



Service orientert arkitektur (SOA)

- Reduserte kostnader
- Mer gjenbruk
- Større fleksibilitet
- Raskere implementering av nye forretningsprosesser

Begrepsavklaringer og et konkret eksempel på bruk av SOA ved publisering av SAP HR Infotypes som Webtjeneste.



Udo Martens

Senior konsulent/ SOA arkitekt
EDB Consulting Group

1 Innledning

I denne artikkelen ønsker jeg å gi en innføring i Service Orientert Arkitektur og hvordan SAP PI integrasjonsplattform kan benyttes som et samordningsverktøy. Jeg forsøker å belyse styrker og svakheter ved SOA gjennom konkrete eksempler. For å få en praktisk tilnærming til SOA begrepet har jeg brukt publisering av SAP HR som webtjenester i ERV (EDB Ressurs- og virksomhetsstyring) som et eksempel.

1.1 Definisjon

Begrepet „Service Orientert Arkitektur“ ble for første gang brukt i 1996 av Gartner Group som derfor kan sies å være den som satte SOA på kartet.

SOA kan på mange måter sies å være et paradigmeskifte for strukturering og bruk av funksjonalitet hos forskjellige applikasjons og prosesseiere. Nedenfor beskrives noen viktige egenskaper og prinsipper ved SOA-arkitekturer.

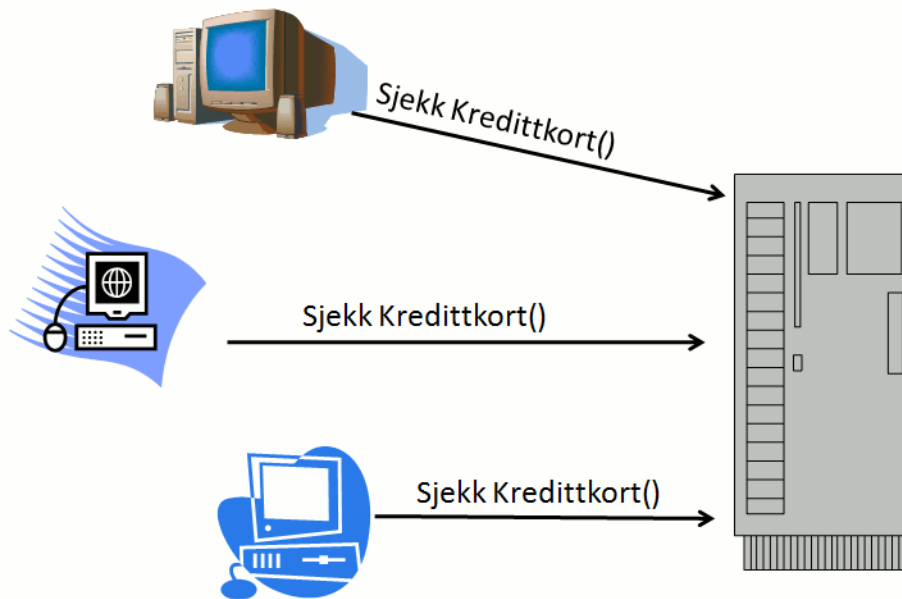
1.2 Målsetting ved bruk av SOA

Det egentlige målet med SOA er å senke bedriftens IT kostnader ved maksimal gjenbruk av standardiserte grensesnitt og SOA tjenester. Derfor må også brukergrensesnittene være standardiserte slik at alle brukere har muligheter til å utnytte tjenestene.

I tillegg må sammenkoplingen mellom avsender og mottaker være løstsittende. Det betyr at applikasjonen tilbyr tjenester og bryr seg ikke om hvem som bruker den og hvorfor. På en måte kan man si at applikasjonene er uavhengige. Tilgang til tjenester beskyttes som regel med bruker-/passord- kombinasjon og eller på andre måter.

Eksempel:

En stor bedrift har mange forskjellige avdelinger med forskjellige applikasjoner. De benytter forskjellige plattformer og teknologier både i og utenfor bedriften. Allikevel kan de fleste tjenestene gjenbrukes, for eksempel kredittkort gjennomgåelse eller masterdata administrasjon.



