelsen af serviceorientering i virksomheder, vil jeg i dette kapitel beskrive både hvad DRs planer er i forhold til serviceorientering, og hvad serviceorientering betyder for en organisation. Mange ser serviceorientering som ren SOA, og tænker det kun som teknologisk relevant. Som tidligere beskrevet vil jeg i dette speciale fokusere på begrebet serviceorientering, der omhandler hele virksomheden, frem for den teknologisk funderede SOA. Tager man de samlede konsekvenser af serviceorientering op, ser man at det giver store udfordringer for alle involverede; udviklerne skal lære en helt ny metodik og vil blive ansvarlige for at have styr på organisationens services mens det vil blive arkitekternes ansvar at stille dokumentationsværktøjer og -metoder til rådighed så man kan sikre konsistens mellem teknologi og forretning.

Ovenstående giver udtryk for at der er behov for enterprise architects i forbindelse med SOA, men det er der dog ikke enighed om. Meninger om forholdet mellem SOA og EA er mange:

- SOA vil erstatte EA
- SOA vil overflødiggøre EA
- SOA vil være en metode under EA

Det er denne skribents mening at de to er fuldt ud afhængige af hinanden. For at SOA skal kunne lykkes, kræver det effektiv SOA-governance, og for at den slags teknologistyring kan blive effektiv skal arkitekturen (det samlede sæt regler, beslutninger, standarder og politikker vedrørende serviceorientering) være dokumenteret, kommunikeret og efterlevet.

Case: Serviceorientering i DR

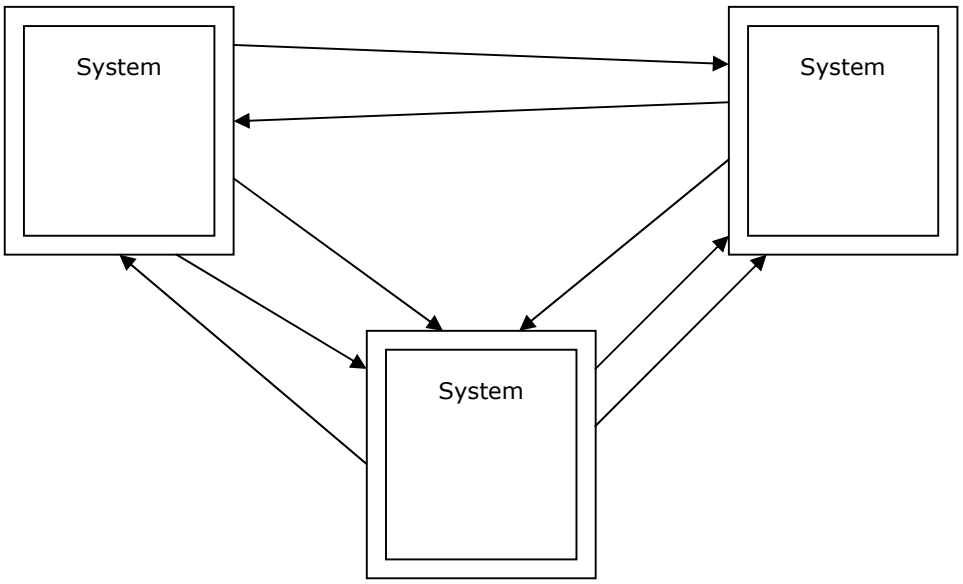
I 1999, da man havde besluttet det man dengang kaldte ”Ørestadsprojektet”, igangsatte man et strategisk arbejde for at belyse hvordan man i fremtiden skulle producere. Man vidste at der ikke ville være samme muligheder i DR-Byen i Ørestad (på nogle områder flere, på andre færre), og derfor måtte man overveje hvordan DR skulle fungere i tiden op til og efter udflytningen. Det resulterede i fem strategiske arbejder;

1. Belysning af DRs brugeres ønsker og behov

2. Fremtidens multimedieproduktion
3. Sammenhængende teknologistrategier
4. Organisation
5. Byggeriet i Ørestad

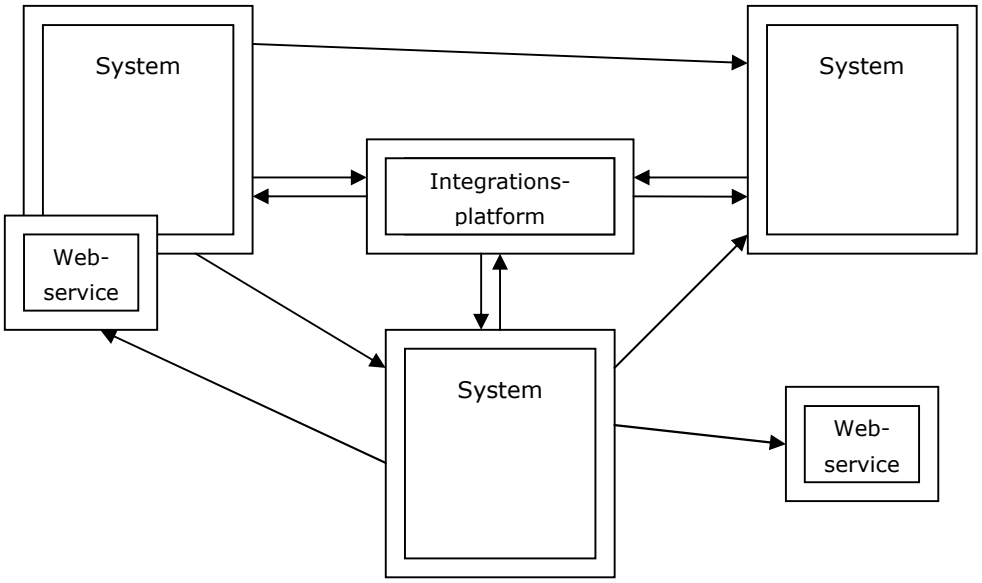
Den tredje rapport er her relevant, da den på baggrund af produktionsstrategien (rapport nr. 2) skulle planlægge hvordan vi ville arbejde med teknologi i en lang årrække fremover. Konklusionen blev at "…kombineret med DRs trimediale forretningsstrategi [må de] nuværende isolerede IT-løsninger omlægges til komponent-baserede løsninger hvor de enkelte komponenter er baseret i lag adskilt med veldefinerede snitflader, hvor komponenter kan genbruges og hvor de samme data kan indgå i flere sammenhænge".[1] Formålet var at det endelige byggeri i Ørestad, og den implementerede nye teknologi, skulle understøtte den måde DR vil arbejde på bedst muligt. Det havde dog naturligvis også konsekvenser for den eksisterende teknologi i TV-Byen, og derfor har man i mellemtiden udviklet en integrationsplatform som først led i en serviceorientering, samt den tidligere nævnte metadatamodel.

Nedenstående tre figurer illustrerer den gradvise overgang fra en verden baseret på proprietære og svært integrerbare systemer, over en midlertidig hybrid arkitektur og endelig en tilstrækkelig serviceorienteret arkitektur.

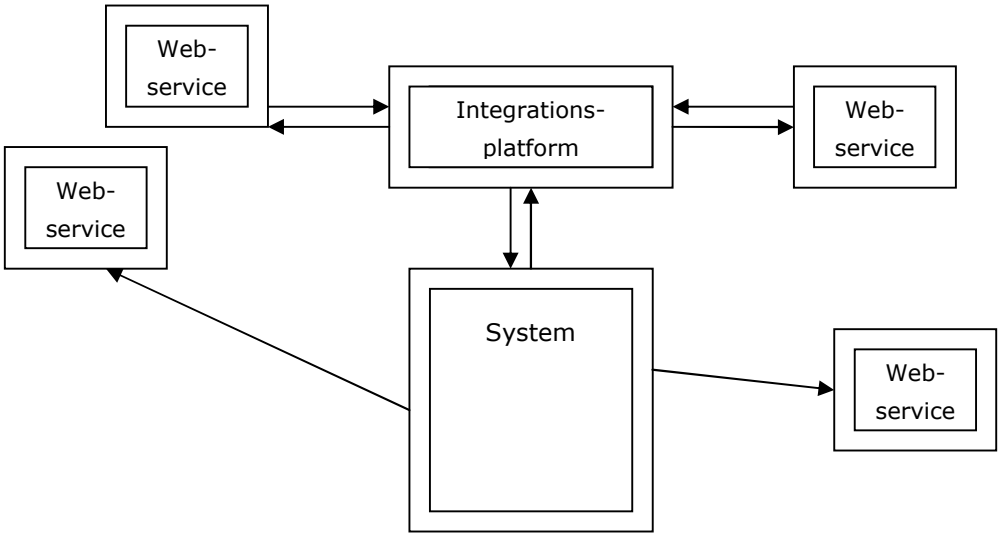


Figur 5.1 viser hvordan DRs systemer før i tiden havde få integrationer, og hvor de eksisterede var de særegent udviklet mellem unikke systemer. Som lederen af DRs udviklingsafdeling kaldte det; "DRs spindelvævsarkitektur".

[1] "Finger C: Sammenhængende IT-strategier" (Internt dokument)



Figur 5.2 viser hvor vi er i dag. Flere og flere integrationer foregår gennem DRs integrationsplatform (DRIP) som ”publish/subscribe” eller i direkte kontakt med webservices som ”request/reply”. Der findes få webservices, men allerede mange integrationer gennem DRIP. Der findes stadig en lang række proprietære systemer, hvorfor en af de største udfordringer i serviceorienteringen er at åbne disse systemer op, så man kan bruge dem som services (angivet ved webservicen der er halvt ude af systemet til venstre i figuren).



Figur 5.4 viser hvordan vores applikationer i fremtiden vil hænge sammen. Der vil være væsentligt færre proprietære systemer, og mange services der både kommunikerer selvstændigt med hinanden gennem request/reply-metoder, samt via DRIP som publish/subscribe.

Hvad betyder serviceorientering for DR?

”Counterintuitive as it may seem, SOA requires more organizational discipline than previous development models” (Windley, 2006)

I DR har vi i dag et sted omkring 4-500 applikationer, systemer eller hvad man nu vil kalde dem. Ingen ved præcis hvor mange vi har, hvor gamle de er eller enddog hvilke teknologier de anvender. De webservices der kommer til at erstatte en lang række af vores systemer, vil naturligvis overstige det antal systemer de erstatter. Lad os gætte på at DR i en fuld serviceorienteret tilstand (om end utopisk) ville have et par tusind webservices, integrationer etc. Hver af disse services vil have en række forretningsregler og interfaces, hvor udfordringen bliver at administrere dem alle.

Citatet der indleder afsnittet beskriver tydeligt behovet for disciplin, men spørgsmålet er hvordan man opnår en sådan?

1. Forankring af serviceorientering i teknologi og forretning
2. Dokumenteret, kommunikeret og efterlevet arkitektur
3. Arkitektur relateret mellem forretning, information og teknologi.

I forhold til dette speciale er det specielt de sidste to punkter der har interesse. Som Windley (2006) skriver, er det urimeligt og urealistisk at forvente at udviklere overholder politikker de ikke er blevet gjort bekendt med. Det betyder naturligvis ikke at man som arkitekt skal gå rundt til udviklerne og fortælle dem om hver ny beslutning, standard eller politik der er vedtaget, men at man må have gjort den information til rådighed på en en samlet og konsistent platform. Den platform er naturligt ens dokumenterede arkitektur.

Som beskrevet ovenfor forbliver serviceorientering kun en teknologisk disciplin, hvis man ikke forbinder serviceunderstøttelsen til de processer der serviceres. Det er ikke, og kan aldrig blive, udviklernes opgave at forbinde teknologi og forretning, hvorfor det vil være at gøre virksomheden en bjørnetjeneste hvis man serviceorienterer sin organisation, styret af serviceudviklernes behov.

Ovenstående begrunder hvorfor serviceorientering må overvejes i forbindelse med EA, og viser tydeligt at serviceorientering sætter store krav til teknologistyring. I de følgende underkapitler af analysen vil jeg vende tilbage til serviceorienteringens betydning for virksomheden under de emner hvor det er relevant.

5.1 Analysemne 1 Rammeværk og arkitekturelementer

Grundlaget for dynamisk dokumentation må man finde i et grundigt forarbejdet og forankret rammeværk og et sæt veldefinerede arkitekturelementer. Rammeværk har siden begrebet EA opstod været en fast del af EA-arbejdet, og de findes i dag i mange former. De fleste baserer sig stadig på John Zachmans (Zachman, 1986) oprindelige rammeværk fra 1986, om end mange nyere er simplificeret i forhold til Zachmans ganske overvældende skabelon. Sam Bernard udviklede modellen i 2004 med sin e^3 kube, der som navnet angiver, er et tredimensionel rammeværk. Andre EA-teoretikere, som eksempelvis Jane Carbone (Carbone, 2004) ligger ikke så meget vægt på rammeværkets opbygning, mens det for Bernard er en helt uundværlig del. Det er en kunst at vælge det rette rammeværk, for der er gode argumenter for både den store og komplicerede som Zachmans, og for de mindre og meget simple, som Carbones. Zachmans 30 rammeværksceller giver uovertrufne muligheder for at indeksere ens virksomhed meget detaljeret, men samtidig må man også overveje om man kan og vil arbejde så specialiseret. Carbone har på den anden side en pointe når hun skriver at jo simplere et rammeværk, jo større er chancen for en succesfuld implementering.

