

Ordrupvej 101
2820 Gentofte

tel. +45 44 50 21 50
fax. +45 39 64 56 37
www.lector.dk

Service Orienteret Arkitektur

- en kommerciel betragtning

d. 22. november 2006

Indhold

Indledning	4
Forventet publikum	4
Forkortelser	4
Service Orienteret Arkitektur (SOA)	5
Baggrund	5
Hvad er SOA?	5
Hvad bringer SOA til en virksomhed?	6
Skal alt nu til at være SOA?	8
Sikkerhed	8
Lector & SOA	9

Indledning

I de senere år er begrebet "Service Orienteret Arkitektur" (SOA) begyndt at optræde mere og mere i IT-branchen og i den generelle debat. SOA er et godt "brugt" begreb som desværre ikke altid fremstår lige godt defineret ensige beskrevet.

I dette dokument præsenteres Lector's opfattelse af begrebet SOA; i denne omgang set ud fra en kommerciel betragtning. En teknisk betragtning¹ på SOA eksisterer sideløbende med nærværende dokument.

Forventet publikum

Det forventede publikum til dette dokument er ikke-teknisk personale. Folk med beslutningsansvar i forhold til IT-arkitektur og systemstruktur generelt kan med fordel anvende dette dokument. Et generelt kendskab til IT er dog en fordel, da en del af termer som er anvendt er IT-specifikke.

Forkortelser

Forkortelse	Navn	Beskrivelse
SO	Service Orienteret	
SOA	Service Orienteret Arkitektur	
WS*	WebService *	Samlet betegnelse for en række webservicestandarder
CORBA	Common Object Request Broker Architecture	Standard for udveksling af informationer systemer imellem.

Referencer:

1. "SOA – en teknisk betragtning", Lector whitepaper 2006

¹ "SOA – en teknisk betragtning", Lector 2006

Service Orienteret Arkitektur (SOA)

Baggrund

Virksomheders IT-systemer er til for at understøtte virksomhedernes forretning; aldrig omvendt. IT-systemer er til for at automatisere arbejdsgangene i virksomhederne og derved reducere omkostningerne via reducerede arbejdstider/procestider. Det er ihvertfald hensigten og argumentationen, når de bliver indkøbt til virksomheden. Hvorvidt det faktisk altid lykkedes, et åbent spørgsmål som virksomhederne ofte selv har en holdning til.

Faktum er dog, at de forskellige afdelinger i en virksomhed meget ofte har deres egne behov at afdække ligesom de samtidigt taler deres eget "domænespecifikke" fagsprog (eksempelvis *finansafdelingen* kontra *produktionsafdelingen*); begge forhold som kun bidrager til at disse diskrete afdelinger indkøber egne dedikerede applikationer og systemer som er skræddersyede til at løse netop deres opgaver i virksomheden. Herved opnåes et IT-mæssigt systemlandskab som tilfredstiller de respektive afdelinger fuldstændigt, men ofte opnåes dette på bekostning af øget drifts- og vedligeholdelseskostning i virksomhederne, totalt betragtet. Det samlede IT-systemlandskab bliver ofte til en komposition af "silo'er" eller "øer" i virksomheden med hver deres egne data og egne brugere. Det produkt som alle afdelinger forsøger at bidrage til at producere og som i sidste ende driver forretningen; det være sig fysisk såvel som intellektuelt, vil typisk være det samme produkt. Dette forhold skaber et naturligt ønske om at kunne "genbruge" data de forskellige systemer imellem for derved at reducere behovet for at vedligeholde f.eks. produktstamdata flere steder.

Disse ønsker kunne i nogle situationer være afhjulpet ved at "standardisere" på et større Enterprise Resource Planning System (ERP) som f.eks. BAAN, SAP eller Microsoft Navision. Ikke nok med at dette ofte vil medføre en radikal og stor beslutning for virksomheder; det vil også bringe virksomheden i en situation hvor den vil være "bundet" af dette system og dets evne til at tilfredsstille virksomhedens faktiske forretning. Netop her kommer et væsentligt element ind; *IT-systemerne må ikke være drivende for en forretning, det omvendte skal være tilfældet* – det er forretningen som skal være den drivende. Det er forretningen som genererer overskud til virksomheden, ikke IT-systemet isoleret betraget. Med andre ord skal forretningen diktere hvorledes IT-systemet skal opføre sig eller udpege hvilke processer IT-systemet skal hjælpe/støtte forretningen med for at bistå de opgaver der løses i forretningen.