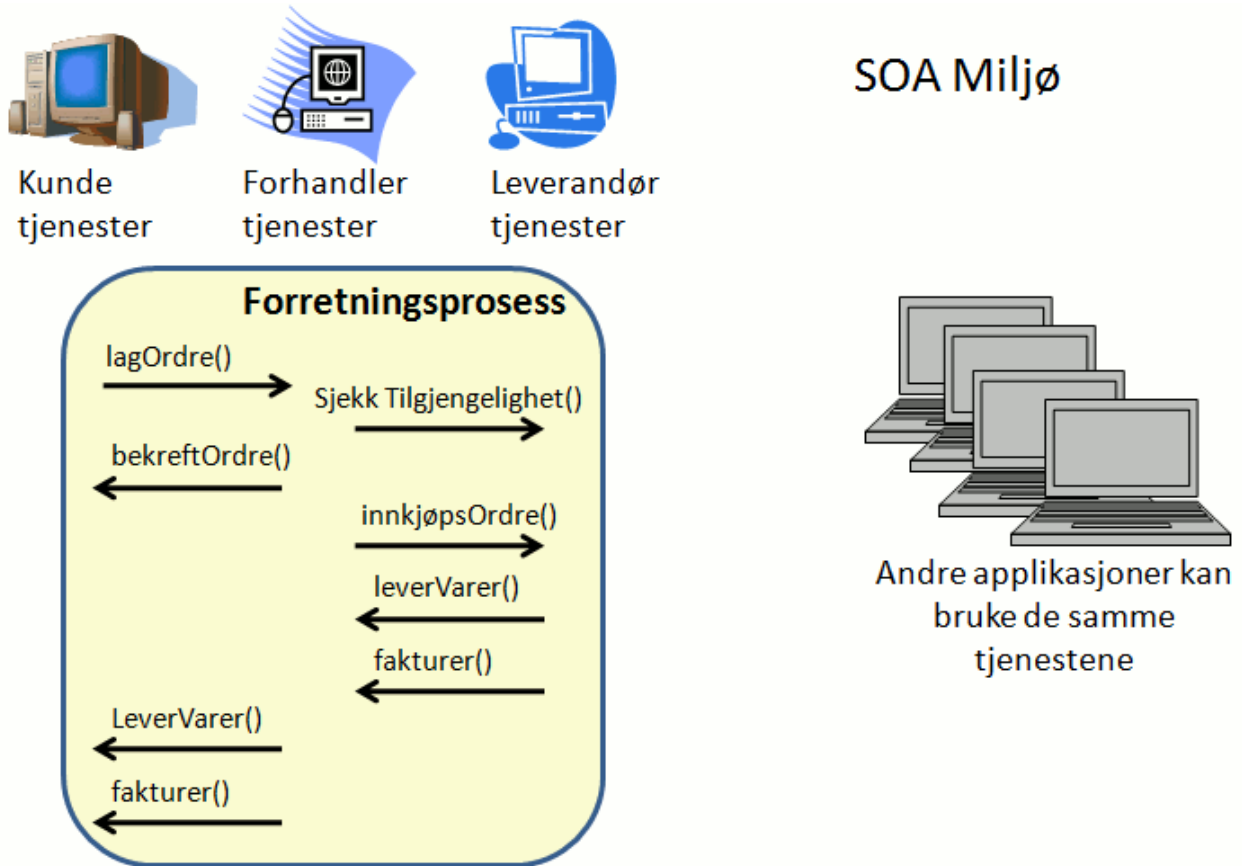
Bilde 1: Gjenbruk av tjenester

1.3 Samordning

Hvis hele bedriftsarkitekturen arbeider med standardiserte grensesnitt og løstsittende sammenkoplinger så blir gjenbruk optimalt. Når man vil opprette en ny forretningsprosess trengs det kanskje ingen utvikling men bare en samordning (engl.: "orchestration") av allerede eksisterende grensesnitt. Ofte kan man også gjenbruke allerede samordnede deler av prosesskjeder. Samordningen kan gjøres ved hjelp av mange forskjellige mellomvare og "Business Process Management" – BPM verktøy fra forskjellige leverandører. I SAP miljø kan man blant annet benytte PI og SAP BPM.

Eksempel:

Forskjellige applikasjoner tilbyr forskjellige tjenester. En forhandler tilbyr en tjeneste "createOrder" og den kan benyttes av forskjellige kunder. Det spiller ingen rolle hvilken kunde som benytter tjenesten. I hver tilfelle blir en prosess startet. Hele kjeden er selvfølgelig i praksis mer komplisert, men fordelene med arkitekturen er mye gjenbruk av tjenester når man lager en ny prosess.



Bilde 2: Forretningsprosesser i et SOA landskap

1.4 Tjeneste-register

I komplekse SOA landskap er det ikke så lett å navigere. Hvis man vil benytte en tjeneste må man først sjekke om den dekker de behov du har. Kanskje har man lyst til å sammenligne pris eller kvalitet eller andre egenskaper hos en tjeneste før man bestemmer seg for å benytte den. På den andre side er en tilbyder av en tjeneste avhengig av at potensielle brukere får kunnskap om tjenesten.

