ÖPPEN TEKNISK PLATTFORM, ÖTP 3.0

REFERENSARKITEKTUR

Postadress
Föreningen Sambruk
Högbovägen 45
SE-811 32 Sandviken

Hemsida
www.sambruk.se

E-post
info@sambruk.se

Organisationsnr
802428-2785

Innehåll

Bilagor

-

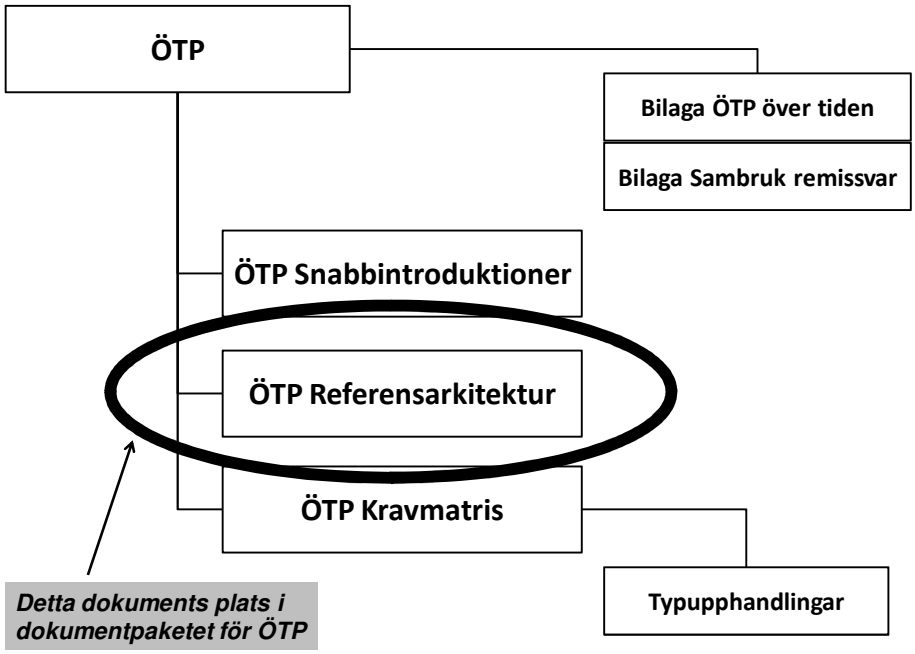
1. LÄSANVISNING

Föreliggande dokument ingår i dokumentpaketet som utgör version 3.0 av Öppen Teknisk Plattform (ÖTP).

Dokumentet är själva referensarkitekturen, samt innehåller även infrastrukturavsnitt.

För en överblick av ÖTP och en introduktion till referensarkitekturen hänvisas till paraplydokumentet Sambruk_OTP_v3_0.

För att däremot snabbt få en känsla för vad ÖTP är hänvisas till de snabbintroduktioner för olika läsargrupper som också ingår i dokumentpaketet.



1.1. Versioner

Datum	Ver.	Namn	Kommentar
2003-08-22	v1.0	Sven-Håkan Olsson	Utgåva av ÖTP v1.0
2004-03-22	v1.1	Sven-Håkan Olsson	Utgåva av ÖTP v1.1
2005-03-31	v1.2	Sven-Håkan Olsson	Utgåva av ÖTP v1.2
2007-09-19	v2.0	Sven-Håkan Olsson	Utgåva av ÖTP v2.0.
2011-03-08	v2.1	Sven-Håkan Olsson	Utgåva av ÖTP v2.1
2012-01-19	v3.0 dokv 1	Sven-Håkan Olsson	Påbörjat arbete med v3.0, separerad referensarkitektur
2012-01-27	v3.0 dokv 5	Sven-Håkan Olsson	Remissversion av v3.0
2012-02-03	v3.0 dokv 6	Sven-Håkan Olsson	Förslag till utgåva av v3.0

1.2.Referenser

Se referenslista i paraplydokumentet Sambruk_OTP_v3_0

2. INLEDNING

En referensarkitektur är en abstrakt arkitektur, som bl.a.innehåller **mallar** för hur en IT-lösnings konkreta arkitektur sedan utformas.

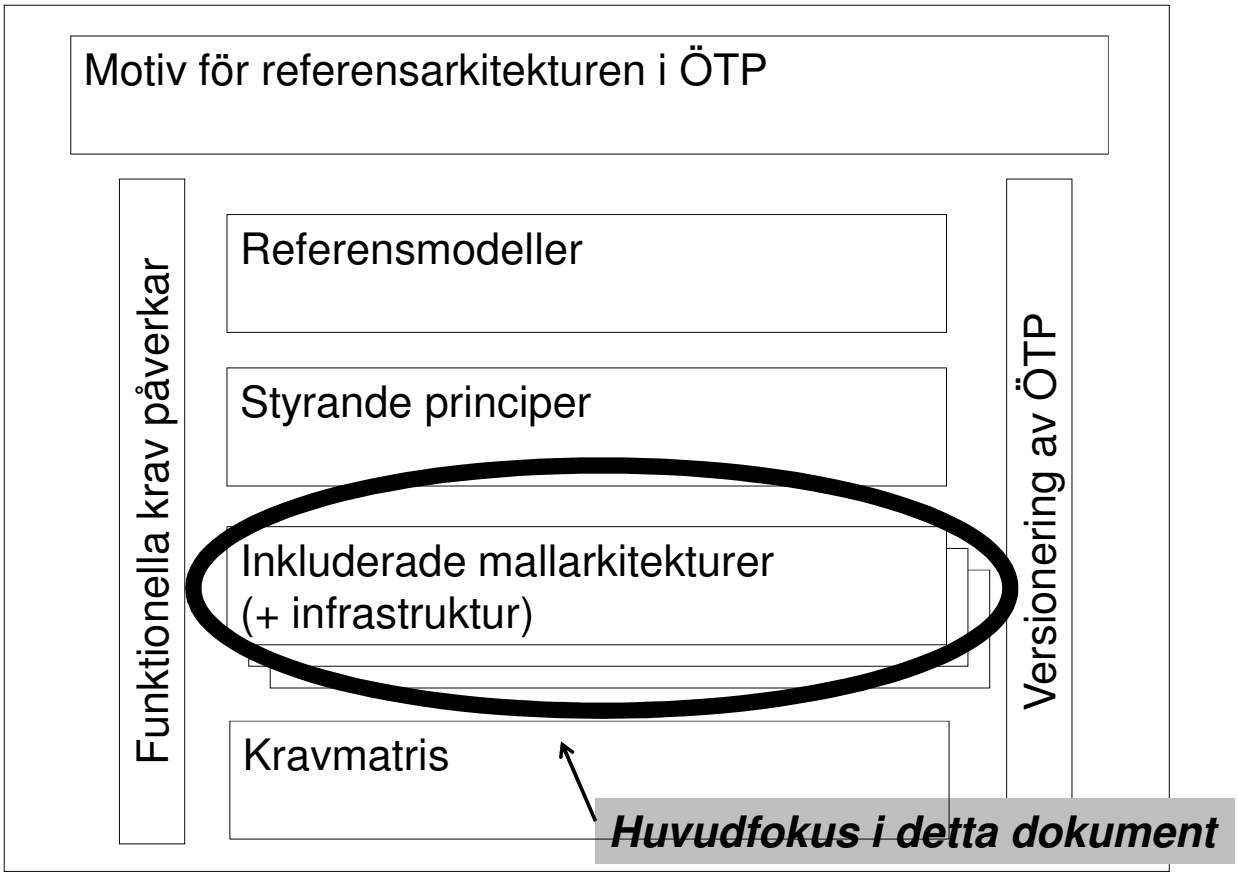
Det är viktigt att påpeka att ”one size does not fit all”, det går inte att föreskriva att en och endast en mallarkitektur ska användas i alla sammanhang - så enkel är inte verkligheten. Många faktorer påverkar de faktiskta valen i ett konkret projekt. Först och främst måste man utgå från funktionella önskemål. Därefter ska man skapa en arkitektur som ger önskad balans mellan (ibland motstridiga) önskemål såsom datafärskhet, datakvalitet, stabilitet, prestanda, flexibilitet, kostnad, existerande verksamhetsapplikationers egenskaper, IT-strategival mm. Mallarkitekturerna är alltså endast något att utgå ifrån, som mönster.

Vid ett anskaffningsprojekt (upphandling, avrop, utveckling etc) ska vanligen en kravspecifikation skapas. Därför finns inom dokumentpaketet för ÖTP även en kravmaster och typupphandlingar. I kravmastern finns en palett av krav som man kan välja bland, men först när man väl har valt mallarkitetktur(er). Gången blir alltså enligt följande:

Figur 1 Arbetsgång för att skapa kravspecifikation

Förutom huvudkapitlet om mallarkitekturer nedan följer även kapitel som är mer infrastrukturnära: Rörläggning, Principer för Nyttomeddelanden och Regler, konventioner, drift. Anledningen är bl a att det är extremt viktigt att använda öppna och interoperabla kommunikationssätt för att välfungerande applikationssamverkan ska kunna uppstå, och därmed större verksamhetsnytta och medborgarservice.

ÖTP:s referensarkitektur



Figur 2 Referensarkitekturen

De avsnitt som beskriver ”kringförhållandena” för referensarkitekturen har vi valt att lägga i paraplydokumentet Sambruk_OTP_v3_0, medan föreliggande dokument fokuserar på de faktiska mallarkitekturerna.

2.1. Översikt referensarkitektur applikationsintegration generellt

I nedanstående mallkapitel med arkitekturbilder och tillhörande kommentarer anges ett antal arkitekturprinciper och modulariseringssätt. Eftersom många avvägningar måste göras mellan önskemål om hög flexibilitet och att vara konkret och tydlig, inkluderas också en del orsaksresonemang i samband med arkitekturskisserna, för att ge kravspårbarhet. Plattformen och arkitekturen måste självklart versioneras och utvecklas över tiden och därför är det extra viktigt att veta *varför* olika val togs vid olika tidpunkter.

Först behandlas översikter, sedan följer renodlingar av ett antal aspekter.

2.1.1. Inkapsling – betoning på gränssnitt

Att utforma alla detaljer inuti alla applikationer i enlighet med exakt samma riktlinjer är i princip omöjligt i en kommun. Oftast har man inte råd att skräddarsy applikationer utan man

får köpa färdiga standardapplikationer. Dessa baserar sig på varierande plattformar. Även om man skulle specialutveckla applikationer så fortgår teknikutvecklingen så att en nyskriven applikation lämpligen inte använder exakt samma plattform som en fem år gammal applikation, trots att den sistnämnda kan vara ge god nytta flera år ytterligare. Dessutom tillkommer fall där man startar kommunsamarbeten (såsom inom Sambruk, SKL, mellan grannkommuner, inom kommunalförbund etc) där man vill dela på applikationer, och då skulle inte alla viljor kring innanmätesdetaljer kunna uppfyllas.