"The framework is intentionally simple. By purposefully limiting the scope of architecture, we are greatly increasing the likelihood that the architecture can be constructed and - just as important - actualized within a reasonable period of time." (Carbone, 2004, s.46)

Om end man ikke må forklejne Zachmans værk, der virkelig var gennembruddet for det der senere kom til at hedde EA, så er et af rammeværkets mest grundlæggende formål at få forretningen til at forstå relevansen af EA. Til dette formål er Zachman for kompliceret. Det er svært for uindviede at forstå Zachmans rammeværk, og selvom de seks søjler næsten er blevet standard i alle rammeværk, så er de fem niveauer simpelthen for komplekse at forstå og anvende. I DR endte vi med et seks-søjlet rameværk med tre niveauer, og det synes folk rimeligt hurtigt at kunne forstå, men det har rent faktisk været overvejet om man skulle lave en endnu simplere, der konsoliderer søjlerne til blot Forretnings-, Informations- og Teknologiarkitektur.

I forhold til dynamisk dokumentation er man nødt til at have et rimeligt udbygget rammeværk, da det er med rammeværket som grundlag at den underliggende datamodel over arkitekturelementer skal udvikles. Her ville Carbones 16-cellede business Framework være for overfladisk. Man kunne overveje at indtænke Bernards tredje dimension *lines of business*, så man for hvert forretningsområde får et selvstændigt rammeværk, men da vi kun beskæftiger os med rammerne for og

typerne af arkitekturelementer, ville disse være de samme for alle forretningsområder. Det er forøvrigt også et godt spørgsmål om ikke denne tredje dimension primært har en politisk funktion, om at hver underdirektør i en virksomhed får sit eget rammeværk. En klog tanke af Bernard, hvis det forholder sig sådan, og en der desværre ikke er nok sider i dette speciale til at gå videre ind i.

Arkitekturelementer er simpelt sagt operationaliseringen af rammeværket. Hver celle i rammeværket er et videnområde, og man må naturligvis have styr på disse områder, for at kunne føre tanken om det fuldt relaterede og integrerede rammeværk til ende. De teoretikere der er anvendt i specialet går så langt som til at lave lister over hvad de anbefaler, eller eventuelt kunne være relevant, at vide noget om indenfor cellen. For at komme skridtet videre og skabe grundlaget for en dynamisk dokumentation, er det dog nødvendigt at grave dybere og lave en datamodel over arkitekturelementerne under hele rammeværket. Det er essentielt at man definerer hvordan en forretningsproces kan understøtte en strategisk beslutning, og hvorvidt et system kræver en infrastrukturservice, eller en infrastrukturservice understøtter et system. Når man senere vil lave dynamiske rapporter og interaktive portaler, er man nødt til at have defineret hvordan de forskellige elementer er forbundne (se figur 4.13), hvorfor en ontologisk fremgangsmåde med navngivne relationer er nødvendig, på trods af at Zachman, Bernard og Carbone alle argumenterer rent taksonomisk. Hvert arkitekturelement skal samtidig have et specifikt symbol associeret til sig. Dermed vil man på virksomhedens mange forskellige diagrammer altid bruge de samme symboler for de samme elementer. Det nytter ikke noget at et system på det ene diagram er markeret med en dobbeltrammet firkant, og på andre som sekskanter eller andet. Udfordringen er så naturligvis at få dette til at passe med de diagrammeringsstandarder man gerne vil anvende, som jeg kommer tilbage til i næste analyseemne.

Arkitekturelementerne kan være meget forskellige. Et element kan være noget konkret, som en fysisk server, mens et andet kan være noget ganske abstrakt som en politik. I arkitekturen kan de dog behandles ens, da de alle er produkter der er vigtige for at integrere og relatere virksomheden på tværs. I udarbejdelsen af virksomhedens dokumentationsmodel må man desuden bestemme hvor langt ned i detaljerne man vil gå for at have dynamiske data. Går man simpelt til værks kan et arkitekturelement være en strategi, mens det i en mere avanceret udgave i stedet kan være en række arkitekturelementer der er kædet sammen (vision, mission, mål, KSF mm). Jo mere detaljeret man bliver, jo større værdi kan man på sigt høste fa sin dokumentation. Hver gang man afgrænser et element og dermed siger at al viden om det element på mere detaljeret niveau står i en beskrivelse, mister man derfra og nedefter fordelene

ved den dynamiske dokumentationsmodel. Man må dog nødvendigvis også erkende at jo mere detaljeret man gør sin objektorienterede dokumentation, jo mere kompliceret bliver implementeringen og den kulturelle forankring af forandringen.

> Case: I DR har vi valgt at detaljeringen på de to øverste niveauer, det konceptuelle og det logiske, skal være dybere end på det operationelle niveau. Vi mente at det var vigtigt at man kan relatere andre arkitekturelementer til specifikke dele af virksomhedens strategier, eksempelvis kan en teknologi være med til at opfylde en specifik del af en strategi (i DR opfylder 16:9 teknologien vores strategiske beslutning om at producere i widescreen format, og denne beslutning understøtter dermed vores produktionsstrategi). Modsat mente vi at det ville blive for stor en opgave at lave detaljering nede på det operationelle niveau. Eksempelvis at kræve en objektorienteret dokumentation af samtlige databasetabeller i DR, mente vi ville blive for krævende at vedligeholde. Dog må man erkende at netop objektorienteringen kan være ekstremt værdifuld på det operationelle niveau, for de folk der arbejder med det. Vores holdning blev at for de afdelinger der kan se værdien af en meget detaljeret dokumentation, vil vi som EA-team være støttende i udviklingen af dokumentationsrammerne og kræve at dokumentation siden opretholdes, men vi vil ikke kræve en så detaljeret fremgangsmåde på nuværende tidspunkt. Det betyder at vi kan få søjler der er dokumenteret meget langt ned i detaljen, mens andre kan synes helt overfladiske i sammenligning, og derfor vil man komme til at mangle relationer fra det mest detaljerede. Har man dokumenteret sin infrastruktur detaljeret, men ikke sine systemer, kan man ikke sætte relationer op for hvilken fysisk server der kører hvilke delapplikationer eller services. Det må komme efterhånden som flere søjler bliver yderligere detaljerede, og indtil da, må relationerne eksistere primært på det logiske niveau.

Ud fra ovenstående kan det synes svært at se nogen *in-a-box* løsninger, da de inddragne værker alle højst når til at anbefale at man laver lister over hvad ens arkitekturprodukter bør være. Der er desuden forskellige opfattelser af hvor generiske ens arkitekturelementer tilsammen skal fremstå. Bernard og Zachman virker til at ville have styr på hele forholdet mellem forretningen og teknologien, mens Carbone mener at listen over ens arkitekturelementer bør reflektere hvilket fokus virksomheden har, og eksemplificerer dette med at en netværkscentreret virksomhed har fokus på de arkitekturelementer der understøtter dette aspekt af forretningen. Derfor kan der ikke siges at være nogen form for konsensus om hvad arkitekturelementer skal være, hvor generisk de samlet set skal beskrive forretningen, og hvor mange man bør sigte efter. Det er også min opfattelse at alle tanker om *in-a-box*-løsninger indenfor et så kompliceret felt som EA, er lige vel optimistiske. Men det betyder på den anden side ikke, at man skal lade være med at prøve at lave generelle modeller, da de i det

mindste kan fungere som gode eksempler. Vil en virksomhed i gang med dynamisk dokumentation, kunne det være en hjælp at tage fat i en eksisterende dokumentationsmodel og arbejde videre ud fra den. Man kan nok aldrig komme ud over at en dokumentationsmodel for en virksomhed vil afspejle bagmændenes perspektiv. DRs dokumentationsmodel viser rimeligt tydeligt at det vi desperat forsøger, er at skabe overblik for bedre at kunne styre både teknologi (-indkøb, -udvikling og -integration) og forretning. Det er svært for en intern IT-organisation at sige nej, når et direktørområde pludseligt vil noget andet end planen var, men kan man henvise til konsekvenser for teknologi, processer, politikker og strategier, har IT-organisationen rent faktisk en mulighed for at tage styringen selv.

Et af udgangspunkterne for dette emne var at se om der fandtes *in-a-box* rammeværk man kunne implementere for at hjælpe en mod en bedre dokumenteret virksomhed. Der findes efterhånden mange forskellige typer rammeværk, men de er alle udtryk for holdninger, og ingen af dem understøtter på nogen måde deres arkitekturelementer. Et rammeværk skal være fasttømret og vidtfavnende, og samtidig smidigt og i konstant udvikling. Det peger på at man, selvom et standard rammeværk vælges, selv må videreudvikle det efterhånden som tidernes fokus skifter.
Der findes ej heller nogen *in-a-box* løsning for dokumentationsmodeller. Dette giver næsten sig selv da de undersøgte teoretikere højst går så langt som til at lave lister over hvilke arkitekturelementer der kunne indgå i ens rammeværk. I relation til en dynamisk dokumentationsmodel er det dog essentielt at rækken af arkitekturelementer præsenteres i en relationsmodel, frem for som en liste. Det er sammenhængene mellem arkitekturelementerne der skal gøre arkitekturen levende i en serviceorienteret fremtid.
For at vende tilbage til Wagter et als udmærkede ide om at al IT- og forretningsudvikling har to modsatrettede interesser; sammenhæng overfor smidighed, så skal rammeværket og dokumentationsmodellen kunne begge dele. Et fasttømret rammeværk og en detaljeret dokumentationsmodel, skal være udgangspunkt for at forholdet mellem forretningen og teknologien bliver så smidig som muligt.
Tilsidst er det også værd at notere sig at tanken om *in-a-box* løsninger indeholder den indbyggede svaghed, at man ikke arbejder så intenst med ens rammeværk, som hvis man selv udviklede det, eller blot tilrettede. Man skaber en stor forståelse for rammeværket og de mange detaljer ved at arbejde sig frem til en egen løsning, om end det altid kræver mere tid.

5.2 Analyseemne 2 Modelleringsstandarder

Det har altid ligget i menneskets natur at tegne for at gøre sig forståelig. Langt tid inden der fandtes nogen form for skriftsprog, malede man storslåede malerier på hulevægge. Med videnskabernes udvikling begyndte man at opsætte bestemte regler for hvordan man tegnede. Man havde fundet ud af at man skulle have et regelsæt for hvordan tekniske tegninger så ud, så alle kunne forstå dem. I løbet af historien er anvendelsen af regelbundne tegninger, der i stedet bør kaldes diagrammer og modeller, blevet mere og mere anvendt, og i dag er det sjældent at se et videnskabeligt eller fagligt skrift uden den ene eller anden form for diagram eller model.

Op gennem de sidste 1000 år er det ellers skriftsproget man har udviklet mest. Mens skrift er uovertruffent til at beskrive og fortælle de fleste ting, er tegninger dog bedre på visse områder. Diagrammer kan kommunikere komplekse sammenhænge langt mere effektivt end skrift. Man skal blot forestille sig at Jørn Utzon havde skrevet sin vision ned for Operahuset i Sydney, i stedet for at modellere og tegne, for at se gyldigheden at dette argument.

Ydermere er der indenfor de sidste 30 år kommet meget fokus på anvendelsen af visuelle kommunikationsformer indenfor både forretnings- og teknologiudvikling. De, for mange, frygtede rationaliseringer der hærgede markedet for 15-20 år siden var startskuddet for workflowdiagrammer, mens man siden computerens tidligste dage har brugt diagrammer til at overskue de komplekse sammensætninger i en sådan. I dag har vi nået et stadie hvor utrolig meget faglig information kommunikeres gennem diagrammer. Det er svært at få folk til at læse et længere skriv inden et møde, men et diagram der fortæller samme historie er mindre uoverkommelig i en travl hverdag. Hvad der dog mangler, er regler for hvordan man tegner sine diagrammer og modeller. Skriftsproget er efterhånden meget standardiseret (om end det konstant udvikler sig), og det er på høje tid at man begynder på det samme for diagrammer og modeller. For selvom modeller kommunikerer sammenhænge bedre end tekst, så bruges der stadig alt for megen tid på at forstå diagramsyntaksen, hver gang man læser et nyt diagram. Alle os der sidder og tegner mange diagrammer for at gøre os forståelige, har hver vores måde at tegne dem på. Nogen tegner kun i sort hvidt og meget stilistisk, mens andre vælger de tykke streger og altid farvelægger dem. Alt

efter forfatterens kreativitet er der så ofte lagt en række WordArt[1] eller billeder fundet på nettet, ind på diagrammet.
For læseren betyder det at man til stadighed skal forstå "et nyt sprog" hver gang man kigger på et nyt diagram.
Udover de rent visuelle ting på diagrammet, har man ej heller nogen standarder for hvad man fortæller på diagrammerne. Nogen tegner et samlet diagram for et projekt der på samme tid kommunikerer problemet, situationen i dag, situationen i fremtiden og så sandelig også hvordan projektet løser problemerne. Det svarer, lidt groft sagt, til at dette speciale ingen overskrifter havde, og blev afleveret sammenblandet uden sidetal på siderne.
Måden at overkomme dette på, er at bestemme sig for hvordan man vil modellere i sin virksomhed. Det er målet med denne del af analysen at kigge på hvad der findes af modelleringsstandarder, hvad teoretikerne anbefaler og på den måde nå frem til et sæt anbefalinger for at opsætte en samlet modelleringsstandard for ens virksomhed.