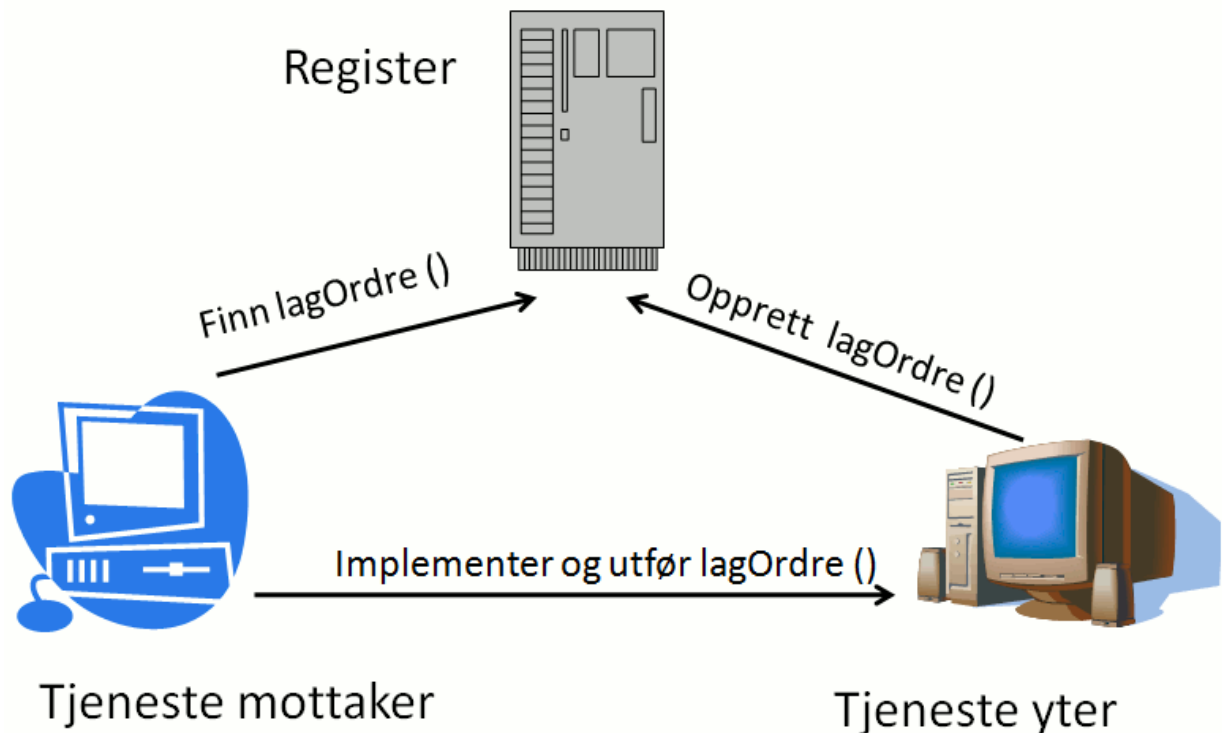
Derfor finnes det ofte et tjeneste-register som virker som en *Gule Sider-katalog*. Man kan publisere sine tjenester der og potensielle brukere kan finne tjenestene med forskjellige søkeparameter. Det er vanlig at man finner tekniske beskrivelser og tekst av tjenestene som gir brukeren de informasjonene som trengs for å ta en beslutning.

Eksempel:

En applikasjon vil tilby tjenesten "opprett ordre". Først blir det laget en tjeneste, normalt en webtjeneste. Teknisk beskrivelse av webtjenesten finnes i WSDL (Web Service Definition Language) -filen blir lastet ned til register-serveren. Det er også en god beskrivelse av tjenesten slik at potensielle brukere (kunder) lett kan finne den.

Brukerne lokaliserer tjenesten og laster ned WSDL og implementerer en applikasjon som bruker den. WSDL er en kjent standard, slik at mange plattformer kan bruke tjenesten uten stor innsats.

Den nye applikasjonen er nå distribuert. Hvis noen bruker den aktuelle applikasjonen så utfører han ikke bare oppgaver på sitt eget system (consumer) men også på et fjernsystem (provider).