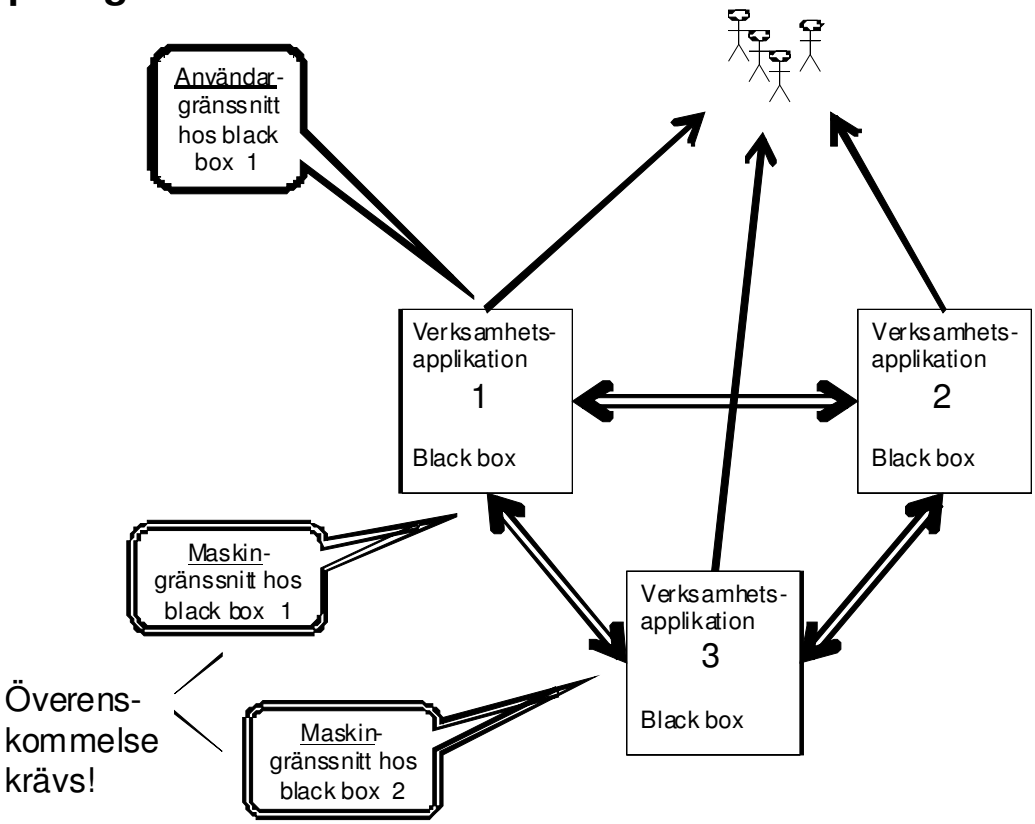
Det har därför varit en fruktbar trend i många år, både inom näringsliv och kommuner, att låta omgångar av applikationslogik tillåtas få utformas på varierande sätt men inkapslat i s k svarta lådor eller ”black boxes”. Det som är det viktiga är inte exakt hur innanmätet råkar vara utformat, utan istället:

- Vilken verksamhetsnytta som den svarta lådan ger
- Vilket ev. användargränssnitt den har
- Vilka ev. maskingrässnitt som den interagerar med andra applikationer och parter via.

I ÖTP-dokumenten skriver vi vanligen verksamhetsapplikation och menar då samtidigt att en sådan ska utgöra en svart låda, med väldefinierade gränssnitt. Detta tankesätt har utvecklats inom IT- och verksamhetsutvecklingsområdet under namnet SOA: Service Oriented Architecture, eller tjänsteorienterad arkitektur. Inom SOA-skrifter skriver man ofta ”tjänst” för att mena svart låda (eller ibland mindre block, en del av en ”applikation”), men ordet tjänst används synnerligen inkonsekvent i IT-branschen så vi undviker ordet inom ÖTP. Möjligen använder vi begreppet SOA-domän för att markera att något håller ihop som en svart låda, har väldefinierade gränssnitt samt handhar något visst verksamhetsområde – ”verksamhetsdomän”.

En mycket viktigt aspekt är att det måste finnas en ”ägare” till varje black box, som har ett tydligt ansvar för innanmätet i den. Detta ansvar gäller även för de maskingränssnitt och användargränssnitt som publicerats av den svarta lådan. För att definiera behoven av sådana gränssnitt behövs självklart en kravprocess där verksamhetspersoner och ansvariga för andra black boxes deltar, men när väl gränssnitten är definierade ska black box-ägaren tydligt ansvara för dem.

Inkapsling – black box



Figur 3 Inkapsling – black box, och gränssnitt

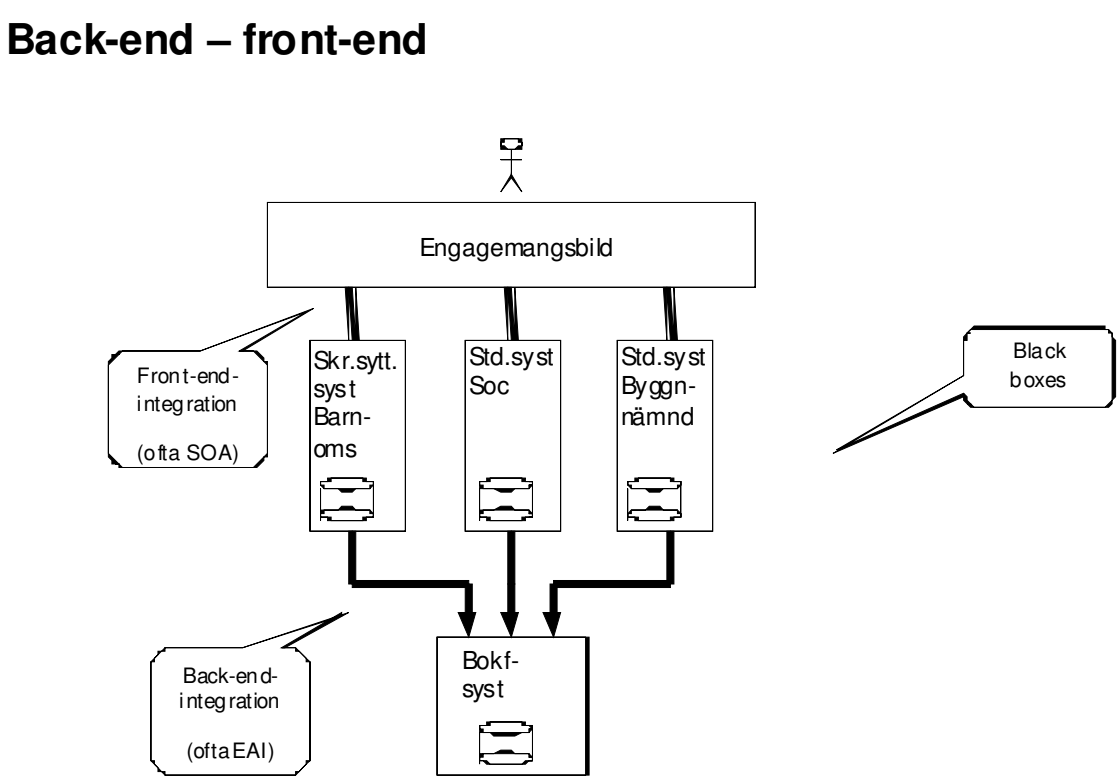
De flesta applikationer har användargränssnitt – dessa kan sägas vara en överenskommelse mellan människa och system. Sambruks Nyttomeddelanden handlar i hög grad om att också definiera maskingränssnitt av det slag som visas i figuren ovan. I det fallet handlar det om att skapa en överenskommelse mellan två system som vardera utgör black boxes.

I ökande grad hoppas vi också att inköpta verksamhetsapplikationer har färdiga maskingränssnitt redan från början, i efterföljd av det som för många år sedan skedde inom näringslivet, och i enlighet med SOA-vågen som gått genom IT-världen.

Man inser ur figuren ovan att ett vanligt fall kan bli att färdiga maskingränssnitt från två olika black boxes som behöver kommunicera med varann inte direkt är kompatibla, det finns ingen direktfungerande överenskommelse. Någon form av översättning måste då göras. I ÖTP kallar vi normalt komponenter som gör sådan översättning för adaptrar.

2.1.2. Integration nära användargränssnitt eller ej

Samfunktion, integration, mellan olika verksamhetsapplikationer nära användargränssnittet behöver vissa typiska egenskaper medan integration längre bak i kedjan behöver andra.



Figur 4 Back end – front end

Front-end-integration har under några år nu i hög grad utgjorts av Service Oriented Architecture (SOA), oftast utfört med Web Services[1] som teknik. Ofta behöver användaren information som är färsk på sekunden, som i en Internetbank eller i en skolschemaapplikation där en lärare kanske lägger in att man ska vara i ett anant klassrum.

Back-end-integration har historiskt brukat kallas Enterprise Application Integration (EAI)[2]. Datafärskhetskrav i klass med sekundsnabbhet har sällan funnits med i kravbilden för EAI. Inom kommunvärlden har en dominant integrationsprodukt (Decapus) varit av enklare slag och knappast kunnat kallas EAI, men produkten har funktionalitet för att få grundläggande fil-integration att fungera (vilket ibland är fullt tillräckligt).

[1] WS: Web Services, ett sätt att återanvända webbprotokoll för interoperabla maskin-till-maskin-anrop
[2] EAI: Enterprise Application Integration, integration som betonar överföring av större dataflöden

I mitten på 2000-talet har ett begrepp dykt upp, Enterprise Service Bus (ESB)[3] som förenklat innebär EAI fast som dessutom kan gå sekundsnabbt och klara prenumeration på information. Emellertid går dessa världar ihop nu, EAI-plattformar klarar ofta SOA och har ESB-egenskaper t ex. Dock är det inte säkert att de är fullständigt optimerade för detta, de kan också vara onödigt bra och därmed ha en dyr prislapp. Ett stort antal kommuner har också uppgraderat från Decapus till den nyare produkten TEIS som har vissa EAI/ESB egenskaper, så denna produkt har också en slags särställning i praktiken idag. Några kommuner har skaffat andra integrationslösningar, kanske även som komplement till Decapus/TEIS. Ibland ingår någon integrationslösning i köpet av en applikation. Det är idag därför inte ovanligt att kommunen måste hantera en ”federation” av integrationslösningar, och avväga exakt var i kedjan som konvertering etc ska utföras. Se särskilt kapitlet *Applikationsintegration EAI, ESB* nedan. Det bör också nämnas att olika s.k. metakataloglösningar ofta har egenskaper liknande EAI/ESB.

SOA brukar utgöra s k funktionsintegration där man ser det som att programkod för verksamhetslogik körs i serverändan när ett anrop ankommer, och att i många fall ett svar skickas tillbaka. Dataintegration däremot är gamla EAI:s kärnområde där det vanligen endast är information som ska skickas från ett system till ett annat, utan att man inom integrationen fokuserar på exekvering av verksamhetslogik. Genom dessa skillnader passar SOA extra bra för front-end-integration och EAI extra bra för back-end-integration, även om detta börjar flyta ihop mer idag, t ex vad gäller ESB.

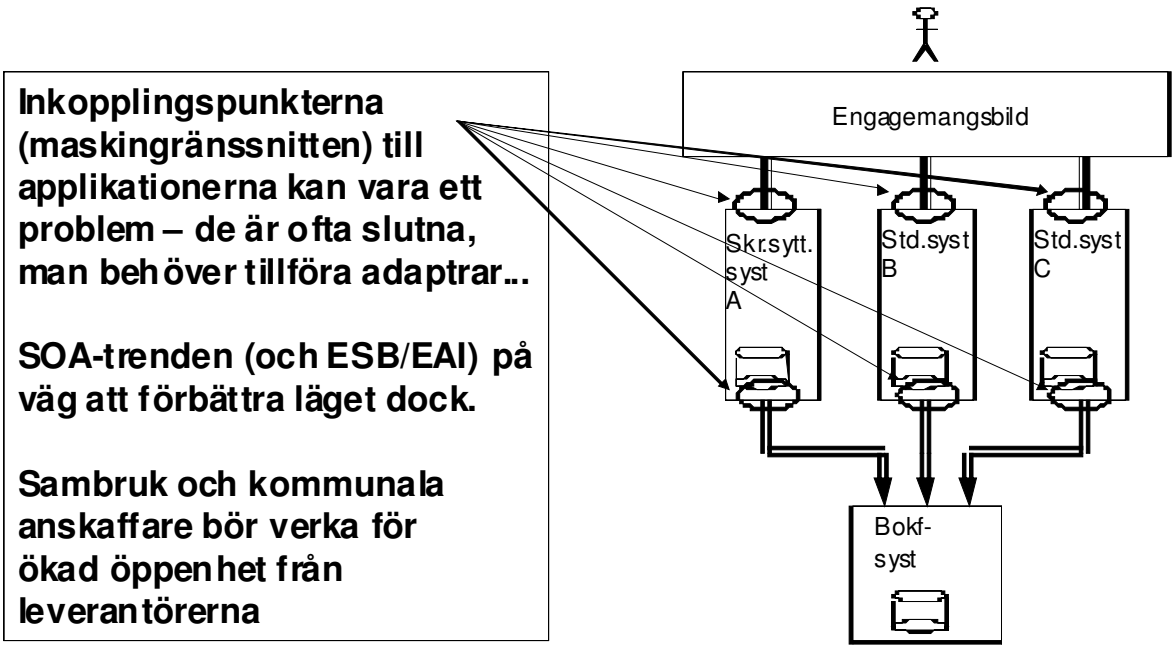
Om nu införande av integrationslösningar bedöms lämpligt, vad står man sedan inför för utmaningar? Det svåraste brukar vara själva öppenheten, inkopplingen i stuprörssystemen. Ifall har kommunen egna skräddarsydda applikationer så kan man själv utveckla maskingränssnitt, adaptrar etc. Vanligast i kommunvärlden är dock standardapplikationer och dessa är tyvärr ofta inte så öppna idag.