De, i dette speciale, anvendte teoretikere udviser ikke stor interesse for dette felt. Størst opmærksomhed finder vi hos Jane Carbone, mens Scott Bernard ikke vil forholde sig til emnet og Wagter et al udviser en komplet ignorans overfor det. Sidstnævnte kan måske undskyldes da deres værk omhandler den processuelle side af EA udelukkende. Men at udgive et værk befolket med de tændstikmænd man finder i WordArt kataloget uden nogen form for syntaks eller regler, synes dog noget ugennemtænkt. Bernard undskylder sin manglende holdning til emnet, med at der er meget bedre bøger på markedet om modellering indenfor forretningsudvikling, systemudvikling og lignende fagområder. At han ikke ser relevansen i at vurdere modellering i forhold til hele EA spektret synes underligt, for flere og flere modelleringsstandarder kan netop anvendes indenfor flere fagområder. Det skal dog retfærdigvis nævnes at han skriver at valget af modelleringsstandarder skal tages i et forum bestående af chefarkitekt, EA-teamet og relevante interessenter, hvilket tyder på en hvis form for anerkendelse af modelleringsstandardernes vigtighed. Carbone beskæftiger sig en del med modellering, og opsætter endda seks regler for virksomhedens diagrammer. De to vigtigste pointer hun kommer frem til er at der bør være et symbol for hvert arkitekturelement, og at man skal kunne indtegne alle typer symboler på alle diagrammer. Det sidste betyder så også at hun kun har en type diagram.

[1] WordArt er en funktion i Microsoft Office der indeholder en lang række tegninger, billeder og symboler man kan fremsøge og indsætte i tegninger.

Den første pointe om forholdet mellem symboler og arkitekturelementer er også en integreret del af den dynamiske dokumentationsmodel. Carbone kan opsætte denne regel da hun selv definerer rammerne for hendes anbefalede modelleringsstandard. Vælger man i stedet at anvende flere forskellige standarder, interne som eksterne, volder denne regel dog hurtigt problemer, som jeg vil komme nærmere ind på nedenfor.
Den anden pointe er i lodret uoverensstemmelse med holdningen bag dette speciale. Carbone anbefaler at man kan indsætte alle typer symboler på et diagram. Hun mener det hæmmer overblikket hvis man opsætter regler derfor. Man skal altså kunne indsætte samtlige af hendes arkitekturelementer på ethvert diagram. Det synes ganske ødelæggende for forståelsen af diagrammerne, da man på ingen måde kan styre hvad folk prøver at fortælle gennem deres diagram. Hvorfor en slutbruger skal kunne optræde på et diagram der omhandler virksomhedens back-end systemer, mener jeg ingen mening giver. Ydermere argumenterer hun selv for at man ikke må tegne ”apples and oranges”, dermed forstået at kunne fortælle historier der er specifikke for bestemte fagområder. Hun sammenligner med en arkitekt der tegner en tegning af et halvt hus med kloaksystem, i stedet for at tegne to forskellige tegninger. Dette lyder i denne forfatters ører som en selvmodsigelse, for det er da netop hvad det vil føre til, hvis man må putte et hvilket som helst arkitekturelement på et diagram. Havde man nu, som dette speciale agiterer for, en type diagram til at tegne huset, og en anden til kloakeringen, ville det problem aldrig opstå. Hun når aldrig frem til hvordan hun vil sikre at folk så ikke fortæller disse sammenblandede historier ud fra hendes model.

Da der således ikke er meget hjælp at finde blandt lærebogsforfatterne på feltet, var det i forbindelse med casen, hårdt arbejde der var vejen fremad. Nedenfor beskrives derfor de overvejelser der ligger til grund for DRs modelleringsstandard, samt beskrivelser af casen, og til sidst gives en række spørgsmål relevante for den der ser relevansen i en forankret modelleringstandard.

Det vigtigste for at opsætte modelleringsregler er først at have en forankret dokumentationsmodel, da den kan være et godt udgangspunkt. I en perfekt verden skulle der være komplet overensstemmelse mellem dokumentationsmodellen og modelleringsstandarden. Arkitekturelementerne skulle alle have et fast symbol i modelleringsstandarden, og de i dokumentationsmodellen indtegnede relationer, skulle svare til den måde man måtte sætte pile og streger mellem symbolerne. Dette er dog desværre ikke helt muligt, da;

1. Arkitekturelementer symboliseres forskelligt i de mange generelle modelleringsstandarder der findes i dag (UML, BPMN m.fl.)
2. Nogle gange bliver symbolerne på diagrammerne for specifikke til at være arkitekturelementer, hvor i stedet diagrammet bliver et arkitekturelement.
3. Det er nødvendigt at indtegne forbindelser på diagrammer der ikke er standardiserede.

Case: For punkt 1 har vi i casen to eksempler derpå, begge mellem vores generelle modelleringsstandard og så UML-standarden. På workflowdiagrammer anvender man et symbol der hedder *Rolle* for at angive hvem i organisationen der udfører en given aktivitet. På et UML Use case diagram anvendes man på samme måde *aktører* til at tilkendegive hvem der igangsætter en Use Case i et system. Rolle og Aktør er, set fra arkitekturens side, det samme. Så må man vurdere hvad man mener, er bedst i den situation; at omdøbe et af symbolerne hvilket vil betyde at en af standarderne nu ikke længere er helt standardiseret, eller lave en teknisk løsning der automatisk konsoliderede de to typer i databasen. Den sidste metode betyder dog samtidig at der ikke i folks hoveder skabes sammenhæng mellem de to. Samme udfordring har man mellem System ikonet vi anvender på samtlige diagrammer, undtagen UML-diagrammerne. På UML Use Case diagrammer indsætter man et *System Boundary* symbol, som er det system Use Casene sker indenfor. På samme måde som før er System og System Boundary samme arkitekturelement, repræsenteret på to forskellige måder. Det vil her ikke rigtig give mening at adoptere UML-udtrykket, så alle skulle gå rundt og snakke om System boundaries, i stedet for bare systemer. Ændrer vi derimod navngivningen i UML, følger vi pludseligt ikke denne markedsførende standard længere, og for hver tilretning bliver vores UML-diagrammer sværere at forstå for andre.
I DR nåede vi, i forhold til punkt 2, frem til at man på nogle områder skulle repræsentere arkitekturelementer som hele diagrammer, frem for som enkelte symboler. Det hænger sammen med vores umiddelbare holdning til hvor detaljeret vi vil kræve vores arkitektur skal blive, hvor vi har besluttet at man skal kunne gennemskue hvilken teknologi der understøtter en arbejdsgang (workflow), og ikke helt ned på aktivitetsniveau. Derfor er det mere logisk for os at repræsentere disse arbejdsgange som workflows. På disse diagrammer optræder så en masse symboler der ikke er arkitekturelementer, om end de kan være af stor værdi for arkitekturen. Det vender tilbage til diskussionen om krav til detaljeringsniveau i kapitel 4.
Punkt 3 omhandler ovenstående problematik. Man kan ikke udelukkende tillade relationer fra dokumentationsmodellen, når man anerkender symboler der ikke er arkitekturelementer. Holdte man stædigt på det, ville man eksempelvis ikke kunne indtegne forbindelserne

mellem aktiviteter på et workflowdiagram. Her valgte vi at følge værktøjets standard, hvilket var den nemmeste, og stadig grundlæggende fornuftige, måde at håndtere den sag.

Punkt 1 i casen herover, peger ind på en interessant problematik i denne forbindelse. Der findes mange modelleringsstandarder på markedet i dag, men hvilke skal man vælge og hvilke konsekvenser får det for arkitekturen? Den grundlæggende fordel ved markedsstandarderne er at de er en god måde at kommunikere med eksterne interessenter, partnere eller leverandører og at de er udviklet med et meget stort erfaringsgrundlag. Vælger man en fornuftig standard (der er, eller bliver, dominerende på markedet) har man pludseligt en meget nem måde at kommunikere med sine omgivelser på. Det kunne eksempelvis være UML-standarden. Alle udviklere kender, eller kommer til at kende, denne standard rigtig godt. Den anvendes indenfor næsten alle teknologiske aspekter og er blevet modelleringsstandard for de fleste store projekt- og udviklingsmodeller[1]. Derfor er netop denne standard svær at komme udenom, men implementeringen af den er ikke problemfri, som casen ovenfor viser. På andre områder er valget ikke så ligetil. Dokumentationen af forretningsprocesser er aldrig rigtigt blevet standardiseret gennem en certificeret modelleringsstandard, men foregår ud fra en generel konsensus om hvordan sådan nogen bør se ud. Om man kalder diagrammerne for workflow-, swimlane-, activity- eller procesdiagrammer så er meningen altid at opsætte en rækkefølge af aktiviteter for at kortlægge en arbejdsgang. Det gør området meget svært at standardisere i en virksomhed, da folk har mange meninger om hvordan sådanne bør se ud.

Case: Vi valgte at tage udgangspunkt i Gane & Sarsons klassiske workflowdiagrammer af to grunde. Diagrammet var det der kom med vores dokumentationsværktøj, og der var alligevel ikke nogen fornuftig standard at ligge sig opad på det tidspunkt. At vi valgte denne type fordi vores værktøj understøttede det, er naturligvis ikke nogen saglig begrundelse. Men da vi samtidig ikke kunne se en relevant standard at satse på fremover, valgte vi altså leverandørens. Også fordi at den standard vi vurderer til at blive den fremherskende, både indenfor system- og forretningsudvikling, er blevet lovet som en del af en fremtidig udgave af vores værktøj, med mulighed for automatisk konvertering fra deres nuværende type workflowdiagram. Den kommende standard er BPMN, der ser ud til at blive UML´ens pendant indenfor forretningsmodellering.

[1] Eksempelvis er UML standard for Unified Proces (UP) og eXtreme Programming anbefaler Use Cases til at beskrive brugerens historier.

At vi i DR har valgt at se tiden an og forventer at satse på BPMN standarden, er baseret på omfattende research-arbejde i EA-teamet. Det er umådeligt svært at gætte sig til disse ting, men vi nåede dog frem til en række råd og spørgsmål man bør stille sig selv:

1. Har standarden vundet indpas på markedet endnu?
2. Hvilke eksisterende udviklings-, projekt og forretningsudviklingsmodeller anbefaler denne modelleringsstandard?
3. Hvem står bag standarden, og er den certificeret?
4. Har *the big guns* valgt en standard de satser på?
5. Hvordan vil den spille sammen med den eksisterende modelleringsstandardi virksomheden?
6. Hvor langt vil vi gå for at følge de valgte standarder?
7. Stol aldrig på en konsulentvirksomhed.

Det første spørgsmål skal vise om standarden måske bare er en meget *hypet* modelleringsstandard, uden egentlig anvendelse. Dette undersøges gennem de næste tre spørgsmål. At overskue om en standard kan gå hen og blive markedsledende eller i hvert fald en større spiller på markedet, er det vigtigt at den allerede inden man vælger den, er blevet optaget i de store standarder for udvikling, projekter og forretningsudvikling. UML er, som nævnt, basis for modellering indenfor UP, hvilket er et stort plus i udvælgelsen. Det kan også være relevant at overveje hvem der har udviklet standarden og vedligeholder den, samt om den er certificeret. Visse virksomheder er så store på markedet at de forsøger at presse deres egne, proprietære, standarder frem, og gennem deres størrelse og indflydelse, *hype* standarden så meget at den bliver en generel standard. Er standarden i stedet udviklet og certificeret af de store uafhængige organisationer som OpenGroup, Object Management Group og ISO, kan man være sikker på en vis kvalitet og markedskendskab.
Virkeligheden er også at de store spillere på dette marked, er afgørende for hvilke standarder der vælges. Peger de største konsulenthuse, eksempelvis IBM, Deloitte og Accenture, på samme standard, er den pludseligt meget svær at komme udenom.
De næste to spørgsmål går internt på EA-teamet, og dermed virksomhedens, indstilling til modelleringsstandarder. Når man evaluerer en mulig ny standard, er det vigtigt at måle hvor godt den passer ind i ens eksisterende standarder, og måske endda overveje hvor langt man i virksomhedens vil gå for at overholde markedsstandarderne. Det kan være meget tillokkende at modificere

diagramsyntakser, men umådeligt svært at ændre dette senere, så det må overvejes nøje. Har det stor værdi for virksomheden internt at kunne indsætte et ikke-standardiseret symbol på et standarddiagram, kan det være en god ide at lade dette forekomme. Så må man dog sikre sig at disse ændringer kommunikeres til eksterne interessenter. Det sjette råd lyder en smule hårdt, men det antyder ikke at konsulenthuse bevist søger at føre kunderne bag døren. Man må bare erkende at konsulenthuse også har mange faktorer der bestemmer hvad de vil anbefale. Kan konsulenten kun den ene standard, vil man foreslå den, eller understøtter et værktøj konsulenthuset gerne vil sælge en anden standard, vil det være konsulenthusets anbefaling.

Den eneste vej til en fornuftig modelleringsstandard er altså hårdt arbejde, udført af EA-teamet selv. En grundig udvælgelse er til gengæld en god investering, efter denne forfatters mening. Udover at skabe grundlaget for præsentationen af ens dynamiske dokumentation, kan et godt udgangspunkt også sikre at man ikke behøver ændre i sin standard i lang tid fremover, og det vil på sigt spare tid og ressourcer for hele virksomheden.