Bilde 3: Register i et SOA landskap

1.5 SOA Prinsipper

- Hovedprinsippet til serviceorientert arkitektur er løstsittende sammenkøpling
- Tjenestene er gjenbrukbare
- Tjenestene beror på en formell avtale (Webservices: HTTP, SOAP)
- Tjenestene er uavhengige og kan operere alene.
- Tjenestene er i seg selv lukket.
- Tjenestene kan oppdages i et nettverk: For hver tjeneste finner man en forretningsmessig beskrivelse, en grensesnittbeskrivelse og en tilgangsbeskrivelse. Beskrivelsen følger en standard (Webservices: WSDL)
- For bruk av tjenestene er kun kunnskap om grensesnittet en betingelse, men dette gjelder ikke implementeringen av tjenesten.
- Tjenesten er plattformuavhengig
- Tjenesten er dynamisk bundet, dvs. at det ikke er nødvendig at tjenesten eksisterer mens en anvendelse bygges opp.

1.6 Noen ulemper og forutsetninger for bruk av SOA

For små og begrensede prosjekter kan man ikke bestandig forvente en hurtig "return of investment". Et SOA område er avhengig av implementerte standardtjenester. Bare hvis området er stort nok og graden av implementering høy nok oppnås ønsket gjenbrukgrad.

Et SOA krever investeringer med å løskoble tjenester. Hele it-arkitekturen blir på en teknisk måte mer komplisert og kan derfor tape i ytelse.

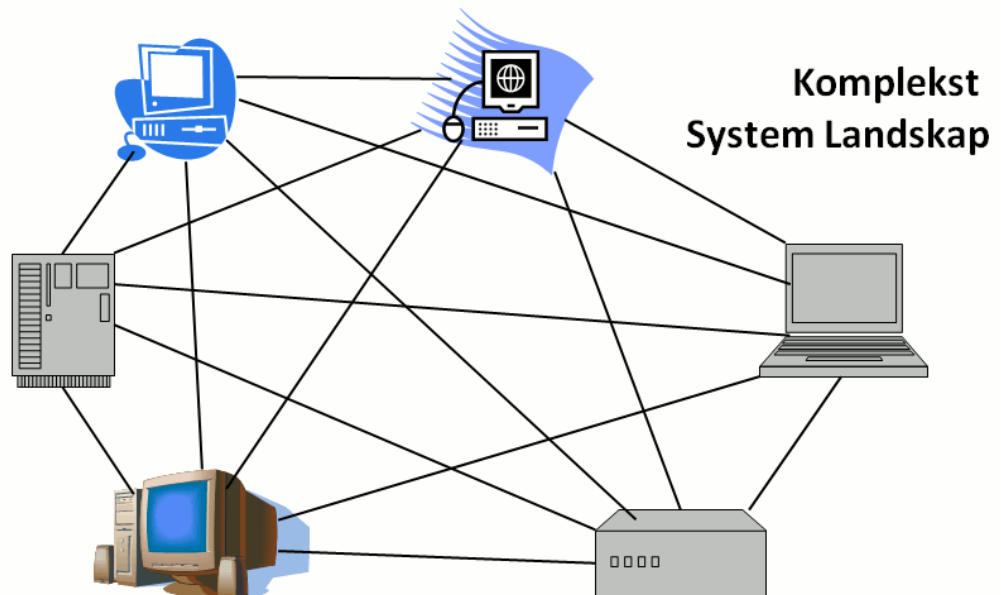
En bedrift som innfører SOA kan fort bli avhengig av eksperter med spesielle SOA-kunnskaper. Det er derfor viktig at bedriftene selv bygger opp slik kompetanse gjennom prosjektene.

Det finnes ingen "standard" SOA som man kan levere til en bedrift. Begrepet "Service Oriented Architecture" blir derfor ofte misbrukt i markedssammenheng av selskaper som lover å løse de fleste problemer med SOA mens SOA normalt ikke er i stand å løse alle faglige krav.

2 SAP PI som integrasjonsplattform

Integrasjonsscenarioer blir mer og mer kompliserte og omfatter ofte forskjellige applikasjoner på forskjellige plattformer forbundet med forskjellig teknologi. De enkelte systemene er tradisjonelt ofte bundet sammen med såkalte "Point-2-Point -forbindelser. Det betyr at alle integrasjonsfunksjoner er en del av applikasjonen. Hver ny integrasjon betyr en ny implementering av hele funksjonaliteten. Derfor finnes det mye forskjellig integrasjonsteknologi og flere steder for overvåkning. Det er vanskelig å endre noe fordi man først må analysere påvirkningen mot andre integrasjoner. Det trengs i tillegg ganske forskjelligartet kunnskap og derfor en rekke forskjellige ekspertise.