Denna typ av öppenhet är däremot redan mycket vanlig inom näringslivets standard-applikationer. Det finns därmed god erfarenhet av hur detta kan utformas tekniskt, oavsett man har använt äldre teknikmiljöspecifik kommunikation (som t ex Microsoft DCOM eller Java-RMI), eller interoperabla Web Services med XML-meddelanden (såsom enligt ÖTP:s specificerade Nyttomeddelandestruktur vilka följer Statskontorets / Vervas / Kammarkollegiets s k Standardmeddelanden). Inom kommunsektorn har alltså standard-applikationerna släpat efter ur öppenhetssynvinkel. Vi förväntar oss dock att detta ska kunna lösas, dels baserat på generella systemutvecklingstrender, dels grundat på Sambruks och andras påverkan och upphandlingstryck.

Nyttomeddelanden tänks specificeras för att vara s k transportoberoende. Det innebär att en och samma meddelandedefinition ska gå att använda genom helt olika sorters datakommunikation, beroende på aktuella behov. Ifall filöverföring är lämpligt av olika skäl så bör Nyttomeddelandet gå att transportera på det sättet. Ifall Web Services, SHS, Decapus etc är lämpligt bör Nyttomeddelandet gå att överföra på de sätten, allt efter behov. Det som avgör är ofta ifall verksamheten behöver sekundfärskt data eller inte, liksom vilka kommunikationsprodukter som redan finns tillgängliga. Se även det separata kapitlet om Nyttomeddelanden.

[3] ESB: Enterprise Service Bus, enkelt uttryckt EAI plus större snabbhet och vanligen ej nav-struktur

Öppenhet – inkopplingspunkter - maskingränssnitt

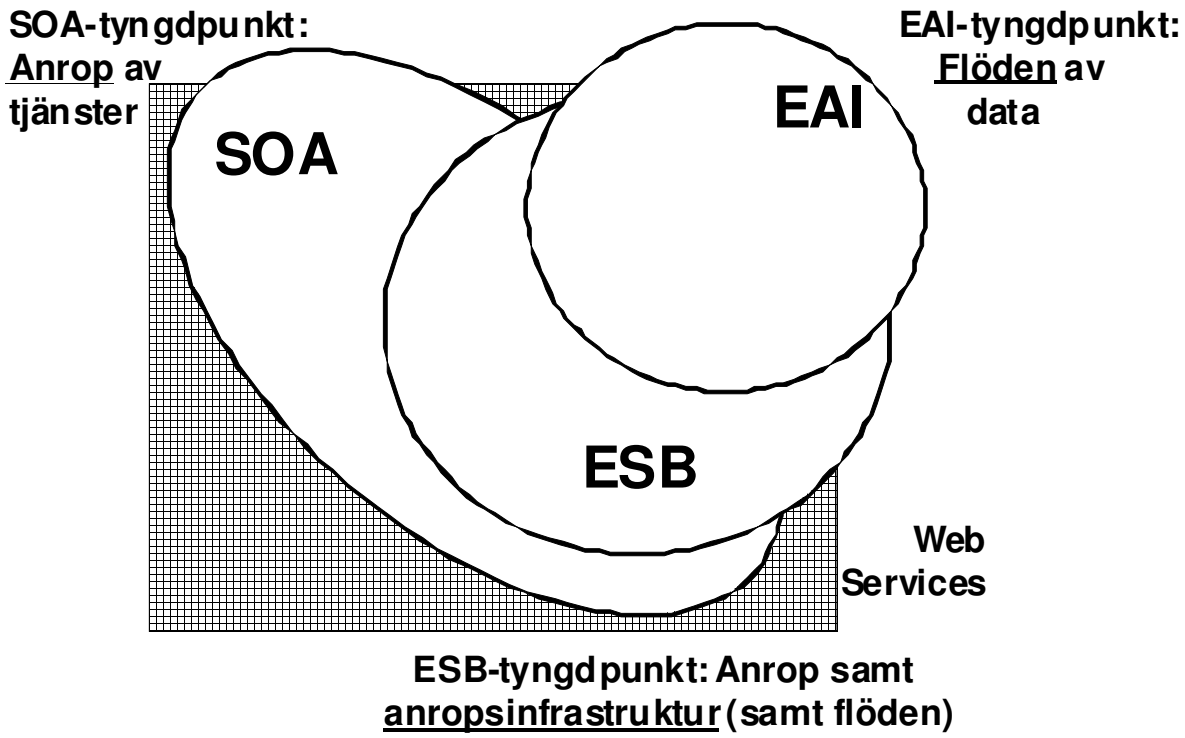


Figur 5 Problem med bristande maskingränssnitt

Det är värt att notera att de generella integrationslösningarna ofta inte idag innehåller adaptrar eller färdiga inkopplingspunkter (maskingränssnitt) för kommunala verksamhetsapplikationer, dessa måste då tillföras av applikationsleverantören eller tredje part etc vilket skulle ge kostnader och förvaltningsspörsmål. ÖTP rekommenderar därför en viss skepsis relativt kostnads/nyttobalansen för integrationsmotorer såsom EAI och ESB. Se även de mer detaljerade kapitlen nedan, såsom *Applikationsintegration EAI, ESB.*

Sammanfattningsvis, ÖTP förordar starkt ett tänkande där hängrännor och liknande tvärfunktionella lösningar kan åstadkommas. Därvid ska SOA-principen vara grundläggande. Huruvida EAI eller ESB-plattformar behövs måste däremot bedömas beroende på aktuell behovsbild eftersom de har en hög kostnad och komplexitet att väga mot deras fördelar.

Följande bild beskriver hur vi i ÖTP ser att EAI, ESB, SOA och Web Services relaterar till varandra (och vilka överlapp som finns):



Figur 6

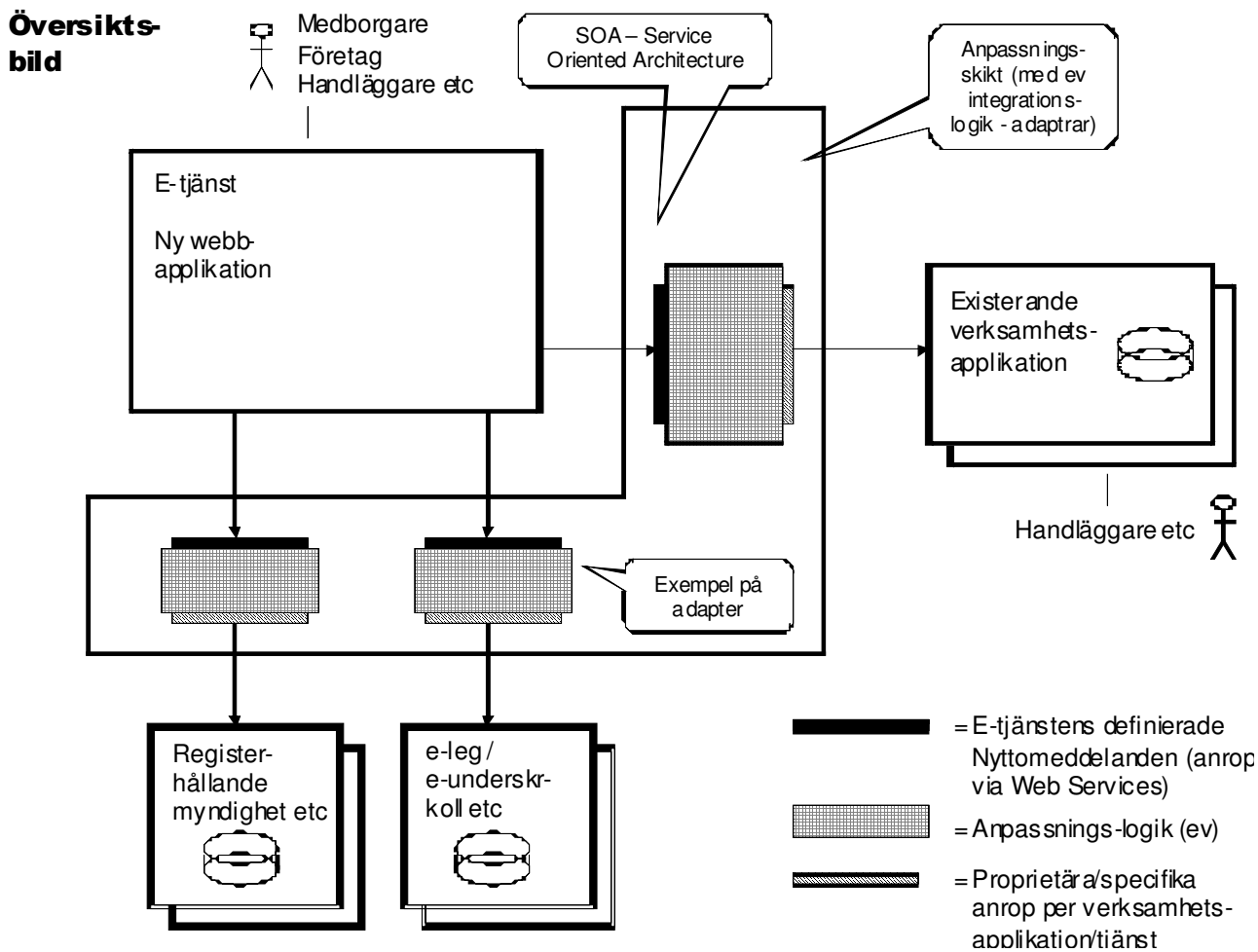
3. MALLARKITEKTURER INOM REFERENSARKITEKTUREN

Nedan följer ÖTP-referensarkitekturens olika mallarkitekturer. Tanken är alltså att man beroende på funktionella behov och andra krav ska identifiera en relevant mallarkitektur (eller i vissa fall flera) som man sedan kan vidare beskriva och detaljera relativt de aktuella kraven så att det blir en konkret och införbar arkitektur.

3.1. Tjänsteorienterad arkitektur, SOA

Här följer en mallarkitektur som utgår från Sambruks tidiga utredningar, och som framförallt lyfter fram och fokuserar på integrationslogiken som behövs för e-tjänster. Detta eftersom det är en av de svåraste nötterna att knäcka för att åstadkomma modularisering/komponenter, återanvändbarhet och interoperabilitet för deltagande kommuner (vilka har gjort varierande interna val av applikationer och teknikbas). Mallarkitekturen fungerar dock ofta även för kommuninternt verksamhetsstöd.