For en fungerende arkitektur kan et velstruktureret diagram blive et rigtig godt *view*[1] ind i arkitekturen. Opnår man den dynamiske dokumentation, hvor det er muligt at finde alle arkitekturelementer frem, og se deres relationer, kan det stadig være vigtigt at præsentere en række arkitekturelementer sammen for at kommunikere en bestemt historie.

Med en grundig modelleringsstandard opnår man mange ting for virksomhed og medarbejdere. Diagrammer bliver lettere læselige da folk kender symbolerne, medarbejderne får en fast måde at kommunikere på visuelt og virksomheden kan i forbindelse med dynamisk dokumentation høste viden på tværs af diagrammerne. Men som beskrevet ovenfor, er det dog en større udfordring at udvikle en standard der både tilfredsstiller interne krav og overholder markedsstandarder, så man fremmer forståelsen mellem ens virksomhed og leverandører.

[1] Termen kommer fra databasemodellering, hvor et view er en samling af tabeller og entiteter til en fornuftig kommunikativ enhed. Derfor anvendes termen også her, da diagrammet på samme måde bliver en samling af det arkitekturmateriale man fremfinder fra den underliggende database, for at præsentere det som en samlet historie.

5.3 Analyseemne 3 Dynamisk dokumentation overfor dokumentbaseret dokumentation

I de fleste virksomheder er situationen den samme mht dokumentation. Det er kun de tekniske aspekter af virksomheden man dokumenterer (såsom IT og lagerføring), og dokumentation er, hvis ikke papirbaseret, så stadig dokumentbaseret. Skal man sikre at folk til alle tider har den opdaterede dokumentation, oftest i forbindelse med ISO-certificeringer, sender man løsark rundt til folk og laver sikringsmetoder for at tjekke om folk nu rent faktisk også opdaterer deres ringbind med de nye ark, eller bare diskret smider dem i papirkurven. Indenfor IT er det egentligt paradoksalt at de muligheder IT giver os for at administrere dokumentation kun i sjældne tilfælde udnyttes. De fleste driftmanualer står stadig som ringbind i mørke arkiver, og de der ligger tilgængeligt elektronisk, ligger på mappedrev, som kun visse personer har adgang til. Skal man finde frem til information, som ligger udenfor ens normale arbejdsområde, bliver opgaven nærmest uløselig, uden at kende de rette personer. Som projektsekretær har jeg selv oplevet at skulle finde information om en voice-server, som man i projektet kun kendte navnet på. Der gik 3 uger uden held, indtil jeg en dag brokkede mig over det mens jeg spiste frokost, hvorefter en mand ved nabobordet pludseligt bekendte sig som administrator for den server. Den slags er naturligvis ikke acceptabelt, men er desværre virkeligheden i mange virksomheder. Det er for stort et arbejde at holde de mange worddokumenter opdaterede, printede og indsat i ringbind, når man i forvejen er presset med arbejdsopgaver.

Nu kunne man så forvente at ovenstående afsnit afdækker det største problemområde indenfor dokumentation, men tværtimod er teknisk dokumentation faktisk det område der er bedst dokumenteret, generelt set. Når det kommer til personale, organisation, strategiarbejde og forretningsprocesser, er der endnu længere mellem fornuftige løsninger. Har virksomheden et eller flere dokumenter der beskriver dens forretningsprocesser, kan man næsten være sikker på at de ikke længere er gyldige. En aktivitet har ændret sig, nogle ansvarsområder skiftet, eller en hel organisationsrokade er gennemført, og ingen fik siden opdateret de forretningsprocesser man betalte konsulenter gode penge for at dokumentere 1½ år tidligere. En god fingerregel er at forretningsprocesdokumentation som regel er forældet når man har fået printet den og sat den på hylden.

Ser man på virksomheders dokumentation af deres strategiske arbejde ser det tilsvarende slemt ud. Ledelsen kan føre sig frem med at fortælle om virksomhedens strategi for fremtiden, men oftest er strategien blot en powerpoint præsentation der

indeholder en række strategiske beslutninger. Findes der et strategidokument, så er det, ligesom forretningsprocesdokumentationen, højst sandsynligt forældet. Strategien kan være blevet ændret som følge af andre strategier (eksempelvis kan en teknologistrategi påvise at elementer af en produktionsstrategi ikke kan udføres) eller omgivelserne kan have ændret sig så markant at man ikke længere kan følge en strategi betingelsesløst. I DR oplever vi eksempelvis sidstnævnte i forbindelse med politiske forhandlinger omkring medieforlig og public service aftaler.

De ovenstående afsnit beskriver de problemer de fleste kender til i deres virksomheder, når dokumentation er baseret på dokumenter. Undertegnede kan næppe titulere sig selv som den første til at gennemskue omfanget af problemer, og der har da også været gjort mange forsøg på at effektivisere udarbejdelsen og vedligeholdelsen af, samt genfindingsmulighederne i virksomhedens dokumentation. I DR forsøgte man sig med et dokumenthåndteringssystem (ESDH).

> Case: Da man påbegyndte det store teknologiprojekt til DR-Byen, indså man hurtigt at man ville blive begravet i dokumenter, hvis ikke man opsatte et system fra begyndelsen af. Man valgte et ESDH-system[1], og opsatte strenge regler for indekseringen af dokumenter. Efter det nu har eksisteret i fem år ligger der over 18000 dokumenter i det, og selv de mest erfarne brugere, tør ikke med sikkerhed sige om de kan finde et bestemt dokument igen.

Det indbyggede problem i et dokumenthåndteringssystem er at man stadig arbejder med dokumenter, og, som man har gjort helt fra de gamle grækeres biblioteker, putter dem i de kasser hvor man tror der er størst chance for at folk vil lede efter dem. I ESDH-systemet fandtes der hundredevis af kategorier et dokument kunne indekseres under, for et dokument skulle indekseres under en, og kun en, mappe. Man havde så mulighed for at indsætte relationer til andre typer dokumenter, men eftersom at disse typer var forsøgt bestemt da teknologiprojektet blev igangsat, passede det hurtigt dårligt overens med virkeligheden. Man havde ikke længere tid til at opdatere, og derfor endte det, som langt de fleste lignende indsatser, som spildt arbejde.
Grundlæggende er man simpelthen nødt til at droppe denne årtusinde gamle ide om at man skal putte alting i kasser og mapper, og i stedet begynde at fokusere på hvilke

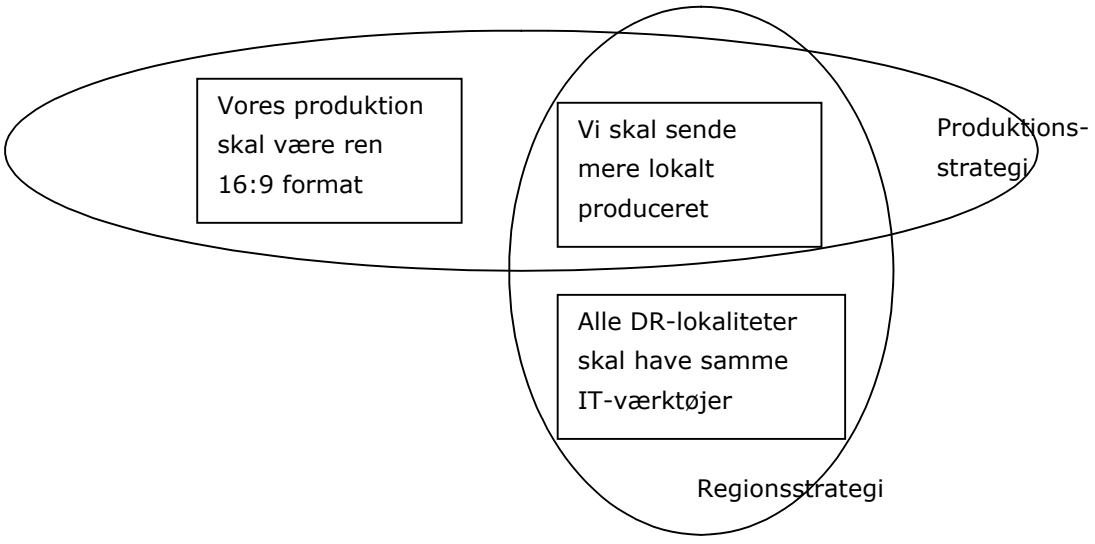
[1] Elektronisk sags- og dokumenthåndtering

relationer de har, frem for hvor de ligger henne. Men at skabe disse relationer giver ikke den fulde værdi, når den viden der er indeholdt stadig bare er tekstdokumenter.

Problemet ligger simpelthen i at de er dokumentbaserede. I årtusinder har man været slave af det format man har kunnet anvende; Hamurabis lov skrevet på stentavler, Biblen nedfældet i papyrusruller, Don Quixote foreviget i bogform og de sidste mange år har man måttet tænke rent i A4, A3 og A5 papirformater, for det var hvad printerne kunne udskrive. Selvom der er langt fra en mellemøstlig stentavle til en rapport udskrevet på laserprinter, så deler de en meget begrænsende egenskab. Det der står skrevet i dem er unikt for dem. En sætning på en stentavle kan kopieres, men sætningen selv, kan ikke bruges andetsteds.

Det forandrer dynamisk dokumentation fuldstændig ved at gøre al dokumentation objektorienteret. Det at begynde at tænke sin forretning objektorienteret, er faktisk ganske revolutionerende. Man er pludseligt ikke begrænset af om man kan finde den rapport en detalje står i, men kan derimod frit fremfinde objektet om detaljen og desuden se hvad den detalje relaterer sig til. Havde man kigget i en rapport, kunne man kun vide hvad der relateres til op til det tidspunkt rapporten er udskrevet. Er omstændighederne ændret (navnet på en ansvarlig), noget nyt kommet til (nu understøttes endnu et interface) så er det vigtigt for forretningen at man ved det med det samme.

Figuren herunder viser et eksempel der udspringer i to objektorienterede strateger:



-Figur 5.3 En strategisk beslutning er ofte relevant i flere strategiske sammenhænge. Eksemplet her viser 3 strategiske beslutninger, og hvordan den ene hører til to forskellige strategier. De fleste beslutninger vil gå på tværs af klassiske

strategidokumenter. De to strategiske beslutninger på figuren, der umiddelbart kun hører til en strategi, kunne jo desuden begge være en del af DRs teknologistrategi.

Hvor indsatsen i en dokumentbaseret virkelighed altid har ligget i udarbejdelse af dokumentet, ligger der nu en mere konstant opgave. Mange har nok oplevet tendensen til at folk tænker "nu er dokumentationen printet og gemt på fællesdrevet, så er den opgave overstået", men denne holdning må fundamentalt ændres. Det er langt hurtigere at sætte objekter sammen til en historie man kan viderebringe, men til gengæld må man også siden hen opdatere de relationer der gør det muligt. Derfor er tanken om dynamisk dokumentation, eller objektorienteret forretning, ikke en pludselig omkostningsfri frelser for virksomheden, men et krævende alternativ der på sigt kan give forretningen værdi.

I den store forandring det vil være for hele organisationen at indføre dynamisk dokumentation, er selve dokumentationen den tydeligste ændring, da det er selve udformningen der fundamentalt ændres. Fem underpunkter er identificeret der vil blive brugt til at vurdere den objektorienterede tilgang i forhold til den dokumentbaserede;

- Indtastning
- Genbrug
- Vedligehold
- Genfinding
- Præsentation

Indtastning

Indtastningen er meget forskellig, da man med en objektorienteret tilgang ligger energi i at oprette objekter og relatere dem til hinanden, frem for at udarbejde dokumenter. Det kræver mere teknisk og forretningsmæssig indsigt at arbejde objektorienteret, da man skal kunne relatere sit objekt til andre objekter på samme niveau eller i samme søjle i ens rammeværk. At arbejde dokumentbaseret betyder oftest at man blot skal kende det problemområde man sidder med. Dermed stiller den objektorienterede tilgang tydeligt større krav til medarbejderne.

Der bliver i dag sat større og større krav til systemudviklere. Dagene hvor man kunne sidde på sit kontor og ikke have med kunder at gøre, er forbi. I stigende grad forventes det at udviklerne sidder sammen med slutbrugerne, for bedst muligt at forstå deres behov. De agile systemudviklingsmetoder, og specielt eXtreme Programming, er på den måde slået igennem i de fleste virksomheder, inklusiv DR. Denne nye måde at arbejde på ligger i sig selv op til at udviklere og forretningsbrugere kommer tættere på hinanden, og får en større indsigt i hinandens

arbejde. Der er ikke kun tale om at udvikleren får indsigt i forretningsgange, men også at brugerne får forståelse for udviklingen af understøttende teknologi. Denne fælles forståelse er et godt udgangspunkt for implementeringen af dynamisk dokumentation. Den bør føre til at begge ser værdien af et dokumenteret forhold mellem de to verdener, og ved at have styr på sin del af dokumentation, hjælper man kendte personer i en anden afdeling, i stedet for blot at opfylde et dekret oppefra. På denne måde må man identificere de trends der alligevel er oppe i tiden, og anvende dem i denne store forandringsproces, som vi kommer mere ind på i analyseemne 4.

Genbrug

Ægte genbrug gør sig kun gældende for objektorienteret dokumentation. Det er nemlig muligt at genbruge eksisterende objekter, når man skal beskrive en ny situation. Objekterne kan man enten relatere gennem databasen, eller indsætte dem på nye diagrammer.