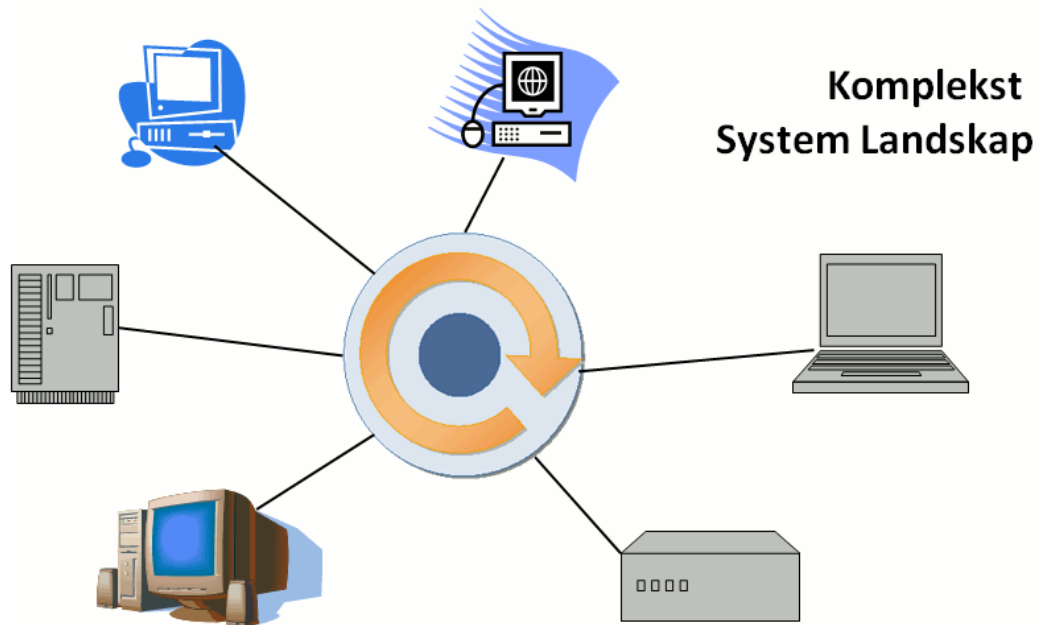
Eksempel: Et komplisert landskap med mange forskjellige applikasjoner uten integrasjonsplattform. Antall forbindelser vokser eksponentielt med antall systemer.



Bilde 4: Landskap med Point-to-Point forbindelser

Med en "middleware" (eller "integrasjons plattform" eller integrasjons "broker") forsøker man å få mer standardiserte og enklere integrasjoner. Hver kommunikasjon går nå over den samme integrasjonsplattformen, derved reduseres antall forbindelser og også omfanget i teknologi og nødvendig kunnskap. Der er nå lettere å gjøre endringer for alle integrasjonsfunksjoner da de finnes på et sentralt sted. Overvåkning følger en sentral strategi og kan bli planlagt.

Eksempel: Et komplisert landskap med integrasjonsplattform. Antall forbindelser vokser lineært med antall systemer.



Bilde 5: Landskap med PI som integrasjonsplattform

3 SOA for ERV (EDB Ressurs- og virksomhetsstyring)

EDB har gjennom mange år investert betydelige beløp i videreutvikling av administrative løsninger for offentlig sektor. Resultatet er et effektivt og helhetlig system basert på SAP.

Stadig høyere krav til ressursutnyttelse og tjenesteproduksjon legger press på det offentlige. For å løse disse utfordringene er man helt avhengig av effektive administrative prosesser, kombinert med tilrettelagt styringsinformasjon for ulike beslutningstakere. Dette sikrer involvering gjennom hele organisasjonen.

Offentlig sektor krever administrative løsninger som støtter integrerte prosesser, forenkler tjenesteorientert administrasjon og effektiv ressurs- og virksomhetsstyring, fra planer via realisering og tiltaksoppfølging. EDB Ressurs og Virksomhetsstyring er basert på **SAP** og er et effektivt og helhetlig system som ivaretar krav til effektiv transaksjonsbehandling, integrasjon, brukervennlighet og helhetlig styringsinformasjon.