SOA-principen tillsammans med Web Services är det prioriterade mönstret för e-tjänster i ÖTP, framförallt för front-end-integration, enligt följande bild:



Figur 7

Kommentarer:

- E-tjänstens webb-applikation och en stor andel av adaptrarna ska kunna vara gemensamma för flera kommuner (trots att dessa kan ha gjort varierande interna val, t ex vad gäller verksamhetsapplikationer). Förutom att källkod/produkt ska kunna vara gemensam i sig så bör den vara utförd för att kunna samdriftas på ett rationellt sätt för flera kommuner (trots dessas varierande interna val) – ASP/SaaS-typen av drift.
- Fokus i denna bild är online-situationen snarare än stora asynkrona, batchliknande flöden (för det sistnämnda, se separat kapitel).
- E-tjänstens definierade Nyttomeddelanden måste dels vara specifierade till syntax och semantik (funktionella egenskaper), dels till teknik (icke-funktionella egenskaper). För tekniken finns aspekterna rörläggning och regler och konventioner mm (se dessa kapitel). För tekniken gäller också, per Nyttomeddelande, de egenskaper som grupperas ihop inom s k kommunikationsprofiler.
- Services Oriented Architecture (SOA), baserat på enkla Web Services och s k Nyttomeddelanden, är grundstommen i ÖTP.
- Med SOA menas i ÖTP ett fokus på anrop (stora meddelandeflöden behandlas alltså i separat kapitel). Ovan försöker vi även ge en översiktsbild där både anropande och anropade parter samt eventuella mellanliggande adaptrar visas. Om man skulle se det strikt ur endast anroparens synvinkel skulle man mer fokusera på inkapsling och "black box". En huvudsak är dock betoning av överenskommelser (informationskontrakt) i anrop, i form av Nyttomeddelanden.
- Den L-formade SOA-delen av skissen skulle kunna vara kandidat för att utföras med någon form av Web Services Management-plattform, ESB (Enterprise Service Bus) etc, men det är i många fall helt rimligt att *inte* använda någon sådan plattform eftersom Web Services-exekvering i princip ingår i alla moderna plattformar i sig. Se separat kapitel.

3.2. Olika behov av anpassningslogik mot verksamhetsapplikation

3.2.1. Direkta anropsgränssnitt

Anpassningslogiken för att integrera en viss e-tjänst med den verksamhetsapplikation (i undantagsfall flera) som en kommun använder sig av, blir med nödvändighet av olika omfattning beroende på respektive applikations möjligheter (både applikationens funktionella och tekniska egenskaper).

I Sambruks äldre projekt Borlänge-piloten som exempel, så heter e-tjänsten Förnyad ansökan om ekonomiskt bistånd. Specifikationsprojektet definierar ett antal Nyttomeddelanden som behövs för att webbapplikationen ska kunna förses med information från verksamhetsapplikationen, samt kunna uppdatera densamma. I exemplet Borlänge kommun användes verksamhetsaspplikationen Sofia.

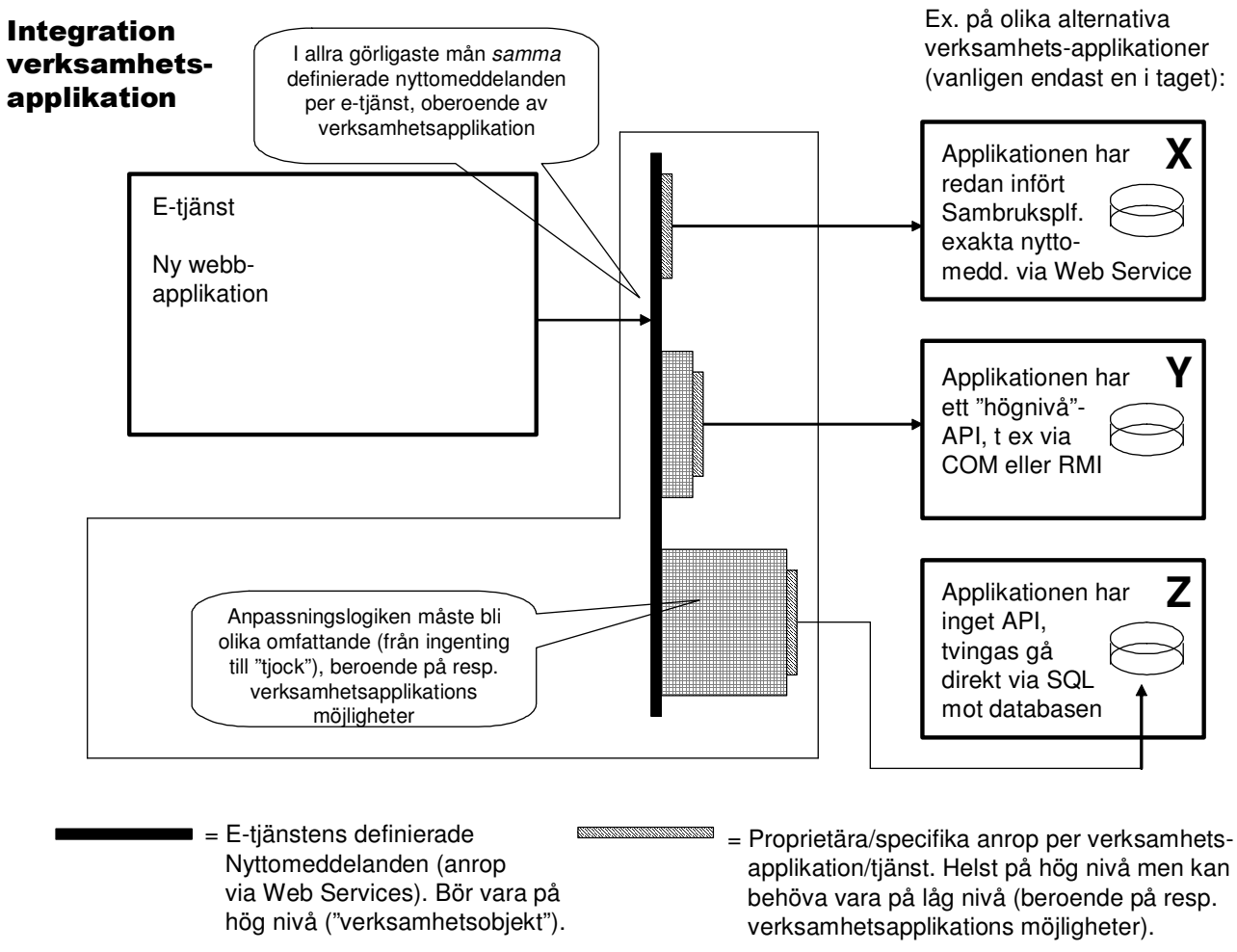
Ambitionen var självklart att Borlänge (eller en annan kommun som ville sambruka e-tjänsten Förnyad ansökan om ekonomiskt bistånd) istället för Sofia skulle kunna använda en annan typisk verksamhetsapplikation utan att de definierade Nyttomeddelandena skulle behöva ändras. Dock får man vara realist och inse att alltför stor variation inte på förhand går att förutse eller täcka in, då bleve Nyttomeddelandena så oprecisa och abstrakta att de inte vore praktiskt användbara. Man måste alltså vara beredd på att över tiden förvalta definitionen av Nyttomeddelandena och skapa nya versioner allteftersom kravbilden ändras. Det blir därmed en mycket viktig framgångsfaktor att organisationen kring Sambruk klarar av detta förvaltningsansvar.

I samverkan med näringslivet eller tänkbara andra konstellationer som utvecklar verksamhetsapplikationer finns det en förhoppning att en standardiserande verkan uppkommer vad gäller Nyttomeddelandena. Detta är viktig roadmap-information till leverantörer. Helst finns därmed Nyttomeddelandena (exakt som de är definierade i Sambruks respektive e-tjänst) redan färdigimplementerade i verksamhetsapplikationen. Anpassningsskiktet består i detta gynnsamma fall (alternativ X i figuren nedan) endast av en specifikation – ingen programkod eller mellanprogramvara behöver skrivas/anordnas, underhållas eller driftas.

I andra fall så innehåller verksamhetsapplikationen någon annan form av proprietärt API[4] som t ex baserats på tekniska miljöer som Microsofts COM/Dotnet, på Java RMI eller på Web Services. Detta innebär att anpassningslogikens programkod måste utvecklas, underhållas och driftas i respektive teknisk miljö (alternativ Y i nedanstående figur). Hur omfattande anpassningslogiken blir är naturligtvis synnerligen beroende av respektive API:s nivå och funktionalitet. En variant av alternativ Y som något närmar sig X är att det finns ett API som tekniskt sett inte är proprietärt utan baseras på Web Services, men att det inte är exakta Nyttomeddelanden som är publicerade av applikationen, utan andra meddelanden. Då får man i alla fall tänka sig en adapter, men den blir enklare. Självklart finns dock fallet att det API som applikationen publicerar inte alls har det funktionella innehåll som behövs och då får man antingen försöka påverka leverantören att tillföra detta, eller också ta till alternativ Z e dyl.

[4] API: Application Programming Interface – maskingränssnitt för anrop emellan program

I ytterligare andra fall innehåller inte verksamhetsapplikationen något API för online-bruk överhuvudtaget. Anpassningsskiktet måste då bli ganska tjockt och beroende av den teknik som applikationen råkat vara skriven med. Ett typiskt scenario är att applikationen har en SQL-databas vars datamodell är publicerad, går att studera och förstå, varefter anpassningslogik måste programmeras på låg nivå, direkt gentemot SQL (alternativ Z i figuren nedan). En mellanvariant är att befintliga eller nyskrivna Stored Procedures i databasen används i stället för ren SQL.



Figur 8

Kommentarer:

- Det är viktigt att Nyttomeddelandena definieras på samma nivå som man brukar mena med ”verksamhetsobjekt”. Det bör alltså inte vara ”rörlednings-meddelanden” eller liknande låg nivå, utan de bör heta t ex laes_medborgarinfo och ha explicit definierade in/ut-parametrar. SOA-begreppet grov-granuläritet beskriver också detta önskade beteende. Motsvarar ”svart skikt” i figuren ovan.
- Meddelanden gentemot verksamhetssystemen kan behöva vara på lägre nivå, t ex via ett ”rörlednings-API” eller SQL. Motsvarar ”snedstreckat skikt” i figuren ovan.