Man kan naturligvis også genbruge information gemt i dokumenter, men som jeg var inde på tidligere så har du pludselig en afhugget kopi, der kan ligge og sprede forældet information. Genbrug er også vejen til en god standardiseret anvendelse af dynamisk dokumentation. Når brugeren ved at han skal kigge i en bestemt liste for at finde objektet for et system han skal bruge, vil man kunne sikre at systemerne er kaldt ved deres rigtige navne. I traditionel dokumentation skriver dokumentøren det han mener systemet hedder, og det passer ofte ikke overens med andres opfattelser deraf. Hedder Microsofts tekstbehandlingsapplikation ”Word”, ”Office Word” eller ”Microsoft Office Word”? Eller mere slående; skal man skrive ”Oracle”, ”Oracle Applications”, ”Oracle 10g” eller hvordan indikerer man at noget forsvinder ind i dette store sorte hul af et virksomhedssystem?

Vedligehold

I forlængelse af ovenstående bliver vedligeholdelse langt mindre krævende. Ændrer man metadata om et objekt, vil det slå igennem i alle de kontekst objektet optræder i. I modsætning er vedligeholdelse en stor udfordring i en dokumentbaseret dokumentation, og den primære grund til at dokumentbaseret dokumentation oftest optræder forældet.

Sikrer ens styringssløjfe at dokumentation reelt vedligeholdes, kan den dynamiske dokumentation virkelig blive effektivt. I produktionsvirksomheder vil man kunne indikere for samtlige aktiviteter hvor længe de tager, hvilket senere kan udregnes til prognoser og estimeringen. Men aktiviteter ændrer sig ofte, eller er afhængige af hvile parametre de blev igangsat af, og det er meget svært at kommunikere og

vedligeholde i en dokumentbaseret dokumentation. I stedet kan man med dynamisk dokumentation ændre detaljerne om estimeringen af aktiviteteten, og den ændring vil slå igennem i samtlige udregninger der indeholder netop denne aktivitet. Dette eksempel kunne naturligvis ligeså godt omhandle metadata som;

- Integrationer for et System.
- Tilkoblede netværk for en server.
- Kritiske succesfaktorer for en strategi.
- Hvilke personer der opfylder hvilke roller etc.

Kort sagt kan alt hvad vi vil indtaste om vores objekter, vedligeholdes på denne måde, for senere at blive anvendt. Der vil være forskellige primære værdier at hente i denne forbindelse for forskellige typer af virksomheder. DR, der er en meget kreativ og anarkistisk arbejdsplads, vil næppe kunne gennemføre en aktivitetsestimering af alle processer, mens det netop kunne give stor værdi for en fabrik. I DR kan vi til gengæld få chancen for at holde kontinuerligt styr på vores utallige integrationer mellem systemer og services, hvilket vi ser som en kæmpe gevinst.

Genfinding

Eftersom der altid vil være et, og kun et, objekt der beskriver et arkitekturelement, kan man være sikker på at finder man objektet, så er det den senest opdaterede udgave man sidder med. Man kan også være sikker på ikke at skulle lede mapper, drev og dokumenter igennem for at finde det, men vil derimod blot skulle slå op i en database eller på en portal.

En dokumentbaseret dokumentation kan ikke sikre en at man finder det rigtige, selvom man lige har brugt lang tid på at rode mapperne igennem. Det kan være at der findes nyere oplysninger om arkitekturelementet i et andet dokument.

Præsentation

Objekter kan sættes sammen i forskellige sammenhænge, og præsenteres på mange måder. Vil man have en rapport, kan man udskrive de relevante objekter og derved får man hurtigt de nyeste informationer leveret på den klassiske måde (herefter overtager man så også problemerne, for ligger den printede rapport i længere tid, er den jo også forældet). Alternativt kan man udvikle en portal hvor brugere kan søge oplysninger direkte i ens database. Gennem en sådan portal kan man følge vejene gennem objekternes relationer og på den måde finde frem til ens mål via mange veje. En vil måske starte med at se virksomhedens værdikæde og følge relationer og nedbrydninger indtil han finder ud af hvilken applikation der understøtter et bestemt workflow. En anden ville starte med at liste hvilke

applikationer der gennem systemfunktioner kunne opfylde et sådant workflow, og på den måde nå frem til samme mål.

I en dokumentbaseret dokumentation kan man udgive sin viden på samme måde, men man vil altid være bundet af datas validitet, og vedligeholdelsen af denne er vanskelig og omkostningsfuld.

> Case: I DR har vi lagt planer for en arkitekturportal, som vi håber, kan blive en central indgang for folk der har brug for viden omkring sammenhænge i DR. Arkitekturportalen vil være en dynamisk udgivelse af de databaser der indeholder vores dynamiske dokumentation, således at man hele tiden har de nyeste informationer med. Men det er naturligvis ikke nok bare at udgive det, man er også nødt til at designe yderligere funktionalitet ind. Egentligt kan folk fremsøge alle elementer (så længe de er relateret, og det burde de alle være), da man blot skal tage fat i et af de overordnede diagrammer eller modeller (dem der ligger på det konceptuelle niveau) og så arbejde sig ned i detaljen, indtil man finder det man søgte efter. Det er dog også vigtigt at lade folk sortere deres information på andre, og mere klassiske, måder. Der skal være mulighed for at fremkalde lister over arkitekturelementtyper (en systemliste eksempelvis), liste arkitekturelementer ud fra metadata (en liste over samtlige beslutninger relaterende til en bestemt forretningsproces, sorteret ud fra deres oprindelsesdato) eller udskrive en dynamisk rapport som man ved indeholder den relevante information. Rapporter vil være en integreret del af portalen, da de, på samme måde som diagrammer, kan kommunikere en række arkitekturelementer i en sammenhæng der fortæller en bestemt historie. Ved at rode rundt i portalen og finde elementer frem, må man selv forstå den historie man er i, for elementerne er kun byggesten dertil. Rapporterne skal til gengæld designes så de kan fortælle brugerne hele historier, som eksempelvis;
>
> - Alle Use Case- og workflowdiagrammer for Licensafdelingen, med udregning af aktivitetsestimering.
> - Alle beslutninger omhandlende et givent system, sorteret efter beslutningstager.
> - Et givent informationsflow startende med et forretningsobjekt, og dets rejse og transformation gennem systemer, integrationer og webservices.

Den dynamiske dokumentation giver mange fordele overfor en traditionel dokumentbaseret dokumentation, men om det samlet set er indsatsen værd, må bero på det enkelte EA-team og deres virksomhed, for som beskrevet i introduktionen til analysen, så er det et område der er meget svært at sætte økonomisk ræsonnement op for. I dag ville det kræve en visionær ledelse at satse på den dynamiske dokumentationm men om nogle år tror jeg på at ideen om dynamisk dokumentation vil vinde frem. Der skal tvingende nødvendighed til før folk vil gå i gang med en så stor forandringsproces, der umiddelbart er så svær at kvantificere. Men jeg er sikker på at når virksomhederne generelt bliver serviceorienterede, vil folk få behov for denne type overblik. Verden går stadigt hurtigere, og man skal handle hurtigere og hurtigere for at få konkurrencefordele på markedet, hvorfor jeg tror, at virksomheder vil se værdien af en tæt forbundet, altid opdateret og godt kommunikeret arkitektur.

5.4 Analyseemne 4 Organisationen

Vi har indtil nu kigget på hvad man skal have udviklet og standardiseret for at kunne implementere en dynamisk dokumentationsmodel, men ikke berørt de konsekvenser ændringerne vil have for medarbejdere og hvad der må kræves af ledelsen af virksomheden for at kunne tro på en succesfuld implementering. Jeg har identificeret tre fokusområder for medarbejderne og ligeledes tre for ledelsen. Medarbejderne vil blive nødt til at udvikle nye vaner, deres arbejdsgange ændres i varierende grad og de må vænne sig til endnu flere IT-værktøjer i deres dagligdag. Af ledelsen er man til gengæld nødt til at kigge hvad den må gøre, og det er i dette tilfælde, som i de fleste andre forandringsprojekter; ledelsesfokus, forankring og involvering.

Medarbejdere

At implementere en objektorienteret dokumentationsmodel vil have store konsekvenser for de berørt medarbejdere, alt efter deres ekspertise. Det vil for IT-folk betyde at stort set alle skal dokumentere på denne nye måde, mens det i forretningen vil betyde at nogen skal udnævnes (og allokeres) til at være udviklere og vedligeholdere af forretningsdokumentation. For at belyse disse forhold har jeg valgt at kigge på følgende, og fokusere på IT-folket da det er her den største udfordring ligger;

- Nye vaner
- Ændrede arbejdsgange
- Nye værktøjer

Nye vaner

For at kunne opnå dynamisk dokumentation er man nødt til at ændre medarbejdernes indstilling til omkring dokumentation, hvilket nok kan vise sig som den samlet set største opgave i projektet. Traditionelt anses dokumentation som et nødvendigt onde for udviklerne, når de selv skal udføre den. I anvendelsen af dokumentationen er medarbejdere præget af udpræget mistillid til skrevne dokumenter, hvorfor megen videndeling foregår mellem medarbejdere i stedet for gennem et dokumentationssystem. For forretningen er dokumentation oftest ikke-eksisterende, og medarbejdere har ingen erfaring med udvikling af dokumentation og oftest kun ringe erfaring som brugere (dette dog lidt anderledes hvor man arbejder meget med certificeringer).

Så der ligger tre store udfordringer for at den samlede medarbejderstab kan blive klar til dynamisk dokumentation. Først og fremmest må folk indse at udarbejdelsen og vedligeholdelsen af dokumentation som et centralt og vigtigt led i alle arbejdsprocesser. For det andet er det en udfordring at få folk til at stole på den information de finder, så de ikke længere bruger megen tid på at finde lige præcis den kollega der ved noget om et emne. For det tredje må man introducere dokumentation for de områder af forretningen der ikke traditionelt har arbejdet med det tidligere.

Ændrede arbejdsgange

Indførelsen af objektorienteret dokumentation vil betyde at mange arbejdsgange må ændres for de fleste IT-medarbejdere samt for den del af forretningsmedarbejdere der skal arbejde med dokumentation. Arbejdet kan ses som to forskellige situationer; udvikling af dokumentation eller vedligehold af dokumentation. For udviklere vil ændringerne være mindre end for driftpersonalet. Udviklerne vil stadig kunne udvikle deres løsninger uden at dokumentere dem undervejs (selvom man bør), så længe løsningerne har et tilfredsstillende dokumentationsniveau den dag de sættes i drift. For driftpersonalet vil den nye måde at dokumentere på række helt ind i de mest basale arbejdsprocesser; ingen applikation kan skifte virtuel server uden af det opdateres i dokumentationen og anvendes en webservice af en ny applikation må det ikke udelades. Alt efter hvilket detaljeringsniveau man vil ned på med en objektorienteret dokumentation kan det blive endnu mere; skal brugeroprettelser, hardwareopgradering og fysisk kabling være en del af den objektorienterede dokumentation?

Nye værktøjer

Den mindste af de tre udfordringer for medarbejderne er at de naturligvis må lære et eller flere nye værktøjer til inddatering, processering og præsentation af dokumentation. For nogen vil værktøjerne være det der giver inspiration til at komme i gang med at dokumentere, for andre kan det virke uoverskueligt at skulle lære noget nyt. I sammenkædning med de to afsnit ovenfor må uddannelsen i nye værktøjer dog ses som en mindre udfordring. Har man først fået folk til at indse at de må ændre deres vaner og arbejdsgange, skal de nok også se værdien og nødvendigheden af et understøttende værktøj. Udfordringen i at vælge det værktøj er til gengæld en hel anden, for at nå den højest mulige accept og tilslutning blandt medarbejderne. Det er umuligt at finde et værktøj der er det bedste på markedet indenfor alle typer dokumentation; fra workflows over strategier til UML-diagrammer. En del af forandringsprocessen er at få folk til at anerkende at et generelt værktøj der

lever op til 80% af brugernes ønsker indenfor de enkelt fagområder er langt mere givtigt for hele virksomheden, end en lang række specialiserede dokumentationsværktøjer hvor man ikke kan relatere objekterne imellem. Det kan dog vise sig at man må gå på kompromis visse steder, og så søge at kunne integrere et par værktøjer gennem services eller fælles databaser. Det er svært at overbevise udviklere om at de skal bruge et Use Case værktøj der ikke understøtter automatisk kodegenerering, når det netop vil give dem en stor gevinst.

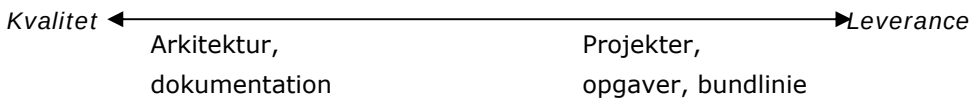
Ledelse

Ledelsen er det primære sted man skal søge at forankre de forandringer der må foretages for at opnå en dynamisk dokumentation. Uden tilstrækkelig opbakning fra ledelsen er ethvert initiativ i denne retning formålsløst. I dette afsnit vil vi kigge på tre faktorer der gør sig gældende for ledelsen;

- Ledelsesfokus
- Forankring
- Involvering

Ledelsesfokus

Som beskrevet i afsnittene om dokumentation og medarbejdere, er det en meget omfattende forandring at indføre objektorienteret dokumentationsstyring. Det er i denne forbindelse essentielt at ledelsen, og hele ledelsen, anerkender vigtigheden af dynamisk dokumentation, de mål der er sat derfor og følgelig holder fokus overfor deres medarbejdere. En af udfordringerne for at have bedre dokumentation er at det umiddelbart strider mod mange medarbejderes mål, nemlig at blive hurtigt færdig med opgaver eller projekter. Ledelsen må gå foran med at kræve kvalitet i dokumentationen, og sikre at der afsættes tid til dette. I en virksomhed der er presset på tiden, vil det blive umådeligt svært at overbevise stressede medarbejdere om at de lige skal overarbejde et par timer mere for at dokumentere.