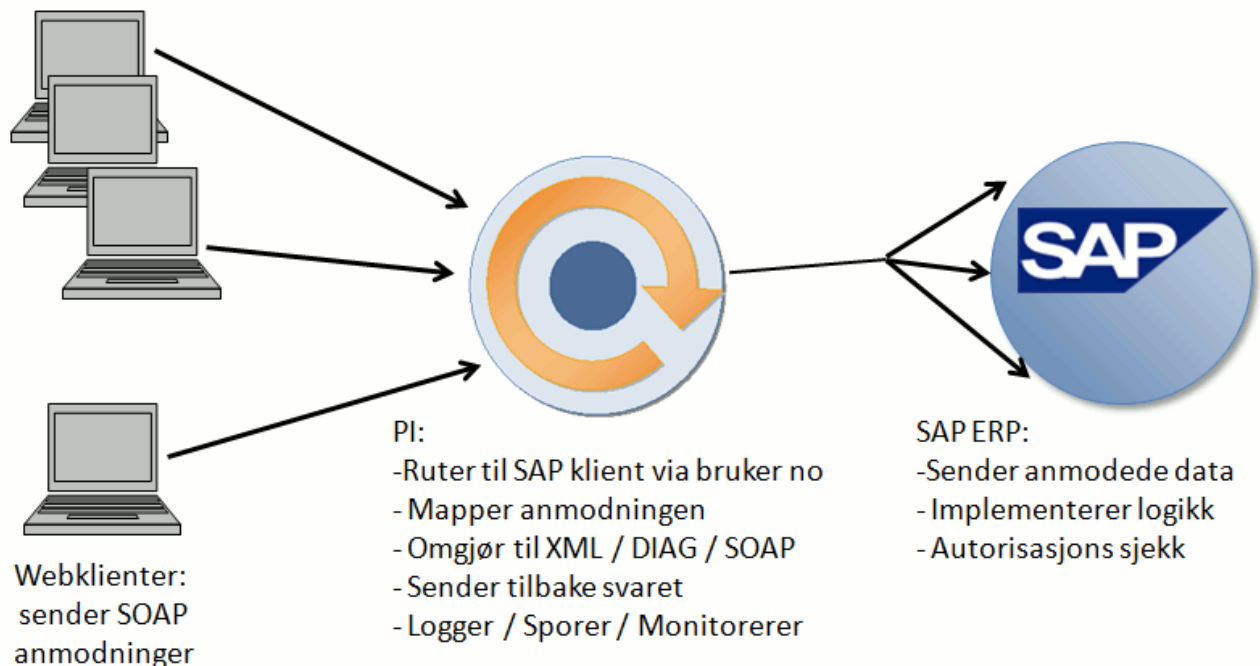
3.1 Moduler i EDB Ressurs- og Virksomhetsstyring:

- Virksomhetsstyring
- Reiseregning
- Økonomi
- Innkjøp
- Tidsregistrering
- Personal
- Fakturering
- Innlån / Utlån
- Lønn
- Plan og budsjett
- Prosjektstyring
- Anlegg

3.2 SOA Pilot

Første pilot SOA i ERV prosjektet var å tilby 7 HR infotypes som webtjenester. HR infotypes har nesten database-struktur i SAP ERP systemet hvor det finnes data for personaladministrasjon (Human Resources).

En webtjeneste ble benyttet fra forskjellige brukere som nå bare trenger WSDL filen for å implementere oppropet. Hver kommune er representert som en egen SAP-klient. I SAP PI benyttes en dynamisk mottakerbestemmelse for å finne den riktige SAP klienten avhengig av hvilken kommune forespørselen kommer fra..



Bilde 6: Oppgaver i ERV SOA pilot

Webgrensesnitt tilpasset systemer utenfor SAP (3.part systemer)

Det ble etablert et standard SOA tjenesteorientert webgrensesnitt som trekker ut endringer i SAP HR-objekter for bruk i 3. partssystemer.

Det ble laget webgrensesnitt for følgende SAP HR-objekter:

- Personopplysninger (personalstamdata, adresse)
- Stillingsopplysninger (flerstilling, organisasjonstilhørighet)
- Stillingskoder (stillingsbetegnelser)
- Organisasjonsopplysninger (organisasjonsstruktur)

Disse tjenesteorienterte webgrensesnittene gjenspeiler HR-strukturen slik de er i ERV - de såkalte Infotyper. Tjenestene er uavhengig av 3. part systemer som gjør at de er fleksible og gjenbrukbare for de som ønsker mer sammensatte tjenester enn det ERV tilbyr.

Kundene/brukere av tjenestene kan så komponere egne løsninger selv ved å benytte flere tjenester og sette disse sammen til den helheten de trenger.

Webgrensenittene leveres som webtjenester over HTTP eller HTTPS, dvs. standard eller sikker internettkobling som ikke krever tilpassning av brannmur. Tjenestene er basert på SOAP – en plattformuavhengig standard for utveksling av XML-meldinger.