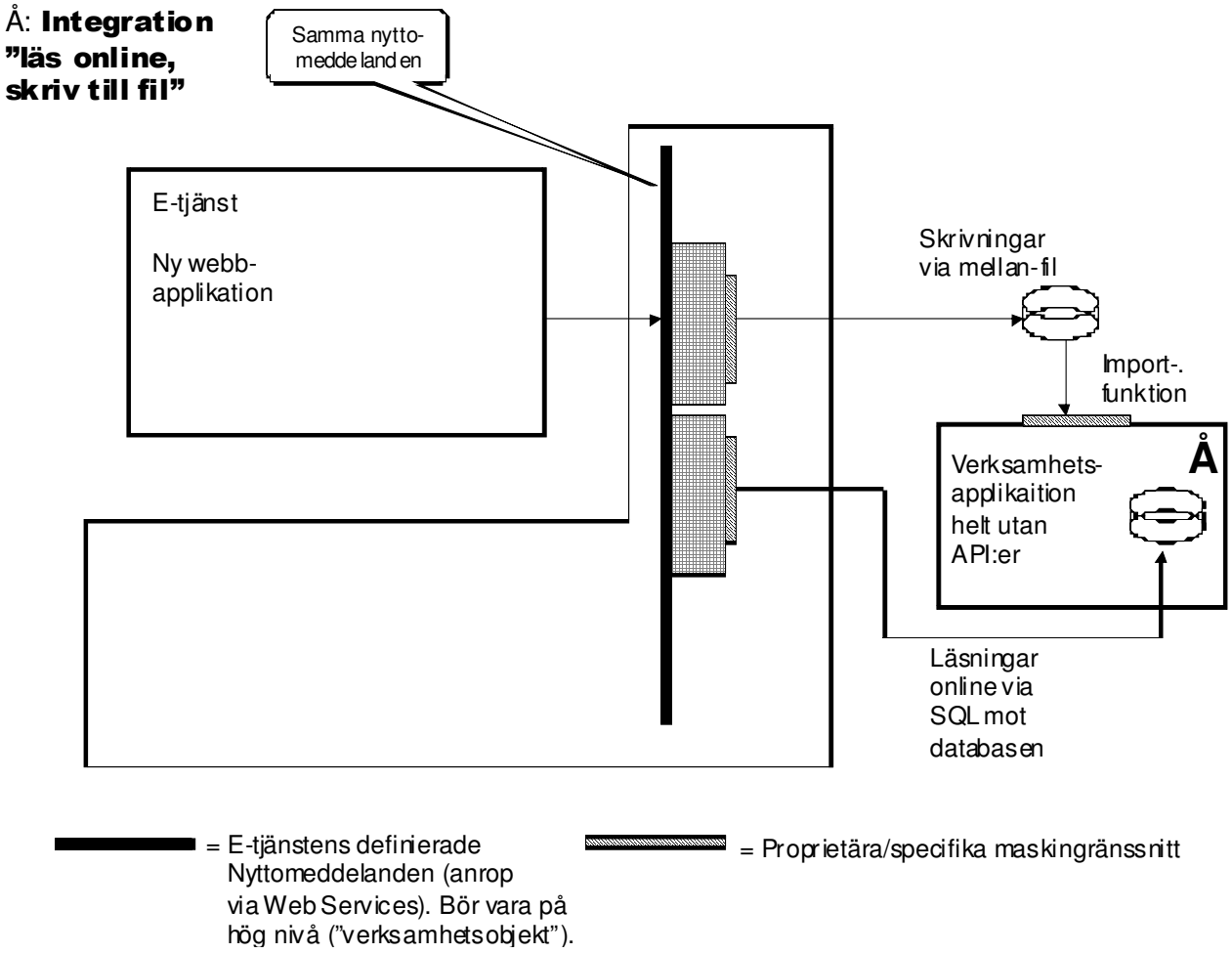
- Ingen allt-eller-inget-uppdatering (s k ACID) är möjlig (eller lämplig) med dagens Web Services som ju är vald teknik i ÖTP. Kompenseringsrutiner måste finnas hos aktuell kommunen för att kunna hantera eventuellt uppkomna diskrepanser, liksom information om hur loggar används för att kunna felsöka etc. Detta måste föreslås utav leverantör (motsvarande) till e-tjänst-applikation respektive anpassningslogik.
- Alternativ Z har självklart nackdelen att maskingränssnittet kan fallera ifall leverantören av verksamhetsapplikationen uppgraderar sin datamodell så den inte längre stämmer med adaptern.
- Adaptern kan i värsta fall behöva innefatta egen datalagring ifall verksamhetssystemet är alltför simpelt för Nyttomeddelandenas krav. Se även kapitlet *Var datalagring sker*.

3.2.2. Mönstret "läs online, skriv till fil"

En reservlösning ifall man inte hittar vettiga sätt enligt ovan eller att applikationsleverantören inte ger rimlig offert på API eller adapter är en hybridlösning:

- E-tjänsten *läser* direkt i verksamhetapplikationens databas när den behöver läsa data. Detta kan man göra online och man kan därmed t ex åstadkomma bra indatakvalitet genom god datavalidering i användargränssnittet.
  Motsvarar mönstret Z ovan, dvs integration direkt mot SQL.

- E-tjänsten *skriver* till en mellanfil när den behöver skriva data. Filen importeras vid senare tidpunkt via batch in i verksamhetsapplikationen, det blir alltså inte äkta online. Troligen kan dock fördröjningen hållas låg, kanske bara några minuter. Detta blir då en något svagare funktionalitet än att vara helt online, men motsvarar ändå stor snabbhet jämfört med pappersflöden.



Figur 9

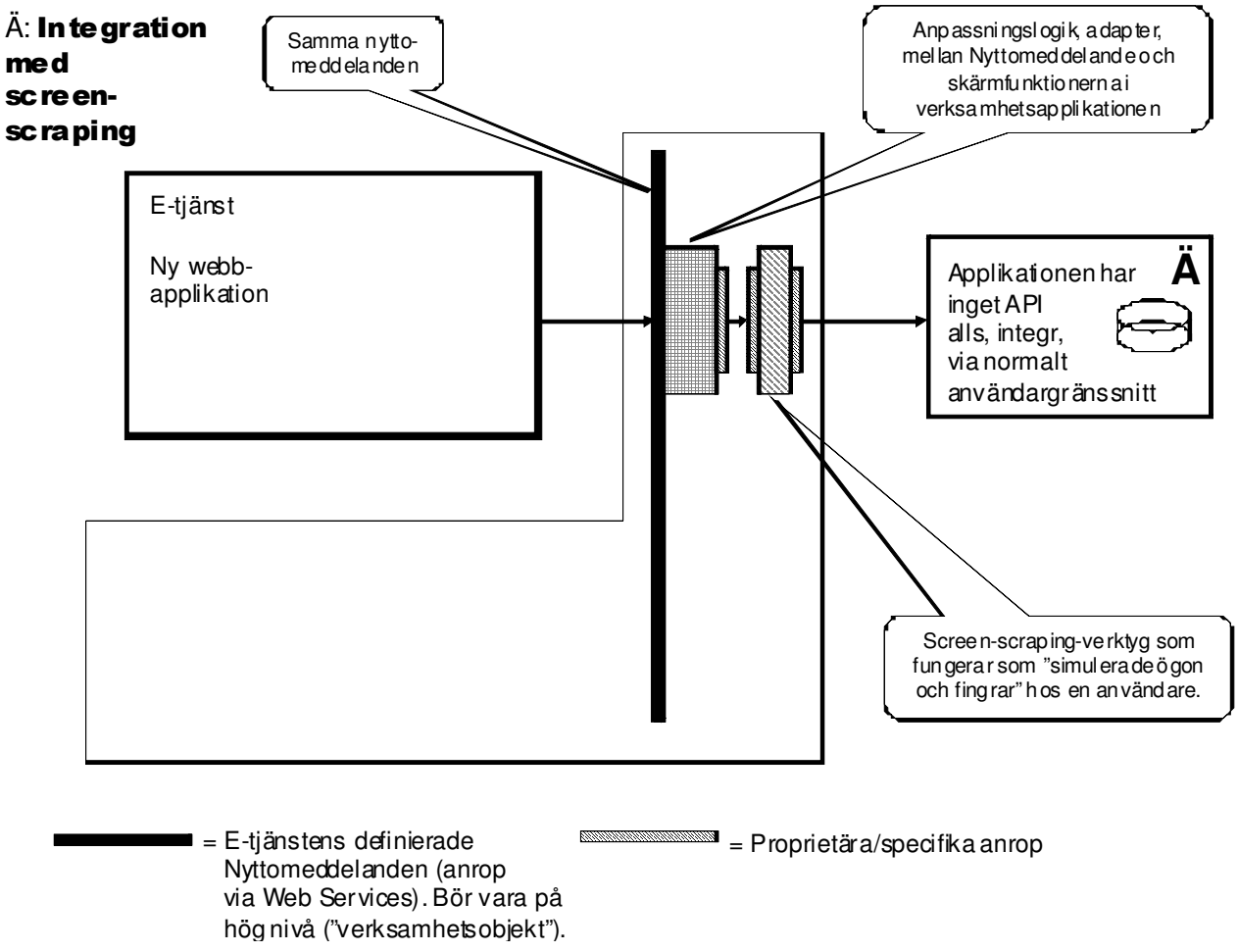
Kommentarer:

- Kräver ingen leverans av adapter eller nyttomeddelandeimplementation från applikationsleverantör, den *läsande* adaptern kan man anskaffa från tredjepartsleverantör e.dyl..
- Frågan är hur man ordnar funktionen för batchimport, den bör ofta inte en tredje part skapa eftersom man ev kunde få problem med garanti/support mot leverantören av verksamhetsapplikationen. Å andra sidan förekommer det ofta att sådan batchimport (med rätt data i) redan existerar. Dessutom har leverantörerna vanligen utvecklat ett antal batchimporter åt kommuner genom åren, så det ska inte vara någon stor eller märkvärdig fråga för dem, de ska rimligen kunna återanvända programkod de redan har.
- Andra hybrider är också tänkbara, t ex att använda fil även för läsning, eller olika replikerande lösningar. Infratjänste-ramavtalet och dess efterförljare (ramavtal Statskontoret / Verva / Kommerskollegium) innehåller även vissa tilläggstjänster som innebär mellanlagring för skrivning/läsning.
- Ifall dokumentation av SQL-datamodellen saknas har kommunen (enligt EU-direktiv 91/250) rätt att göra s k reverse engineering för integrationsändamål för att kartlägga datamodellen.

3.2.3. Screen-scraping

Om inget av ovanstående alternativ visar sig görligt kan en annan reservlösning vara att använda ett bra s k screen-scraping-verktyg för att skapa adaptern mot verksamhetssystemet. Ett sådan verkyg ”inkräktar” inte på applikationens programmering men återanvänder ändå hela verksamhets- och valideringslogiken i den existerande applikationen. Även denna variant är tänkbar för verksamhetsapplikationer som Sofia.

Screen-scraping använder skärmbilderna i applikationens vanliga användargränssnitt. Användargränssnittet körs dolt i en server. Programvaran simulerar att vara användare, den registrerar det som syns på skärmen och kan mata in värden samt utföra mus-klick. I andra änden implementeras Nyttomeddelanden på vanligt sätt.



Figur 10

Kommentarer:

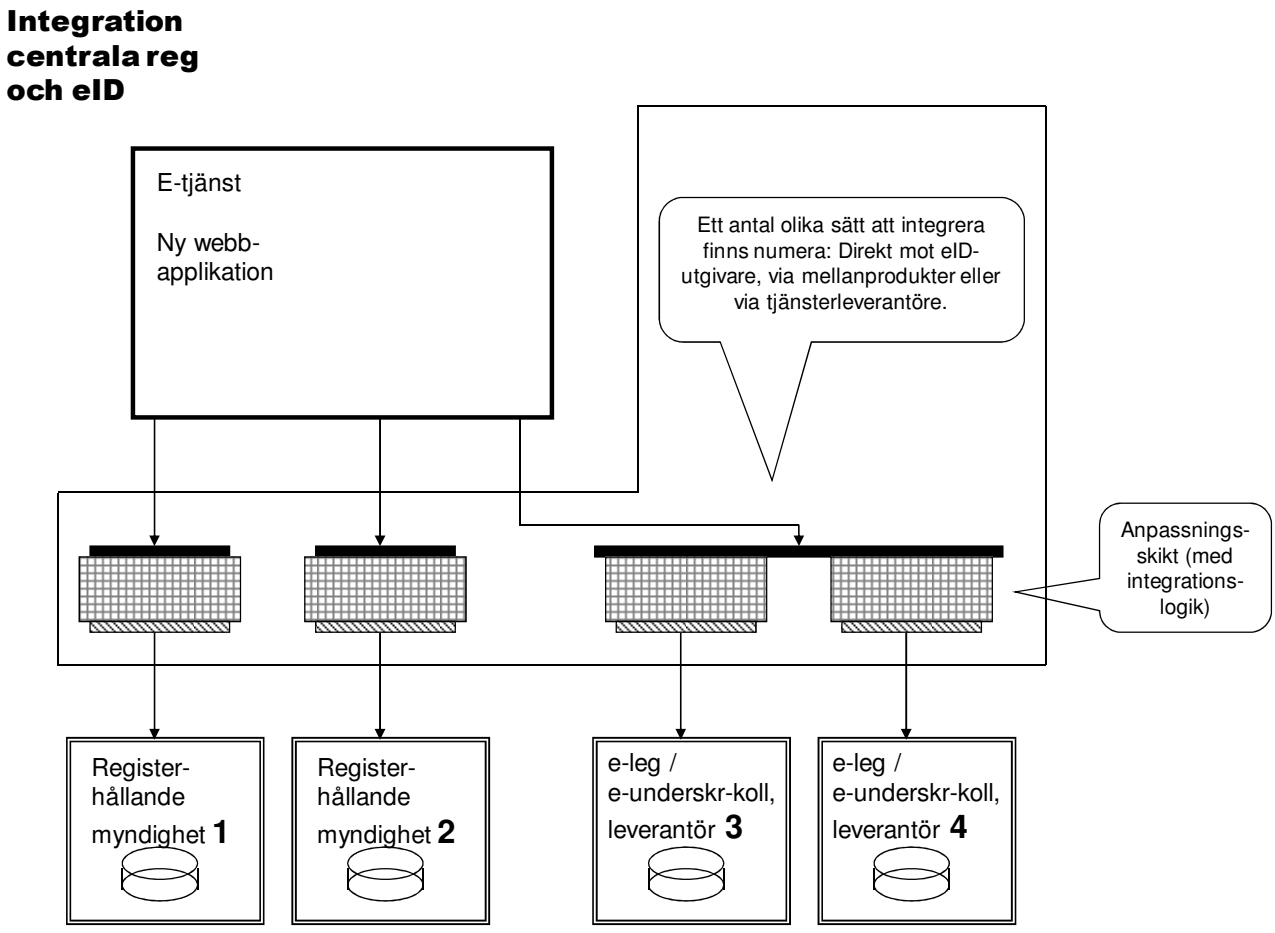
- Screen-scraping-verktyg kom att användas ganska mycket i samband med integration med slutna stordatorsystem, men även mot client/server-system
- På senare tid har utvecklats ett antal screen-scraping-verktyg för webb-användning, s k webb-scraping, dessa är ofta modernare.
- En nackdel är beroendet av uppgraderingar av verksamhets-applikationerna (ifall användargränssnittet ändrats kan screen-scraping-logiken behöva modifieras), se

motsvarande problem för alternativ Z ovan. Man får också risker kring ovanliga/tillfälliga användarmeddelanden.

- Prestandaaspekter kan behövas beaktas så att inte en stor mängd skärmar i användargränssnittet behöver genomlöpas bara för att svara på ett enda Nyttomeddelande t ex
- Man får vanligen viss inlåsning i screen-scraping-verktyget och det har en kostnad.

3.3. Olika behov av anpassningslogik mot centrala register, eID mm

Jämfört med ovan nämnda anpassningslogik mellan e-tjänstens webbapplikation och en verksamhetsapplikation så får anpassningslogiken mot andra myndigheters centrala register, kontroll av e-legitimation/e-underskrift (eID) mm vissa andra egenskaper och dessutom en ännu större återanvändningspotential.



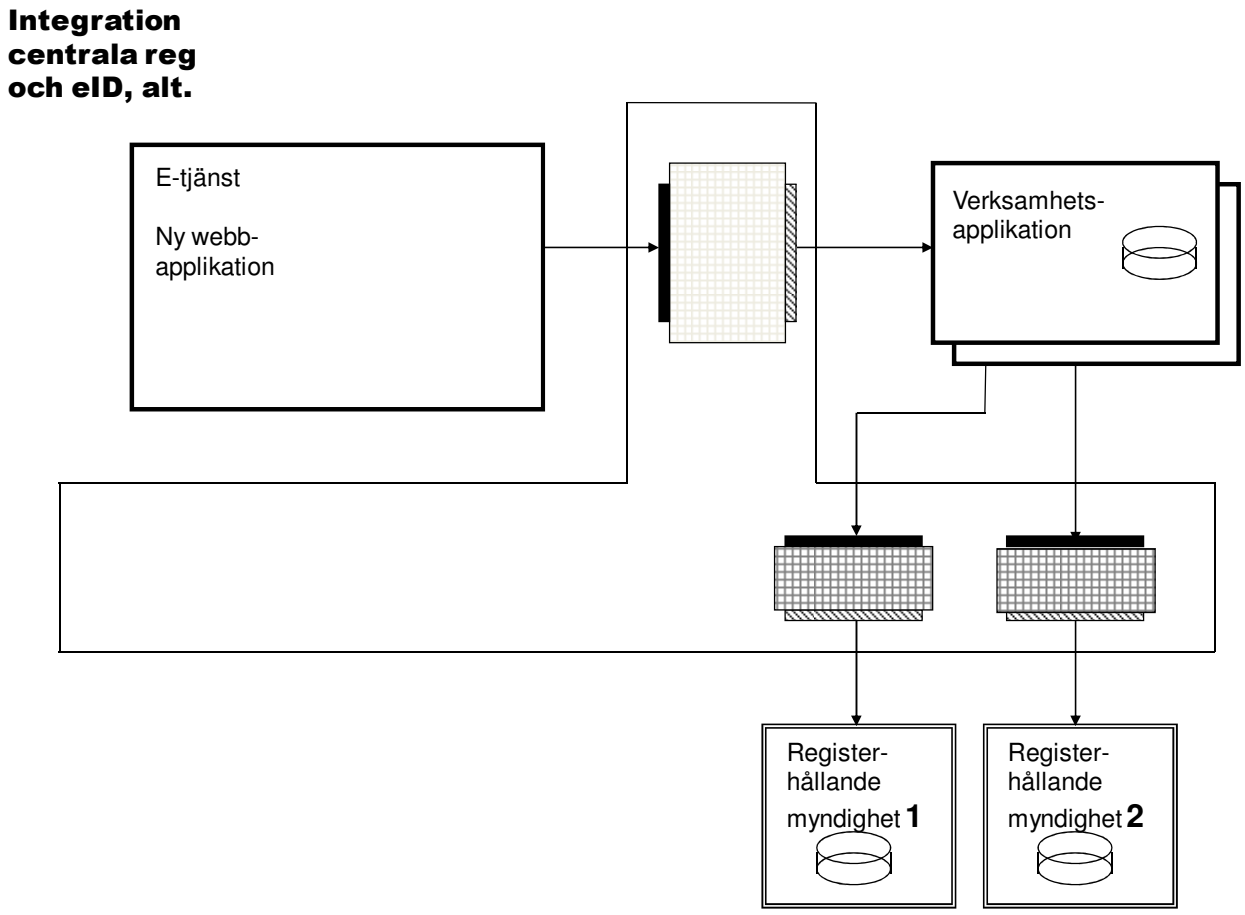
Figur 11

Kommentarer:

- Som exempel på ”Registerhållande myndighet 1” tas någon myndighet som exponerar ett SHS-gränssnitt gentemot sin information. Det innebär att ”snedstreckat skikt” i figuren ovan ofta är ett anropssnitt till Infratjänstens SHS-funktioner, t ex ett ”rörledningsanrop” såsom SHSSendSync. Nyttodatat som transporteras av detta SHSSendSync-anrop bör definieras av ett Standardmeddelande/Nyttomeddelande. ”Rutigt skikt” i figuren översätter och formatkonverterar mellan dessa snitt. Se även det separata avsnittet *Principer för Nyttomeddelanden*.

- ”Svart skikt” bör däremot på samma sätt som gentemot verksamhetsapplikationer vara helt på ”verksamhetsobjekt”-nivå, relativt e-tjänsten ifråga.
- Andra kommunikationsvägar är tänkbara, beroende på en myndighets kapabilitet, t ex skulle ”Registerhållande myndighet 2” kunna nås via Web Services eller annat anropssätt (”snedstreckat skikt”). ”Rutigt skikt” i figuren översätter och formatkonverterar mellan dessa snitt. Skulle ”snedstreckat” och ”svart skikt” vara identiskt definierade kan naturligtvis det ”rutiga” skiktet gynnsamt helt utgå på samma sätt som för anpassningslogik till verksamhetsapplikationer.
- Stora asynkrona flöden (t ex av fil- eller EAI-typ) täcks ej specifikt här. Se vidare i separat kapitel.
- Även mindre, asynkrona meddelanden som ej kan leveraras via Standardmeddelanden, SHS, Web Services etc får lösas med traditionella metoder. T ex kan det i tidiga skeden bli så att registerhållande myndighet inte är färdig med modernare integrationsmodeller, varvid man kanske får skapa och överföra en ”frågefil” som ett tag senare genererar en ”svarsfil” från den centrala myndigheten.
- Med eID (dvs e-legitimation/e-underskrift) menas i detta dokument lösningar enligt Statskontorets/Vervas ramavtal för elektronisk identifiering.
- De olika leverantörerna för e-leg/e-underskrift har numera ett gemensamt API på lägre, mer detaljerad teknisk nivå som heter OSIF. Emellertid kan det förekomma skillnader i logik på högre nivå vilket motiverar skissen med flera tänkbara implementationer varvid det behövs integrationslogik som erbjuder ett gemensamt anropssnitt i görligaste mån gentemot e-tjänstens webbapplikation. Detta gäller även om man vill byta Infratjänsteleverantör. Flera leverantörer erbjuder också tjänster för att kommunen ska slippa egen integration med eID-utgivarna. Olika s.k. federeringslösningar finns även vilka bl.a. kan leverera gemensam inloggningssession mellan flera webb-applikationer. Området kommer för övrigt att ändras i och med kommande eID-förändringar från E-delegationen.
- Man bör också nämna att det även krävs viss integration för respektive eID-lösning direkt i webb-applikationen för en e-tjänst (t ex att ställa frågan om vilken eID-lösning medborgaren vill använda, att lägga upp och referera specifik banks komponent etc).
- Det finns även nyare lösningar för federerad autenticering som baseras på standarden SAML2. En del leverantörer kan erbjuda sig att då vara s k Identity Provider.
- I vissa fall behövs inte så hög säkerhet som eID ger (se även säkerhetsutredningen Saekklassn_v10_050120.pdf tillgänglig på www.sambruk.se) eller att parallella mekanismer såsom SMS-engångskod kan användas, varvid också arkitekturen med flera möjliga implementationer bakom samma Nyttomeddelande ger nytta. Det omvända kan också tänkas i ovanliga fall, att ytterligare starkare säkerhet än eID behövs - integreras på liknande sätt. Inom Sambruksutredningen Sammanhållen ärendehantering (se www.sambruk.se) finns också ett Nyttomeddelande föreslaget där varierande inloggningskvalitet relateras till olika informationssäkerhetsbehov.

Se även kapitlet *Säkerhet*.



Figur 12

Ovan visas ett mallalternativ vad gäller integration med centrala myndigheter (eller andra externa parter). Här är det själva verksamhetsapplikationen som kommunicerar med myndigheterna. I vissa fall finns denna funktionalitet redan inbyggd i verksamhetsapplikationen, i andra fall måste den tillföras.

Kommunikationen kan vara av onlinetypen eller istället byggt på någon aviserande, händelsestyrd princip. Vissa mellanprodukter finns idag att köpa för att underlätta detta alternativ.

Adapterproblematik liknande det som beskrivs i alternativen ovan kan självklart förekomma.

3.4. Generella applikationsaspekter

En kommun har självklart anledning att ställa krav på olika lösningar vad gäller driftegenskaper etc. Olika kommuner har gjort olika strategiska val (såsom Microsoft versus Novell) varför kraven ibland är tämligen olika.