Figur 5.4. De modsatrettede interesser. Arkitektur skal sikre kvalitet i virksomheden, mens projekter afkræves deadlines og hurtig afslutning. Begge dele burde være lige vigtigt for forretningen, men det er nemmere at måle succes på

leverancer, hvorfor de kvalitetsrettede funktioner får sværere ved at vise deres berettigelse.

Forankring

Samtidig med at ledelsen, og samtlige ledere i virksomheden, fokuserer på dokumentation, skal hele indsatsen forankres i virksomhedens fundamentale organisation og kultur. Dokumentation skal, eksempelvis sammen med arkitektur, profileres i en vision, et mål eller i det mindste i en kritisk succesfaktor for virksomheden. Dette vil sikre at fokus kan opretholdes over tid, og det vil være bragt helt op på bestyrelsesniveau. I organisationen skal der tydeligt markeres hierarkiske eller virtuelle forhold angående dokumentation; eksempelvis en markering af et dokumentationsråd med tilhørende personer eller roller (mere herom i næste kapitel). Dette vil legitimere at medarbejderen bruger tid på sin dokumentation, og at den nærmeste leder, der ofte vil være leveranceorienteret, anerkender dette arbejde.

At forankre dokumentationsarbejde i virksomhedens kultur er det endelige og vanskeligt opnåelige mål. Ingen chef kan diktere den kultur der findes i virksomheden, og den eneste måde at påvirke den på er at vedligeholde stabile mål og visioner, og på den måde lade forandringer stille og roligt antage kulturel værdi for medarbejderen. Kan man nå til et punkt hvor det at dokumentere er en ufravigelig del af arbejdet som medarbejderne ikke blot anerkender, men glædeligt udfører, og de sætter deres lid til kvaliteten af andres dokumentation, vil man have opnået en virksomhedskultur der kan sikre dokumentationsindsatsen mange år fremover.

Involvering

Ledere er som alle andre forskellige, og mange bryder sig ikke om voldsomme forandringer. Derfor kan et skift til en objektorienteret dokumentation være et ligeså usikkert og tvivlende skridt for en leder, som for en medarbejder, for dokumentationsindsatsen er ikke kun et anliggende for medarbejdere. For at have en dynamisk dokumentation, er det essentielt at også ledernes arbejde dokumenteres; vedtagne beslutninger skal dokumenteres og eventuelt konsolideres som politik eller standard og egne beslutninger indenfor lederens område må på samme måde dokumenteres, for at medarbejdernes arbejde kan relateres hertil. Derfor er lederne også nødt til at lære sig værktøjerne og komme i gang med arbejdet. Forskellen mellem leder og medarbejder er naturligvis at lederen må gå foran og bakke op om ledelsens planer for arkitektur og dokumentation, mens medarbejderen kan tillade sig at være lidt mere tilbageholdende.

En kollega argumenterede, halvt i spøg, i en diskussion om netop dette, at hele arbejdet ville falde til jorden hvis jeg virkeligt satte min lid til at lederne selv skulle lave noget. Jeg vil på ingen måde definere hvem der aktivt skal inddatere lederens dokumentation, men man opnår ikke en fyldestgørende dynamisk dokumentation uden ledelseslagenes arbejde.

Dette analyseemne rejser kort en række aspekter af at implementere dynamisk dokumentation, set for medarbejdere og ledelsens sider. Dette emne kunne have været et speciale for sig selv, hvis fokus havde været på forandringsledelse. Jeg vil i det næste kapitel komme med anbefalinger til hvordan man kan lette transitionen gennem forandringsprocessen, og hvordan arbejdet kan organiseres for at sikre både dokumentationens kvalitet, men også medarbejdere og ledelsens interesse og involvering.

6. Anbefalinger

Dette kapitel skal tjene det formål at opsummere de vigtigste erfaringer, synspunkter og råd fra analysen, og skabe en vejledende guide for hvordan man kunne gå frem i forsøget på at implementere en dynamisk dokumentationsmodel. Inden man dog når så langt er det naturligvis vigtigt at grundigt overveje om man virkelig vil forsøge sig med en så krævende disciplin, som dette speciale i hvert fald angiver at det kan være. Som jeg skrev i analysen mener jeg det er formålsløst at forsøge at vurdere værdien af en dynamisk dokumentation ud fra kroner og ører. På samme måde overvejede jeg at anvende en udvidet cost/benefit analyse for at vurdere værdien i forhold til strategisk match. Denne er dog meget afhængig af den kontekst man bevæger sig indenfor, og med mit datagrundlag fra et projekt, vil jeg ikke stille mig frem med en skabelon for hvilke faktorer der betyder hvad, i forhold til en succesfuld implementering. Jeg vil nøjes med at angive, at det må være et valg der gøres af EA-teamet i samarbejde med virksomhedens ledelse. Det vigtige i denne sammenhæng er at der gøres et valg og emnet dermed bliver belyst. Vælger man ikke at forsøge sig med dynamisk dokumentation, ved man dog så hvad det betyder. Kommer der så senere folk med ideer der ligger indenfor dette felt, ved man også at man bør genoverveje beslutningen, eller i det mindste overveje hvordan nye dokumentationsteorier kunne passe ind i helheden.

Det er min mening at de fleste virksomheder ikke er klar til en sådan ændring, men det kommer meget an på virksomhedens syn på teknologi. Er virksomheden en klassisk *frontrunner*, kunne dette være næste måde at skabe en konkurrencefordel på. Tror man mere på Nicholas G. Carr´s efterhånden klassiske artikel ”IT doesn´t matter”, der argumenterer for teknologi som hyldevarer, og at indtjeningen ved at ligge som teknologisk *frontrunner* er væk, ser man måske heller ikke dynamisk dokumentation som det vigtigste tiltag ens forretning kan gøre på nuværende tidspunkt. Det er det måske heller ikke, men det kunne det nemt være om blot få år, i takt med at vores virksomheder bliver serviceorienterede.

Jeg har valgt at inddele mine anbefalinger i en trinvis guide, med udgangspunkt i analyseemnerne;

1. Vælge, tilrette eller lave et rammeværk
2. Udvikle en dokumentationsmodel
3. Udvikle en modelleringsstandard
4. Forankre arbejdet i virksomheden

Før man dog kan gå i gang er man nødt til at vurdere om ens virksomhed i det hele taget er moden til at gå i gang med den dynamiske dokumentationsmodel. Som afsnittet vil afsløre, er der nemlig forskel på ens EA-modenhed og hvor klar man er til dynamisk dokumentation. Efter det vil de fire trin blive gennemgået. De første tre vil samle erfaringerne fra analysen op og præsentere dem på en hurtigere overskuelig måde. Det fjerde punkt vil vi i dette kapitel gå lidt i dybden med. Egentligt var det ikke mit mål at beskæftige med planlægnings- eller governance-aspekter af EA, men for at komme hele vejen rundt om dynamisk dokumentation, fandt jeg det nødvendigt at komme med nogle overvejelser omkring hvordan arbejdet med dokumentationen bør struktureres og styres.
Titlen på punkt fire kunne desuden lede tankerne henimod forandringsledelse som helt afgjort vil være en af de store udfordringer ved at implementere en så omfattende forandring for organisationen. Som tidligere nævnt, kunne dette dog uden problemer fylde et helt speciale i sig selv, og er en helt anden udfordring, der desuden ville være meget generel og sammenlignelig med al anden forandring, hvorfor dette aspekt ikke vil blive gennemgået i dette speciale[1]. Generel litteratur om forandringsledelse vil kunne lede videre her, og har virksomheden specialister indenfor dette felt (som flere og flere virksomheder faktisk anskaffer) så er det oplagt at denne side af implementeringen skulle foregå ud fra deres principper.

6.1 Modenhed

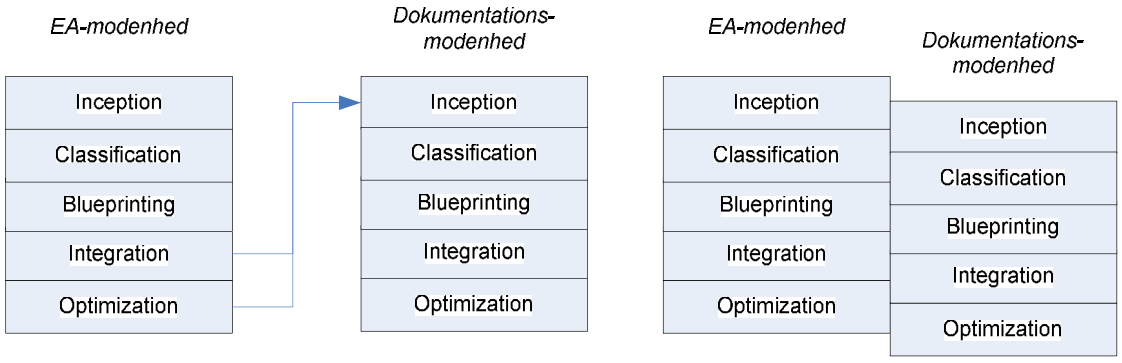
Som tidligere beskrevet er det nødvendigt på sigt at ændre virksomhedskulturen til at hver enkelt ser dokumentation, og udarbejdelsen heraf, som noget selvfølgeligt og nødvendigt. At begynde implementeringen af en så stor forandring på alle fronter, teknologisk, processuel og kulturel, på et for tidligt tidspunkt vil nemt kunne amputere projektet. Jeg ser det som en klar nødvendighed at virksomheden allerede har nået et vist niveau af forståelse for og accept af EA, før man kan indføre en dynamisk dokumentationsmodel.

I relation til Herzums EA-modenhedsmodel (Herzum, 2003) vil jeg nedenfor vurdere hvornår i hans trinvise model man kan begynde at indføre en dynamisk dokumentationsmodel, med størst mulighed for succes. Indførelsen af en sådan model følger i det store hele de samme fem trin som EA. Erkendelsen af at man må indføre en ny måde at dokumentere, og dermed styre sin forretning, leder til et stort klassifikationsarbejde, altså niveau 2. Samtlige arkitekturelementer må identificeres, beskrives og en fælles semantik omkring de mange elementer skal opnås. Vigtigt i

[1] Som beskrevet i afgrænsningen

denne proces er det tilmed at definere relationstyper mellem arkitekturelementerne. Efter man således har defineret den måde man vil dokumentere sin verden på, er det på tide at gå i gang med at dokumentere. I første omgang gælder det at dokumentere "skelettet" for hvad den efterfølgende dokumentation skal kunne relatere sig til, og det betyder at dokumentere det øverste, eller mest abstrakte, niveau i ens rammeværk (her tages udgangspunkt i DRs rammeværk, andres kan jo se anderledes ud) samt ens strategisøjle. Dette skelet udgør værdikæde, organisationsdiagram, informationsarkitektur, domænemodel, strategier, politikker og måske beslutninger. Efter denne opgave er man klar til at lade sine brugere gå i gang med at dokumentere hver deres områder af rammeværket, og dermed er man på niveau 4 der omhandler integration. Jo mere brugerne udfylder og relaterer, jo tættere er man på et, nok utopisk, totalbillede af ens virksomhed fra øverste værdikæde og måske helt ned til serverracks i kælderen. Efter en tid, vil folk forhåbentligt tage dokumentationen til sig og den vil blive en integreret del af det daglige arbejde, og dermed har man nået niveau 5, hvor en driftsituation indtræder. Der er *kun* tilbage at drive og optimere dokumentationsmodellen.

Dette ræsonnement kunne lede til at man sidestillede EA-modenheden med hvad vi herefter kan kalde Dokumentationsmodenheden, og lader de to forandringsprojekter køre samtidigt. Det tror jeg dog ikke er realistisk. Forståelsen af hvad en dynamisk dokumentationsmodel kan give virksomheden i værdi, vil være et resultat af et intenst EA-arbejde, hvorfor dokumentationsmodenheden altid vil følge efter EA-modenheden. At opnå niveau 4 for sit EA-arbejde kan stadig betyde at arkitekterne sidder og udarbejder ad hoc dokumentation hver gang noget nyt skal besluttes, uden et fast dokumentationsgrundlag, og det vil nok først være på modenhedstrin 4 eller 5 at man vil beslutte sig for at give sig i kast med en så kompliceret dokumentationsmodel.



Figur 6.1 - Forholdet mellem EA- og dokumentationsmodenhed.