Webtjenestene beskrives med WSDL, en XML-basert standardisert måte for rask og effektiv etablering av dialog mellom kunder og tilbydere av elektroniske tjenester. WSDL-ene kan importeres i en rekke integrasjonsplattformer, og kan benyttes for å generere programkode for å kalle tjenestene (Dette ble blant annet benyttet av Bergen kommune)

3.3 Webklienter

Web-klientene (sendere av web-beskjeder) bruker tjenester uavhengig av sted basert på SOAP-standard beskjed. Ved hjelp av en WSDL-filer er dette ikke komplisert. Adressen til serveren (PI i dette tilfellet) og strukturen sendes som en del av filen. Det finnes mange applikasjoner som gir muligheten til å bruke en webtjeneste med noen få klikk. Et freeware tool som kan bli brukt for å teste webtjenester er "soapUI". Man kan laste ned installasjonsfilen [her](#).

3.4 SAP ERP system som SOA tjenestetilbyder

SAP inneholder mange klienter for Norske kommuner basert på den samme templatene, men også delvis konfigurert forskjellig. HR-data (infotypes) tilbys via webtjenester til andre applikasjoner som også arbeider med disse HR-dataene.

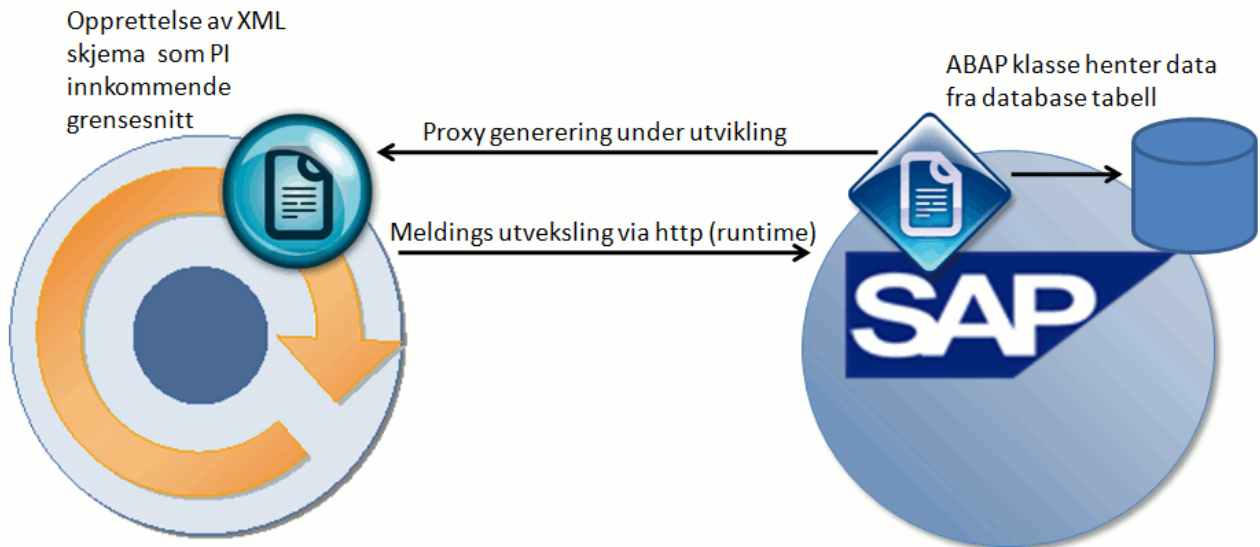
I SAP -systemet er dette realisert via ABAP-proxies som blir generert i PI-systemet. SAP-systemet har en direkte forbindelse til PI-systemet og kan lese grensesnittsdefinisjonen der. Man kan via transaksjon (SPROXY) liste alle grensesnitt fra PI-plattformen.

Med en kontekst knyttet til infotypene kan man lage en "proxy". Det genereres en ABAP klasse i ERP systemet som benyttes til å gi tilbake de etterspurte data. Klassen har allerede en "execute" metode som blir fylt med ABAP kode. Import parameter "input" kommer med request data, "export" parameter "output" må fylles med svar data.

Arbeidsfordelingen mellom ABAP og integrasjons spesialist blir ganske enkel. Integrasjonsspesialisten lager grensesnitt og generer proxyen, ABAP-programmerer lager ABAP-koden. Begge arbeider uavhengig av hverandre og trenger ingen kunnskaper om den andre plattformen. Tilnærmingen blir kalt "Outside-In", fordi man først lager grensesnittet uten applikasjonen og deretter importerer strukturen.

En annen eldre tilnærming "Inside-Out" finnes også i SAP-standard; Man lager først en RFC-modul (Remote Function Call) som deretter blir importert til PI hvor man kan generere et PI-grensesnitt (teknisk et XML Schema).