En mycket intressant aspekt är ifall driften av en lösning önskas ske i kommunens egna regi eller ifall kommunens drift helt eller till dels läggs ut som traditionell outsourcing.

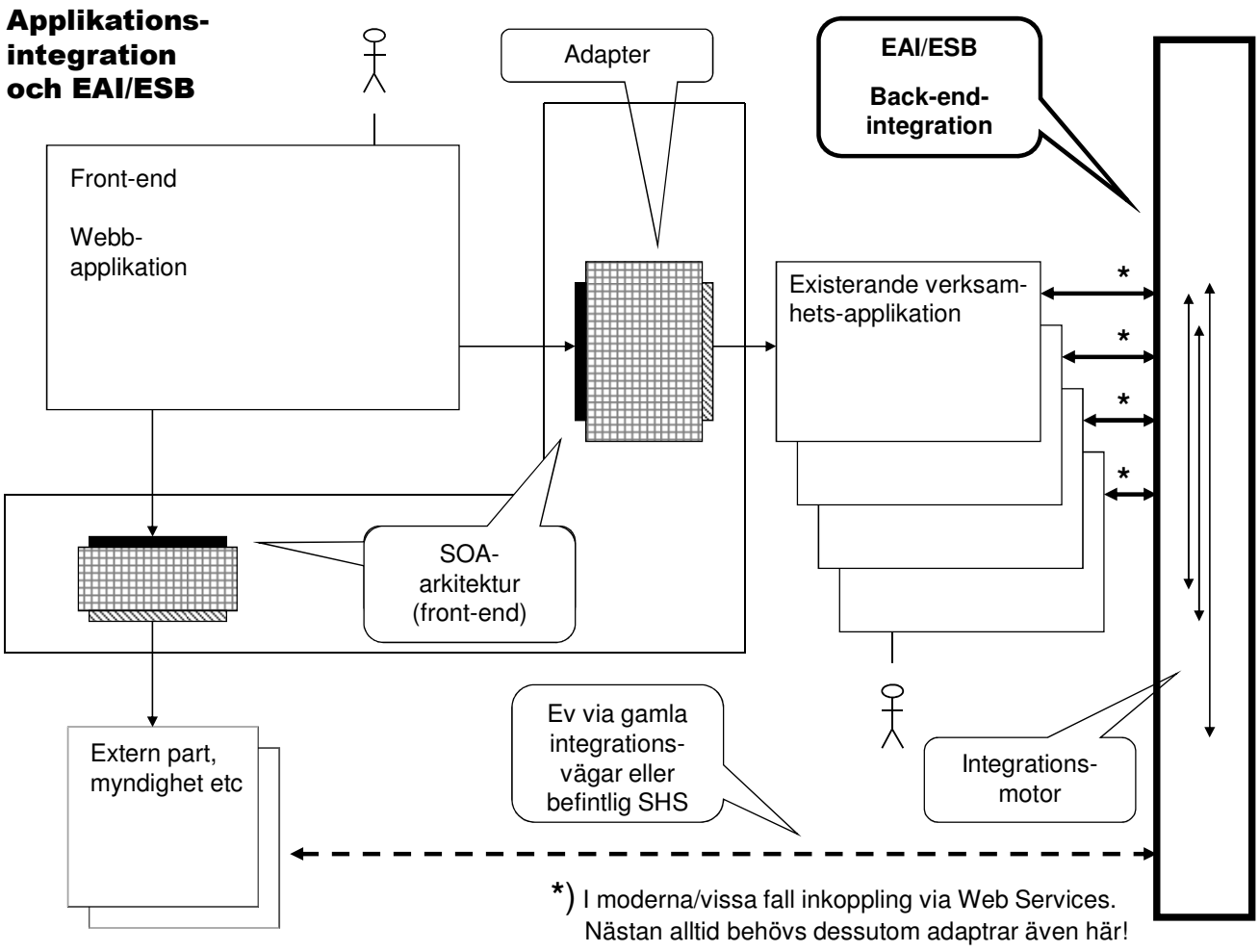
En variant av outsourcing har kommit att bli populär de senaste tio åren, där en specifik lösnings licenskostnader och driftning bakas ihop i ett pris. Detta har kommit att kallas ASP (Applications Service Provider), SaaS (Software as a Service) eller helt enkelt att köpa en viss lösning ”som tjänst”.

SaaS har dessutom på senare tid kommit att klumpas ihop med andra sätt att köpa element i en driftlösning, s.k. cloud computing (”molnet”). Detta är oftast till ett rörligt, löpande pris och med stor möjlighet att öka/minska leveransvolymen över tiden för att kunna anpassa sig till belastningstoppar. Man bör observera att de olika molnerbjudandena är synnerligen olika sinsemellan.

3.5. Applikationsintegration EAI, ESB

Tidiga utgåvor av ÖTP tog inte med detaljer kring avancerad applikationsintegration med EAI (Enterprise Application Integration) eller ESB (Enterprise Service Bus) utan hade endast ett övergripande kapitel. I ett separat Sambruksprojekt under 2006 skapades dock en kravskrift kring Generell integration vars innehåll inkluderades i ÖTP2.0.

Se även det inledande avsnittet *Översikt referensarkitektur applikationsintegration generellt*. Här följer en mer detaljerad översikt:



Figur 13

Ofta kallar man den användargränssnitts-nära integrationen för front-end och den bakomliggande för back-end. EAI har mest kommit att användas för back-end, ESB för både och. Man får räkna med att back-end ofta mera har karaktären batch/asynkront och front-end mera karaktären online/synkront (ofta räcker SOA som integrationsteknik för front-end).

Inom området finns en speciell samklang med SOA, förutom fokus på funktionsintegrationen i SOA-kapitlet ovan så kan även Web Services (SOA-influerat) reduceras till att bli en kommunikationsteknik där EAI-plattformen ska koppla ihop sig med omvärlden (dvs för EAI-pilarna som stjärnmärkts i figuren ovan).

Man kan notera att SHS-specifikationen (se *www.kammarkollegiet.se*) ger vissa goda ESB-egenskaper (en delmängd av typisk ESB-funktionalitet). Vissa av de aktuella SHS-produkterna implementerar mer än specifikationen kräver och tillför därmed vissa ytterligare ESB-egenskaper.

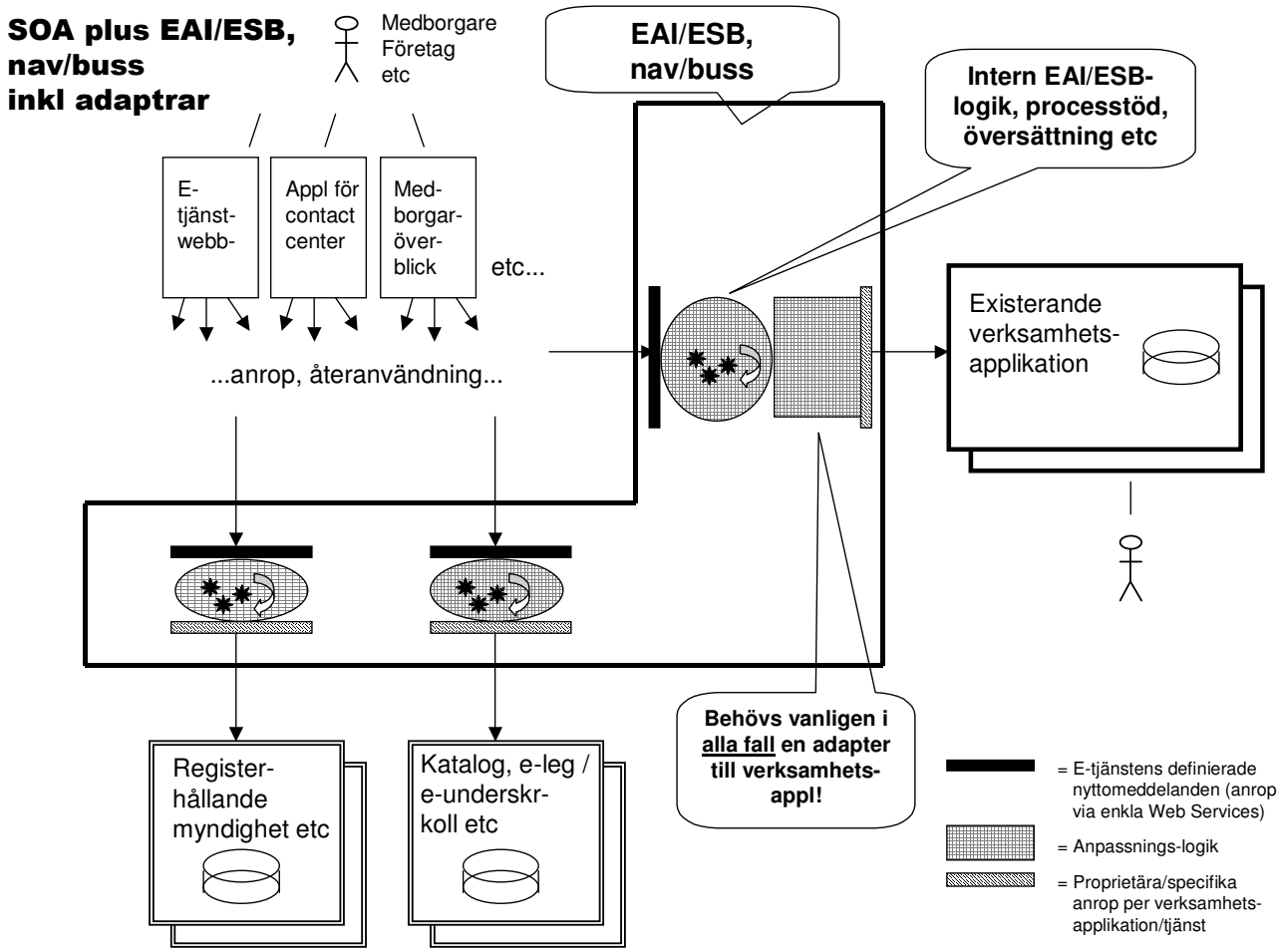
Se i övrigt bl a avsnitten *Styrlogik* samt *Synkront/asynkront* nedan.

3.6. SOA understött av EAI/ESB

SOA och Web Services räcker ofta till som integrationsplattform i sig, framförallt när det handlar om funktionsintegration nära ett användargränssnitt - front-end-integration. Det vill säga, i stort sett alla moderna programspråk och exekveringsmiljöer har redan inbyggt stöd för att anropa eller svara på Web Services. Denna teknik bygger i grunden på webbserverteknik och bör förväntas ha mycket god skalbarhet och feltolerans samt viss loggning i sig.

Vad som dock torde hända efter ett tags lyckosam lansering av många SOA-tjänster är att de blir svåra att hålla ordning på - det uppstår "inter application spaghetti" för SOA. Vid en viss tidpunkt är det därmed troligt att man vill införa någon lösning som kan öka hanterbarhet, övervakning, förvaltningsbarhet mm, en slags SOA-nav eller SOA-buss. Här kan en EAI/ESB-lösning vara helt rätt ansats, förutsatt att vald EAI/ESB-plattform har den exekveringssnabbhet (framförallt latenstid) som behövs ifall SOA-användningen är online. En annan, enklare lösning är att tillföra produkter för s.k. Web Services Management, SOA Governance etc.

Det bör samtidigt poängteras att man vanligen *inte* behöver tillföra en SOA-nav/buss redan från början när antalet tjänster är få. Tack vare SOA:s betoning på meddelandekontrakt och Web Services som teknik torde det gå utmärkt att "infoga" en nav/buss först vid den tidpunkt som den verkligen tillför något. Då slipper man onödig komplexitet i tidiga skeden.



Figur 14

Det bör noteras att även i detta fall så behövs inkopplingspunkterna till standard-applikationerna via adaptrar (eller i bästa fall öppna maskingränssnitt) - EAI/ESB-produkten tillför inte så mycket härvidlag för en typisk kommun.