Ovenstående viser to mulige sammenhænge mellem de to typer modenhed. Til venstre ser vi den ovenfor beskrevne, model 1, hvor erfaringerne med EA leder til at tænke i bedre dokumentation. Indsatsen begynder først ved EA-modenhedsniveau

Integration eller Optimization. Til højre ses model 2 der er en tidsmæssigt hurtigere implementering af både EA og dokumentationsmodel. Her følger hver dokumentationstrin umiddelbart efter det tilsvarende EA-modenhedstrin. Det er min klare opfattelse af virksomheder ikke normalt vil være så afklarede og fokuserede på dokumentation så tidligt i EA-processen at dette normalt vil kunne lade sig gøre. Indfører man EA med ekspertbistand, hvor ledelsen fra begyndelsen er klar over potentialet i dynamisk dokumentation som følge af eksperternes vejledning, kan modellen dog blive relevant.

For at medarbejderne trykt og entusiastisk skal gå ind i denne nye måde at arbejde med dokumentation på, er det vigtigt at ledelsen, og dermed organisationen, er modnet så der kan meldes klart ud. Skal medarbejderen lige pludseligt ligge en større umiddelbar indsats i dokumenteringsarbejdet, skal han eller hun være tryg ved at gøre det[1]. Har virksomheden opnået et af de to højeste EA-modenhedsniveauer bør der ingen intern tvivl være om værdien af EA. Denne accept skal bruges til at sikre ledelsesfokus og -forankring, samt garantere en tryg forandringsproces for alle involverede. Dette er udgangspunktet for overhovedet at igangsætte et forandringsprojekt med forhåbninger om succes.

6.2 Punkt 1: Vælge, tilrette eller udvikle et rammeværk

Der er som titlen angiver tre måder at skaffe sin virksomhed et rammeværk på. Vælg et eksisterende, tilret et eksisterende eller selv at udvikle et fra bunden. Analysen kunne vise at det kan være svært at vælge et, da de eksisterende rammeværk i stor stil også indeholder forfatterens holdning til EA. Men at udvikle et rammeværk helt fra bunden vil være omsonst, da der er gjort mange gode tanker i de eksisterende rammeværk, så derfor er den klare anbefaling at tilrette et eksisterende. Det kan resultere i;

- Flere eller færre søjler
- Flere eller færre niveauer
- Andre navne på celler
- Andre afgrænsninger mellem søjler, niveauer og celler.

Om man er til carbones idé om at tilrette rammeværket henimod ens egen kontekst, eller som Bernard, vil omfavne hele ens virksomhed, så vil rammeværket tydeligt børe

[1] Det er naturligvis dette speciales tese at den tid er givet godt ud, for mere tid ville blive anvendt til at undersøge sin kontekst og forsøge at skaffe sig en viden derom, uden en dynamisk dokumentation. Umiddelbart vil det dog for medarbejderen se ud som om han skal bruge længere tid, i stedet for kortere.

præg deraf. Jeg har et sted set en søjle der hed Projekt, da organisationen var meget projektorganiseret, mens andre holder sig helt til de, efterhånden, klassiske seks søjler. Men der er ingen rigtige eller forkerte valg. Den bedste måde at finde frem til det mest optimale rammeværk for ens virksomhed, vil være at arbejde ud fra den trinvise model i figur 6.1, og så samtidig med rammeværket udvikle sin dokumentationsmodel. De to skal hænge sammen på alle måder, hvorfor en sidestillet udvikling er at foretrække.
For at kunne komme ordentligt i gang med ens dokumentationsmodel er det vigtigt at man afgrænser søjler, niveauer og celler. For senere at kunne ligge et arkitekturelement i en bestemt celle er det vigtigt at kende afgrænsningen for de involverede søjler, niveauer og cellen. Det er her arbejdet med Zachmans model nemt bliver uoverskueligt.
Det er også i rammeværket at man ligger kimen til hvor detaljeret ens dynamiske dokumentation kan blive. En simpel model, ala Carbones Business Framework, ligger ikke op til en særlig detaljeret dokumentation, mens Bernards eller DRs har en passende størrelse, der både sikrer overskuelighed og detaljering.

6.3 Punkt 2: Udvikle en dokumentationsmodel

Virksomhedens dokumentationsmodel er et relationsdiagram over alle vigtige ting i virksomheden, eller rettere sagt, de ting der er vigtige for sammenhængene, nemlig arkitekturelementerne. I DR startede vi med et stort kanvas der indeholdte en mat udgave af rammeværket bagerst, og rigeligt plads til at lave relationer mellem flere elementer indenfor hver eneste celle. På denne måde kunne vi igennem hele processen overskue placeringen af elementerne og hvilke sammenhænge de havde til andre relationer.
Det anbefales at begynde med en bruttoliste over de ting man finder vigtige for forretningen, og så bryde dem ned til det detaljeringsniveau man vil have. Derefter placeres de i dokumentationsmodellen, og til sidst indsætter man relationerne mellem dem. Dette vil unægtelig blive en iterativ proces, og som anbefalet i punkt 1, bør det gøres i forlængelse af rammeværksudviklingen.

6.4 Punkt 3: Udvikle en dokumentationsstandard

Dokumentationsstandarden for virksomheden vil i mange tilfælde komme til at fremstå som en blanding af egne interne standarder, og eksterne markedsstandarder. Ingen markedsstandard er omfavnende nok til at dække hele dokumentationsmodellen, og det er samtidigt tilrådeligt at anvende

markedsstandarder, for at kunne kommunikere bedre med ens partnere og leverandører.
I denne del af opgaven ligger der et stort arbejde med at analysere hvilke markedsstandarder man bør satse på. I analysen nåede jeg frem til følgende trinvise model:

1. Har standarden vundet indpas på markedet endnu?
2. Hvilke eksisterende udviklings-, projekt og forretningsudviklingsmodeller anbefaler denne modelleringsstandard?
3. Hvem står bag standarden, og er den certificeret?
4. Har *the big guns* valgt en standard de satser på?
5. Hvordan vil den spille sammen med den eksisterende modelleringsstandard(er) i virksomheden?
6. Hvor langt vi vi gå for at følge de valgte standarder?
7. Stol aldrig på en konsulentvirksomhed.

Konkrete anbefalinger vil jeg ikke komme med, men umiddelbart er det min overbevisning at åbne standarder i længden vil sejre over proprietære, og det synes de store virksomheder også at have indset.

Udfordringen er at skabe et forhold mellem ens dokumentationsmodel og -standard, hvor der er et 1:1 forhold mellem arkitekturelementer og symboler. Dette er dog sjældent muligt, men hav det med i overvejelserne.

6.5 Punkt 4: Forankre arbejdet i organisationen

En gennemtænkt og forankret organisering af arbejdet vil være essentiel for en succesfuld forandringsproces og et uundværligt trin på vej mod at gøre virksomhedsdokumentation til en del af virksomhedskulturen. Termen organisering dækker over mere end blot organisationen bag arbejdet, men først vil organisationen dog blive gennemgået. De vigtigste roller i dokumentationsarbejdet, uden at tænke ledelsen ind, er dokumentationsarkitekten, chefarkitekten, forretningsbrugerne og IT-superbrugerne. Efter en præsentation af disse roller vil styringsformen bag selve dokumentationsindsatsen blive gennemgået.

Dokumentationsarkitekt: Det er denne forfatters holdning at der i en EA-afdeling bør være en arkitekt der beskæftiger sig med dokumentation som hovedfokus. Er dette ikke muligt, eller ønsker man at prioritere anderledes, er det en mulighed at tilknytte en egnet medarbejder fra en anden afdeling. Dette kunne logisk være en metodespecialist der får et virtuelt tilhørsforhold til arkitektteamet. Det må i det mindste anbefales at personen med dokumentationsansvaret referer direkte til chefarkitekten som vist på figur 6.2.

Carbone definerer at man skal have en ansvarlig for modelleringsmetode, men fokuserer udelukkende på datamodellering. Dermed udelades resten af virksomheden, hvilket synes mangelfuldt, men Carbone er naturligvis fokuseret på EAP og er "data-centreret"[1].

Dokumentationsarkitekten er ansvarlig for rammer og metoder i forbindelse med rammeværk og alt det dokumentation der tastes ind i rammeværkets sammenhænge. Personen skal stå som ansvarlig for kvaliteten og mængden af dokumentation, og på den måde bliver dokumentationsarkitekten et vigtigt bindeled mellem forretning og IT, og samtidig rådgiver for de andre arkitekter i EA-teamet.

Chefarkitekt: Lederen af EA-teamet står for det overordnede ansvar for sammenhængen mellem forretning og IT-understøttelsen deraf. Chefarkitektens rolle er ofte at kommunikere og forankre sine arkitekters arbejde for ledelse og topledelse. Alt efter om virksomheden har valgt modellen med EA underlagt IT-direktøren ("CIO") eller virksomhedens øverste direktør ("CEO"), vil chefarkitekten have sin daglige gang blandt lederne. Den ledende arkitekt vil have brug for et solidt datagrundlag og kvalificerede rådgivninger, for at kunne få EA-teamets arbejde forankret, og derfor vil en dynamisk dokumentation være et centralt værktøj for ham, og dermed dokumentationsarkitekten.

IT-superbrugere: I IT-organisationen vil der generelt være mange der aktivt anvender dokumentationsværktøjerne, men ikke alle disse kan være med i det udviklende arbejde. Derfor må der udnævnes superbrugere for dokumentation. De vil være folk med interesse i modelleringsmetoder, dokumentation og typer der brænder for at få styr på virksomheden. Typen af superbrugere vil her primært foregå ud fra dokumentations- og modelleringsteknikker og metoder. Man kunne have en superbruger for UML-diagrammer og en anden for infrastrukturdokumentation. Disse superbrugere ville så have alle de brugere under sig, der beskæftiger sig med deres områder og metoder, på tværs af organisatoriske tilhørsforhold.

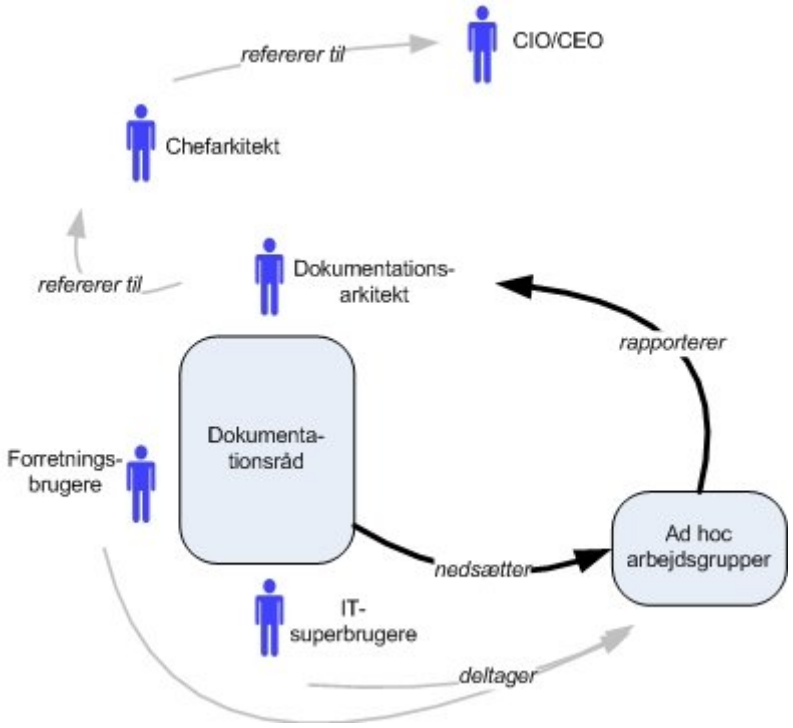
Superbrugerne bør have samme profil som forretningsbrugerne, og samme type egenskaber med dokumentationsværktøjerne, om end de altså bør være inddelt udfra funktion, frem for område. Dermed kan disse superbrugere også blive sekretærer for de ad hoc arbejdsgrupper (se nedenfor) der nedsættes indenfor deres ekspertiseområde.

Forretningsbrugere: I forretningen må man forvente at der er færre dokumentører end i IT-organisationen. Generelt set bør man i forretningsdelen af virksomheden have en dokumentør for hvert "område". Dette område må så defineres

[1] "…our philosophy is somewhat data-centric…" (Carbone, s.165)

som en række forretningsprocesser der hænger sammen ud fra deres indbyrdes relationer, organisatoriske tilhørsforhold eller dokumentørens ekspertise. Jeg har ingen teori til at bakke en af de tre områdedefinitioner op, men vil mene at det bedste vil være at dokumentørerne udvælges for en række logisk relaterede forretningsprocesser. Eksempelvis kunne alle processer omhandlende lagerstyring ligges under en dokumentør, mens en anden tog sig af processer omkring produktionen eller en del af denne, i en typisk produktionsvirksomhed. Alternativt kunne man nedsætte en dokumentør udfra organisatoriske tilhørsforhold, hvilket måske kunne give en bedre lokal forankring, når dokumentørens leder kan ser at dennes arbejde understøtter lederens ansvarsområde. I mange virksomheder sidder der desuden folk der har en stor ekspertise på specielle områder, og det kunne være en mulighed at udnævne dokumentører udfra ekspertiseområder. En hybrid udgave af ovenstående vil nok være det mest realistiske, og det vil bestemt være bedre at gå efter at få ildsjæle ind som dokumentører, frem for at udnævne ud fra en stram standard.

Forretningsbrugerne vil have gode evner og med tiden få stor erfaring med dokumentationsværktøjerne, og bør gennem dokumentationsrådet inddrages i generelt udviklende arbejde. Det skal desuden være deres ansvar at videregive slutbrugernes (dem der i forretningen "læser" dokumentationen) meninger, ris, ros og ideer til dokumentationsarkitekten.