Proxy-kommunikasjon er den nyeste SAP-standard integrasjonsteknologien og er anbefalt for de fleste nyutviklinger.



Bilde 7: Proxy Kommunikasjon

3.5 SAP Process Integration

PI består av forskjellige applikasjoner (engines). De viktigste er Adapter Engine og Integration Engine. Adapter Engine kjører Java mens Integration Engine er skrevet i ABAP. Derfor har man også forskjellige brukergrensesnitt.

Man kan prinsipielt si at hver integrasjon krever tre deloppgaver:

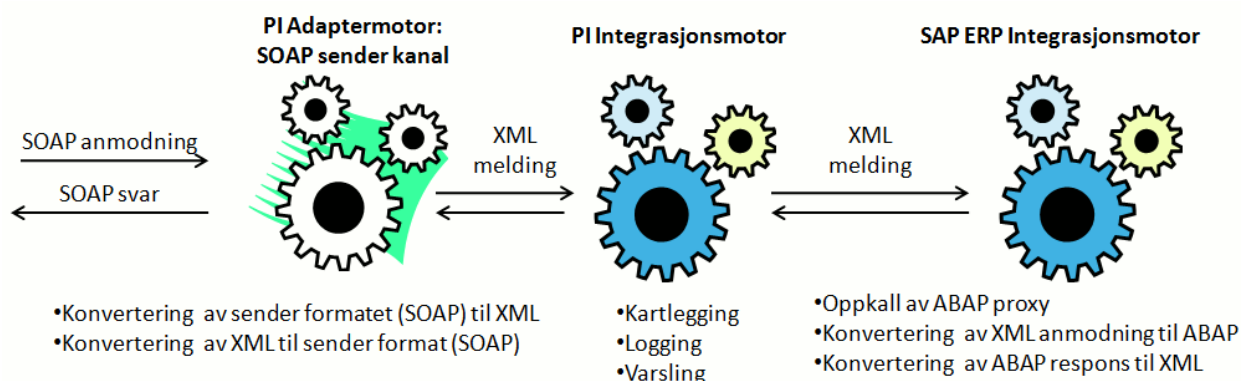
1. Motta anmodning og konvertere kildeforamt til XML
2. Mappe kilde-xml-struktur til mål-xml struktur
3. Konvertere mål-xml til mål-format

Adapter Engine kan motta en SOAP anmodning og oversetter til XML. Denne kanalen blir konfigurert ved at det sendes en bestemt signatur til Integration Engine.

Deretter blir signaturen meldingen identifisert og mapped med programmer som er laget i utviklings fasen. Også den blir routed etter regler som ble laget i configuration time.

Tilslutt blir XML sendt til Integration Engine av SAP ERP systemet. Her finnes det en ABAP klasse lik det grensesnittet har generert. ABAP koden i "execute_synchronous()" blir utført og henter nødvendige data fra databasetabellen.

Hele flyten er synkron og svaret blir sendt hele veien tilbake. I svaret finnes mapping og konversjon.



Bilde 8: Drift i PI Løpetid

4 Konklusjon

Sammenligner man tiden brukt til utvikling av anmodningslogikken som er programmert i ABAP i SAP applikasjonen, med tiden for utvikling og konfigurasjon av webgrensesnittene, så viser det seg at det å levere hvert grensesnitt som en uavhengig, standardisert webtjeneste kun krever små tilleggs anstrengelser.

Hvert gjenbruk av grensesnitt med en annen kunde (kommune i dette tilfelle) vil derfor gi en rask "return of investment".

Bruk av standardiserte grensesnitt viste seg å være ganske enkelt. Det var ikke nødvendig med noen forklaring når man utleverte en WSDL. Dette medførte at man har store kostnads og tidsbesparelser både i utviklingstid og opplæringstid for brukere.

PI var brukt som integrasjonsplattform, men dette er ikke nødvendig. Dette gjelder spesielt ved synkron meldingsutveksling. Fordelen ved bruk av PI er imidlertid at en får en veldig god overvåkning i tillegg.

En melding blir logget hele veien og det gis mulighet til hurtig å oppdage og analysere feil.

Tjenester er – som definert i SOA paradigmer – uavhengig og i seg selv lukket. Det betyr at utviklere av tjenestene og brukere på SAP ERP siden kan arbeide og teste uten at det trengs eksplisitte avtaler. Her ser man også store kostnadsbesparelser, raskere implementering og bedre kvalitet på løsningen.

Udo Martens

Senior konsulent/ SOA arkitekt
EDB Consulting Group