Det er yderst sjældent, at et enkelt system kan understøtte alle afdelinger komplet, hvorfor ovennævnte løsning med et ERP system vil være meget sjældent at se; derimod vil det oftest være tilfældet at flere diskrete systemer vil understøtte specialistfunktioner i virksomheden supplerende et gennemgåede ERP-system. Uanset hvordan – opstår behovet for at kunne udveksle data de respektive systemer imellem.

Hvad er SOA?

Konceptet med at bringe afkoblede system istand til at tale sammen er som sådant ikke nyt, da det har været kendt længe under andre navne. Blandt andet har CORBA² været anvendt i en lang årrække som "metoden" til at bringe systemer istand til at kommunikere. Men - dog først efter at XML i slutningen af 90'erne blev generelt accepteret som værende "sproget" som kunne bringe forskellige IT-systemer istand til at tale sammen begyndte flere og flere systemer og softwareleverandører at tilbyde anvendelse af XML som standard for distribueret kommunikation.

Med muligheden for at kunne tilbyde sine data via en standardiseret grænseflade og protokol til omverdenen åbner også muligheden for at kunne tilbyde en "service" sig; en service som i teorien kan være hvad som helst.

² "Common Object Request Broker Architecture", OMG group

Et interessant spørgsmål på dette tidspunkt ville være: *Hvad er så en service?* Der eksisterer mere end een definition af dette, men den opfattelse Lector anvender af begrebet er følgende:

*En **service** er et selvstændigt stykke funktionalitet som kan anvendes autonomt uden tvungne afhængigheder af andre eksterne systemer/forhold.*

Services ses oftest præsenteret som XML-webservices, da dette som nævnt er den standard de fleste leverandører understøtter idag. Dette er dog ikke noget krav, men en teknologisk implementering og ligeledes den implementering/teknologi som gav hele SOA-begrebet momentum ved udgangen af 90'erne.

Da vi nu har begrebet Service defineret vil det være relevant at tage skridet videre og definere begrebet *Service Orienteret Arkitektur (SOA)*? Igen her eksisterer der en stor mængde definitioner af dette; men Lector opfatter SOA som følger:

*At lave en **Service Orienteret Arkitektur (SOA)** er arbejdet med at sammensætte diskrete services til et koherent systemlandskab der understøtter de forretningsprocesser virksomheden har identificeret som værende "drivere" af forretningen.*

og

*En **Service Orienteret Arkitektur (SOA)** er betegnelsen for et etableret systemlandskab som gør anvendelse af diskrete services for at tilbyde et sammenhængende og koherent IT-system til at bistå de forretningsafledte opgaver.*

Et nøglepunkt i ovenstående definitioner af SOA er at forretningen skal være den styrende enhed og aflede IT-systemet; ikke omvendt! *Dette er en absolut forudsætning* for at kunne høste de fordele SOA lover.

Note: En ofte set og meget udbredt misforståelse af SOA er at WebServices = SOA. Dette er ikke korrekt; webservices er blot en teknologi som kan anvendes til at etablere det tekniske grundlag for en SOA Arkitektur.

Hvad bringer SOA til en virksomhed?

Hvad kan SOA faktisk anvendes til i en virksomhed? SOA lover en række fordele – fordele som ikke alle virksomheder/organisationer kan hente fordi de ikke nødvendigvis er parate til det.