Figur 6.2 Organisering af dokumentationsarbejdet.

Figuren ovenfor viser hvordan dokumentationsindsatsen kan styres, så man kan sikre optimale betingelser for dokumentørerne, og deraf følgende høj kvalitet. I centrum for denne lille historie er dokumentationsforummet, hvor der dog ikke udføres noget arbejde. Forummet indeholder samtlige forretningsbrugere og IT-superbrugere. Forummet er dokumentationsarkitektens kommunikationskanal, og vil primært fungere som fødekanal for at nedsætte ad hoc arbejdsgrupper. Eventuelle meget generelle diskussioner, fælles uddannelse eller workshops kan betyde at hele forummet kan mødes, men det er urealistisk at forestille sig en fast rutine af møder. Er dokumentationsindsatsen gennemført vil forummet jo indeholde ganske mange mennesker.

Det arbejde der søges gjort i denne styringssløjfe er;

- konsolidering af relationer der ligger mellem ansvarsområder
- oplæg til tilretning af værktøjer, skabeloner og metoder
- Vurderinger og afklaringer af ændringer eller eventuelle nyindkøb

Netop for ikke at sidde og debattere i en større forsamling er det dokumentationsarkitektens ansvar at nedsætte arbejdsgrupper med udspring i dokumentationsforummet, til at udføre disse opgaver. Arkitekten skal have mulighed for at allokere medlemmerne af forummet i en rimelig grad, og arbejdet i grupperne skal have høj prioritet for de udvalgte.

Den første type opgaver omhandler udelukkende kontekst i dokumentationen, hvor man finder det nødvendigt at sikre relationer i dokumentation der ligger mellem ansvarsområder. For at fortsætte samme eksempel som ovenfor omhandlende forretningsbrugere i en produktionsvirksomhed, så kunne et eksempel her være at man har dokumenteret både produktions- og lagerprocesser. Mellem disse vil der dog være mange indbyrdes relationer. Ender et workflow med at en vare er klar til at blive lagerført, er det vigtigt at det relateres til et workflow der beskriver selve lagerføringen. Satte man ikke de to forretningsbrugere sammen, kunne man risikere inkonsistens eller mangler i dokumentationen. En sådan arbejdsgruppe kan være meget uformel, og ansvar må ligges hos de enkelte forretningsbrugere, da mindre tilretninger af dokumentation kan have betydning for relationerne, uden dokumentationsarkitekten har nogen chance for at opdage det. Arkitekten vil derimod kunne allokere tid for forretningsbrugerne til samarbejde hvis et større projekt forandrer deres områder markant.

Der vil løbende være opgaver med at videreudvikle og tilrette ens dokumentationsværktøjer, såvel som de skabeloner og metoder der arbejdes ud fra. Dette arbejde bør udføres af de involverede selv, og blot reviewes af dokumentationsarkitekten for at sikre at den samlede pakke hænger sammen.

Det kan også være væsentligt at få afklaret eller vurderet en foreslået ændring, eller om hvorvidt man bør indkøbe nyt. Dette arbejde kan ligge i en ad hoc arbejdsgruppe, eventuelt med dokumentationsarkitekten som deltager.

For ad hoc arbejdsgrupperne vil deres aflevering altid gå tilbage til dokumentationsarkitekten. For den første type opgave kan det have form af en gensidig erklæring fra brugerne om at "nu er vi enige om vores indbyrdes relationer", mens det for de to andre typer opgaver vil være arbejde der kræver dokumentationsarkitektens review og vurdering. Det er min klare holdning at jo mere arbejde man på denne måde kan uddelegere, jo bedre resultat får man. Deltagerne bliver engagerede i udviklingen og bliver måske ildsjæle og bannerførere for kulturforandringen, og det er dem der sidder med det daglige arbejde, der er med til at udvikle systemer, metoder og skabeloner. Dette arbejde kan dog ikke foregå usystematisk, hvorfor dokumentationsarkitekten bliver nødt til at være den endelige godkender for de ændringer, nyindkøb og lignende, som brugerne anbefaler. Skal der overvejes noget nyt, er det desuden dokumentationsarkitekten der skal sætte rammerne for brugernes arbejde, som jeg her vil give et eksempel på.

I dag findes der adskillige forskellige måder at tegne workflows på. UML-standarden kan præsentere *activity* diagrammet, indenfor BPEL hedder det lignende *Business Process Diagram* og endelig er der det gamle ikke officielt standardiserede workflowdiagram, som har været med os siden "business reengineering"-ideen vandt frem. Vi har i DR valgt at anvende det typiske workflowdiagram, i en dialekt som vores værktøj anvender der ligger meget tæt op af den uofficielle standard for workflows. Vi er dog samtidig meget interesserede i at ligge os indenfor officielle standarder, hvorfor vi burde anvende UML Activity-diagrammet. Samtidig ser vi en stor fremtid for BPEL-modellering, hvorfor dette diagram måske burde være vores standard. Denne udfordring kunne være en opgave for et par forretningsbrugere sat sammen med IT-organisationens superbruger indenfor UML-diagrammer. Det er her vigtigt at formulere opgaven til at vurdere netop de tre diagramtyper og deres fordele og ulemper i forhold til hinanden. Stillede man opgaven mere løst, som "vurder forskellige workflow-lignende diagrammer, og find ud af hvad vi skal anvende", ville opgaven blive meget større og mere uoverskuelig for deltagerne, og resultatet garanteret mere speget. Det er altså dokumentationsarkitektens ansvar at have den generelle viden til at kunne opstille spørgsmål i en ordentlig opgaveform. I dette

tilfælde kender arkitekten de konkurrerende modelleringsmetoder, men vil ikke vælge uden et ordentligt internt datagrundlag, hvorfor en ad hoc arbejdsgruppe nedsættes. Deres resultat kan dokumentationsarkitekten så tage med i en samlet overvejelse der desuden omhandler strategisk og økonomisk udsigt ved at skifte metode, fremtidsperspektiver for dem og meget mere.

For den menige indtaster af dokumentation betyder ovenstående at man kontinuerligt vil kunne stole på at man har de bedste værktøjer til rådighed, og at metoder og skabeloner er opdateret og tilrettet. Er der noget man finder uhensigtsmæssigt, eller har man fået en ide til noget nyt eller ændret, kan man få sit forslag frem til dokumentationsarkitekten som så kan vurdere og eventuelt igangsætte en arbejdsgruppe.

Ovenstående arbejdsområder gælder naturligvis også præsentation af dokumentationen som rapporter og portal, hvorfor administrationen af EA-teamets kommunikationskanaler også bliver en sag for dokumentationsrådet, og man i rådet sikrer at både indtastning og præsentation af dokumentation foregår på standardiserede og fornuftige måder.

Konklusion

Det har vist sig at ingen af de anvendte teoretikere kan præsentere rammeværk eller dokumentationsmodeller, der tilnærmelsesvist kan understøtte det en dynamisk dokumentation ville kræve. Jeg forudser at der vil ske skridt i retning af det jeg nu kalder den dynamiske dokumentationsmodel, selvom den givetvis vil blive kendt under andet navn, eller måske som forskellige grene indenfor EA.

Det blev også klart at ideen om *in-a-box* løsninger ikke synes givtig på dette felt. Det synes umuligt at lave modeller alle kan anvende, uden at de bliver for generelle. Desuden er det også et vigtigt skridt for virksomheden selv at tilrette sit rammeværk, da det skaber forståelse for de interessenter der deltager. Det skal dog ikke lyde som et argument imod at forsøge at lave generelle modeller, for som inspiration er de uundværlige. Jeg ville selv aldrig have nået dertil hvor jeg nåede uden at have kendte Carbones og Zachmans rammeværker, selvom jeg i denne kontekst kritiserer dem.

Jeg mener at have anskueliggjort værdien af en dokumentationsstandard for at fremme letlæselighed og udvikling af diagrammer. Det kunne være et emne i sig selv, men i denne kontekst bliver det klart hvor essentiel en dokumentationsstandard også er for den dynamiske dokumentationsmodel. Skal man kunne tillægge relationer på diagrammer nogen værdi, er man nødt til at have standardiseret deres anvendelse.

Den komparative analyse af den dynamiske dokumentationsmodel overfor klassisk dokumentbaseret dokumentation anskueliggjorde også store gevinster og høj værdi for virksomheden, men det sidste analyseemne viste dog også hvor stor en udfordring implementeringen kan blive. Det er en meget stor forandring der skal finde sted kulturelt, såvel som teknologisk.

For at hjælpe den interesserede til at vurdere om emnet er relevant har jeg opsat en fire punkts guide, samt en opfordring til grundigt at vurdere virksomhedens EA-modenhed, hvis man ønsker at implementere den dynamiske dokumentationsmodel.

Det har været en lang og givtig læringsproces at skrive dette speciale, og en god oplevelse at forsøge at arbejde ”praktisk akademisk”. Den konstante refleksion over mit praktiske arbejde, har givet mig balast både som arkitekt og generel akademiker.

Litteratur og bilag:

Litteraturliste

Bernard, Scott. 2004. "An Introduction to Enterprise Architecture". Authorhouse.

Business Process Management Initiative. Business Process Modeling Notation (BPMN). Link: www.bpmn.org

Carbone, Jane. 2004. "IT Architecture Toolkit". Prentice Hall.

Carr, Nicholas G. 2003. "IT doens´t matter". Harvard Business Review. Vol 81, nr 5.

Herzum, Peter. 2003. "Applying Enterprise Architecture". Cutter Consortium Executive Report, vol 6, nr 3.

Morgenthal, JP. 2005. "Enterprise Information Integration: A Pragmatic Approach". Lulu Press.

Schekkerman, J. "Structuring the Enterprise around services - The difference between Hype, Hope and Reality?." Link: http://www.enterprise-architecture.info/EA_Services-Oriented-Enterprise.htm

Schön, Donald A. 1983. "The Reflective Practitioner: How Professionals Think in Action". Basic Books.

Object Management Group. Unified Modelling Language - UML resource Page. Link: www.uml.org

United States General Accounting Office (USGAO). 2003. " A framework for assessing and improving enterprise architecture management". Link: www.gao.gov/new.items/d03584g.pdf

Wagter, Roel, Martin van den Berg, Joost Luipers, Marlies van Steenbergen. 2005. "Dynamic Enterprise Architecture: How to make it work". John Wiley and Sons.

Windley, Philip J. 2006. Governing SOA. link: http://ww6.infoworld.com/products/print_friendly.jsp?link=/article/06/01/19/73698_04FEsoagov_1.html

Zachman, John. 1986. "The Original Article - A framework for Information Systems Architecture". www.zifa.com/ArchitectureArticles (kræver gratis login)

Bilag 1: Anvendte teknikker og værktøjer i casen

Baseret på Bødker et al´s MUST-metode (Kapitel 9 ”Teknikker og Beskrivelsesværktøjer”).

Teknikker og beskrivelsesværktøjer:

- Review
    - Iterativ review-udvikling (hvorfor er det så vigtigt= fordi rammeværket skal ramme 95% rigtigt første gang, og for at lette forankring i brugere)
- Høring (lidt forankring)
    - Før noget kommunikeres bredt ud, eller til ledelse, kommer det for en høring i VAF. Det forestillede vi os ville sikre de værste bommerter fra at kommer igennem (problemer med VAF!)
- Dokumentanalyse
    - Kortlægning af hvad der allerede fandtes i DR af strategier der kunne underbygge vores projekt.
    - Manglen på dokumenter eller manglende genfinding var faktisk central!
- Observation (say-do problematik) (eks server/service, ledte til workshop)
    - Gjorde en dyd af at observere hvad folk mente når de brugte forskellige ord. (server, applikation, arkitekt etc.) For at disse senere kunne udfordres.
- Workshop
    - Frihåndstegning
        - Mest almindelige værktøj ved møder i DR. Whiteboards overalt!
        - Gav osse anledning til at ønske sig en standardisering af tegnemetoder!
    - Dødehavsrulle / workflows og BPN
        - Hvordan ville vi arbejde proces-orienteret med arkitekturdeltagelse i projekter ex.
    - Rolleliste
        - Internt i BIT EA: Hvem kan hvad, hvem gør hvad i en ny og udefineret afdeling?
        - Eksternt BIT EA: Interessentanalyse og disses forventninger til os.
    - Kommunikationsmodel
        - Plan for indledende kommunikationsplan indenfor BIT

        - Skabte overblik over den kontekst vores centrale værktøj skulle eksistere i. (Kun på demo plan desværre)
- Fantasifase i fremtidsværksted
    - For at skabe fælles fodslag og bedre kunne sigte vores visioner, udførte vi en kalenderbaseret tegning af hvad vi fire arkitekter foretog os i løbet af en uge i 2007 (aka drift-situation)
- Virksomhedsbesøg (QLM-forbindelse, ITST, Gardner Symposium)
    - ”Det kan jo ikke passe at vi skal genopfinde den dybe tallerken, der må da være noget ”best-practice” eller nogle andre erfaringer at trække på” Det argument hørte vi hele tiden. Til dels sandt, men vi er langt fremme i skoene.
    - Deltog i brugerkonferencer gennem vores værktøjsleverandør
    - Forsøgte at igangsætte samarbejde med ITST (for forskellige mål desværre)
    - Brugte energi på at orientere sig ift ex Gardner / Metagroup.
- Prototype
    - Udvikling af portaldesign
    - Udvikling af rapportdesign