SOA lover blandt andet følgende fordele:

- Reducerede vedligeholdelseskostninger for IT-systemerne
- Bedre strømlining af IT-systemerne og forretningen
- Større fleksibilitet i IT-systemerne til at understøtte fremtidige ændringer i forretningen
- Mulighed for anvendelse af eksisterende systemer på andre og nye måder

Som det fremgår, er det en ikke uvæsentlig mængde fordele der loves hvis man anlægger en SOA strategi i sin forretning. Det er dog langt fra givet, at alle disse fordele kan realiseres. Nogle virksomheder vil være istand til kun at realisere udvalgte punkter, mens andre vil være istand til at høste flere fordele; det afhænger altså af virksomhedens kendskab og overblik over egen forretning og egne forretningsprocesser.

En forudsætning for at kunne implementere en SOA arkitektur i en virksomhed, er som nævnt at man kender sin forretning. Dette kan lyde som en kliche, da de fleste naturligvis kender deres egen

forretning; men gør de faktisk også det når det kommer til stykket? Hvis ikke man kan beskrive sin egen forretning og dennes processer; så kan man heller ikke identificere hvilke funktioner som faktisk giver værdi til forretningen og dermed heller ikke identificere de funktioner som eventuelt kan eksponeres som services i en SOA struktur. Med de stadigt reducerede levetider produkter (det værende fysisk som intellektuelt) oplever idag på markedet, oplever også virksomhederne behov for at kunne omstille produktionen til disse forhold. De virksomheder som stadig eksisterer idag har alle formået at foretage denne justering, men dette betyder at sandsynligheden for at virksomhedens faktiske forretning ikke længere er den samme som den var for år tilbage. Ved at tvinge virksomhederne til at gennemgå egne processer og identificere hvilke funktioner i virksomheden som *faktisk* bidrager til positiv værditilvækst kan et opdateret og korrekt *forretningsdesign* opnåes. Dette forretningsdesign er en forudsætning for at etablere et succesfuldt IT-design i virksomheden, da det som nævnt er forretningen og ikke IT-systemet som skal være den drivende part i denne sammenhæng.

Ved at aflede IT-systemets design ud fra forretningens design er det nemmere at introducere ændringer i IT-systemet i takt med at forretningen ændrer sig ved f.eks. reducerede produktlevetider, reducerede udviklingstider (Time To Market) etc. Derved opnåes et tilstedighed tilpasset IT-system som netop understøtter forretningen som den ser ud idag, og ikke hvordan den så ud "igår".

Dette ændrer altså IT-systemet til ikke længere blot at være en nødvendig omkostning for at drive forretning, men tværtimod et fundamentalt værktøj til at gøre forretningen mere konkurrencedygtig, profitabel og istand til at reagere hurtigere på ændrer i markedet.

For at gøre dette begreb lidt mindre abstrakt, vil en række eksempler på Services blive nævnt. Disse services vil naturligvis være brancheafhængige i en vis udstrækning.

Interne Services

Da det nu er muligt at tilbyde sine data til andre systemer, kan det faktisk medføre et reduceret systemlandskab i en virksomhed og dermed afledte reducerede omkostninger i forhold til systemdrift og vedligehold.

Et eksempel er de i indledningen nævnte produktstamdata; et andet eksempel er de mange "identitetssiloer" som ofte ses anvendt i diskrete systemer. Ofte har IT-systemer deres egen database over brugere; en database som så anvendes til at tillade eller afvise brugere når disse forsøger at anvende systemet. Dette kan umiddelbart virke som en begrænset fordel, men hvis man forestiller sig den mængde af systemer som større virksomheder ofte har kørende i deres virksomhed begynder det at have en effekt. Ved at have een "identity provider" i virksomheden (f.eks. Active Directory i Windows) kan denne med fordel sørge for at levere identitetsinformation til andre systemer i virksomheden. Disse har derfor ikke direkte behov for at vedligeholde sin egen opfattelse af brugere (identiteter), men kan nøjes med at trække på een provider/service, nemlig Active Directory.

Interne/Eksterne Services

Et eksempel på et eksternt Service som også kan anvendes internt samtidigt kunne være en service som tillader eksterne sælgere at synkronisere deres lageroversigter med virksomhedens faktiske lagerbeholdning. Denne service (LagerService) kan ligeledes anvendes internt til at levere data til de interne systemer, men kan samtidigt eksponeres til eksterne parter for derved at opnå en genbrugseffekt. Et andet eksempel kan være en ordreservice. Denne service kan anvendes af både sælgere og kunder, dog underlagt restriktioner for hvilke metoder som kan aktiveres på servicen.

Skal alt nu til at være SOA?

Det direkte svar på dette spørgsmål vil være et klart nej! Der er intet som taler for, at alt uden undtagelse skal være services, men de forretningsfundamentale funktioner kan med fordel gøres til services i virksomheden som det er nævnt i foregående afsnit.

Dette bliver nemmere og nemmere med tiden at etablere et SOA landskab, da der vil ses en tendens til at flere og flere leverandører og applikationer *også* vil tilbyde sine funktioner som services til omverdenen. I takt med at teknologierne bliver mere og mere anerkendte og udbredt bliver også den tekniske barriere lavere. De fleste større leverandører af IT-systemer tilbyder teknologier som gør det muligt at etablere SOA strukturer i deres systemer og applikationer.

Intet kommer uden ulemper. Hvis man laver hele sit systemlandskab i virksomheden om til at være serviceorienteret og dermed baserer sig på andre systemers funktioner som i ovenstående eksempel, vil man også introducere en afhængighed som kan vise sig at være fatal. Ved f.eks. ikke at holde sine egne data som en lokal "buffer", vil man opleve at eet system kan trække flere andre systemer med ned ved et lokalt systemnedbrud. Det er på ingen måder en ønskelig konstellation.

Der er flere måder at imødekomme en sådan situation og det vil ofte være tilfældet at udvalgte services er redundante (flere udbydere af samme services). Dette er i særdeleshed relevant for mission-critical services som identitetsproviders, men også andre systemer er i sandhed kritiske i virksomheden; dette kunne f.eks. være finansielle systemer.

Sikkerhed

I relation til services, så er det endog meget væsentligt at de services som udstilles til omverdenen (internt såvel som eksternt) er underlagt sikkerhedsstyring. Hvis ikke en service er istand til at afvise folk ved "porten", kan i teorien alle og enhver trække data fra denne service. Man kunne forestille sig hvor galt det kan gå hvis f.eks. en service som leverer fakturaberegninger eller endog genererer fakturer ikke er underlagt styring af hvem som faktisk kan anvende denne service. Denne styring er ikke kun begrænset til personer, men i ligeså høj grad til hvilke applikationer som kan få lov at anvende servicen. Denne styring vil med fordel kunne baseres på de nævnte identitetsproviders for igen at reducere antallet af "ører" i virksomheden.

For at identificere personer som anvender en service er der efterhånden en række muligheder. Hvis den service man eksponerer skal anvendes af eksterne personer, kan man anvende Digital Signatur til at identificere personerne. Denne digitale signatur kræver en 3. part som siger "god" for en persons identitet udfra et kodeord som kun denne person skulle kende. Denne 3. Part er i Danmark TDC som er såkaldt Certificate Authority for digital signatur. En mere teknisk gennemgang af disse emner kan ses i ref. "SOA – en teknisk betragtning", Lector whitepaper 2006 (side 4).

Lector & SOA

Lector anvender SOA i sine egne systemer. Lector har et dokumenthåndteringssystem (såkaldt ESDH-system) kaldet TeamShare. TeamShare anvender services internt til at eksponere funktioner som for TeamShare er kritiske.

Ligeledes tilbyder TeamShare andre webservices (såkaldte "Business Services") eksternt, således at eksterne parter kan tilgå dokumenter gemt i systemet. Dette er f.eks. tilfældet for en organisations kontorapplikationer (Microsoft Office) som tilgår TeamShare via disse webservices. Dermed betrager f.eks. Word eller Excel TeamShare som en udefineret service. Hvorhenne denne service befinder sig eller hvordan denne service sørger for at fremfinde ønskede dokumenter er for så vidt Office applikationen ligegyldig; blot servicen kan fremskaffe et ønsket dokument for Word at arbejde med.

Lector arbejder med SOA i andre sammenhænge også. Men, det er fundamentalt at anvendelsen af SOA skal give mening i en given kontekst. **Det er ikke SOA for enhver pris!**