Svårigheten att förverkliga välfungerande adapters och klara en långsiktig förvaltning av dessa kan i många fall tyvärr helt underminera fördelarna med integrationsprodukter. Det kan istället vara så att någon leverantör kan erbjuda förvaltning/support av vissa adaptrar i samband med att de erbjuder en viss integrationsmotor. Ett sådant erbjudande kan alltså vara mycket värdefullt och det är full tänkbart att värdet på sådana adaptrar överskuggar betydelsen av själva integrationsmotorn ifråga.

Man kan vilja bevaka att en ev adapter inte blir "inlåst" eller beroende av en specifik EAI/ESB-plattform, den bör hellre ha ett interoperabelt anropsgränssnitt i sig (vanligen Web Services).

3.7. Processtöd

3.7.1. Generellt om process-aspekten

(Se även kapitlet Processtöd med hjälp av workflow.)

Detta kapitel är tänkt att beskriva några olika fall vad gäller var verksamhetsprocess-stödet egentligen hamnar.

Om man skulle utgå från ett helt vitt papper så skulle man analysera verksamhetens önskade processtöd från IT-applikationer fullständigt förutsättningslöst och dessutom fullständigt oberoende av vilka verksamhetsapplikationer man har eller som finns att tillgå, liksom dessas teknikmiljöer. Därefter skulle man mappa denna önskebild mot ”verksamhetsobjekt” som man tänker sig finns i verksamhetsapplikationer respektive som skulle behöva nyskrivas i en e-tjänste-applikation. På grund av många orsaker (tidsramar, kostnadsramar, att tillgängliga verksamhetsapplikationer inte är modulariserade i tillräcklig grad, risk för att bli över-teoretisk och aldrig komma i mål, etc) har vi hittills inte valt denna mycket höga ambitionsnivå. Istället har vi utrett önskat processtöd samtidigt som vi sneglat på det processtöd som redan existerar i form av rutiner och stöd via verksamhetsapplikationer. Därmed hopppas vi uppnå en pragmatisk optimering av genomförbarhet och god verkshöjd.

Vad gäller applikationsarkikturen så skulle en tänkt mycket hög teoretisk ambitionsnivå också ha krävt tekniklösningar som i praktiken inte är trovärdiga för Sambruk, bl a vad gäller kostnad, komplexitet och inlåsningsrisk. Se vidare kapitlet *Rörläggning*.

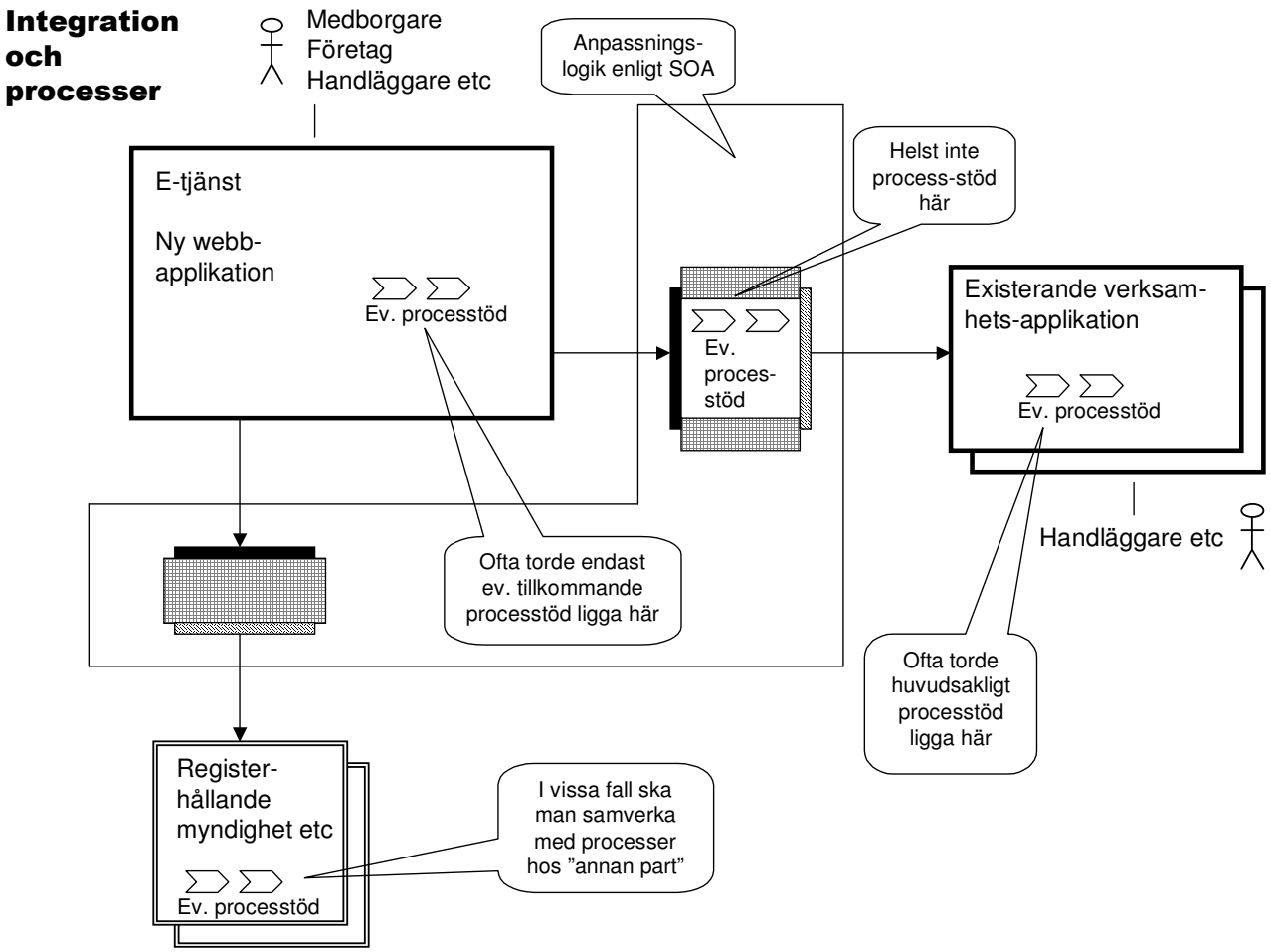
Tanken är alltså att försöka återanvända så mycket processtöd som möjligt ur verksamhets-applikationerna och sedan tillföra det som eventuellt saknas (eller inte passar) i e-tjänstens webbapplikation (eller möjligen i anpassningsskiktet).

Det blir här viktigt att vara säker på vad som är ”master” för processtödet och att göra arbetsrutiner tydliga för handläggarna om det i något fall skulle bli överlappning mellan processtöds-ambitioner i verksamhetsapplikationer och e-tjänst. Detta är ju för övrigt ett normalt kompromissarbete i samband med större integrationsprojekt i IT-världen.

Rent praktiskt kan det alltså tänkas att handläggarna huvudsakligen arbetar direkt gentemot sitt normala verksamhetssystem, men att vissa e-tjänst-relaterade åtgärder får utföras genom att t ex gå in i e-tjänstens webbapplikation (under högre behörighet i rollen som handläggare).

I andra fall är det naturligtvis tänkbart att man väljer att inte använda ett existerande verksamhetssystem alls utan låter hela processtödet både för medborgare och handläggare finnas i en e-tjänst-applikation.

För det tidiga exemplet Borlängepiloten inom Sambruk blev fallet ganska enkelt, hela processtödet för handläggarna tänks ligga i verksamhetsapplikationen (dvs idag Sofia).



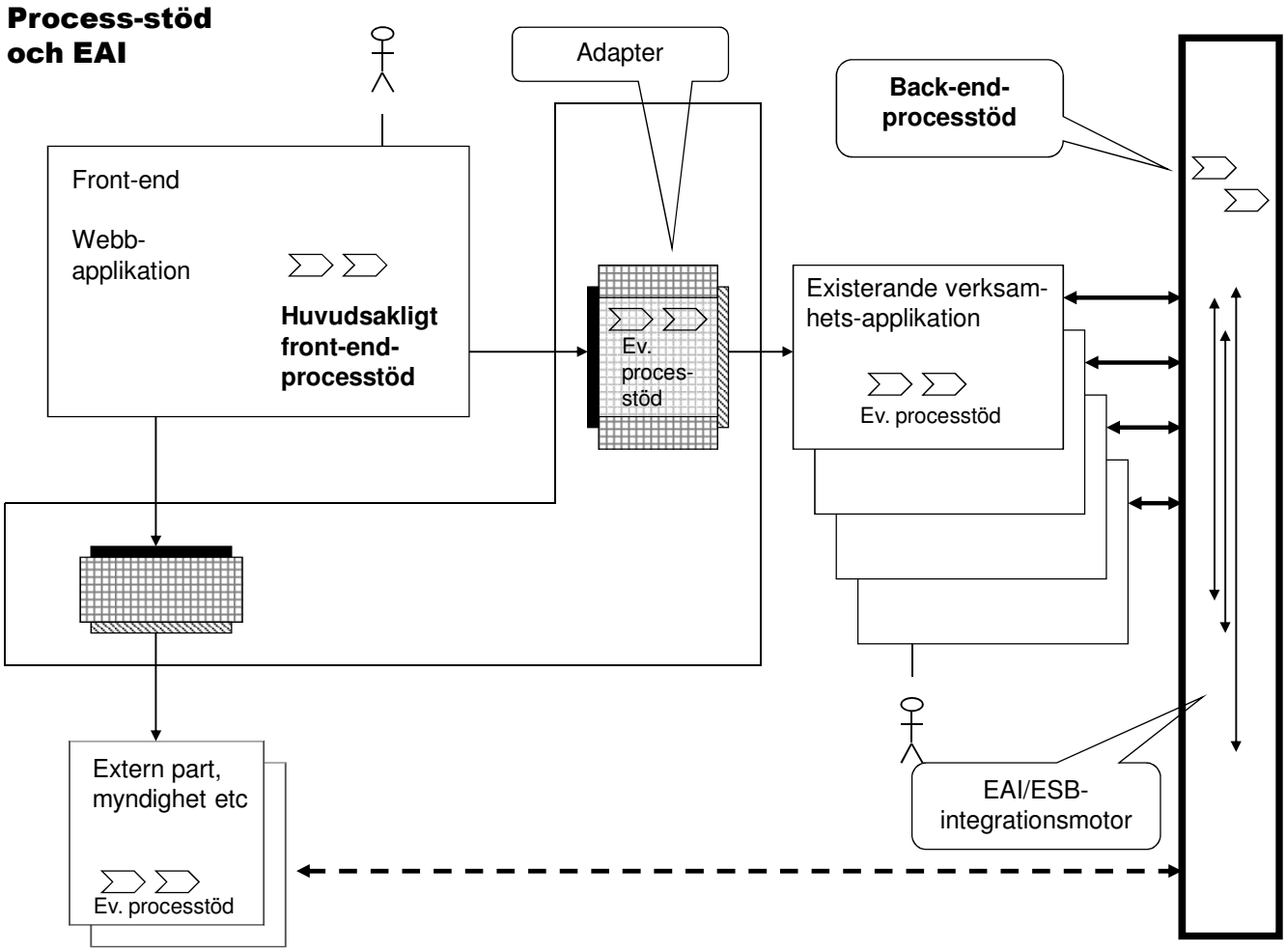
Figur 15

Kommentarer:

- Även inom anpassningslogiken skulle det kunna vara tänkbart att processtöd införs (t ex i fallet att en existerande e-tjänst i ett senare skede ska integreras med en mycket mager verksamhetsapplikation, då måste anpassningsskiktet bli extra tjockt och t o m i ogynnsamma fall behöva innehålla viss processstödslogik).
- Denna version av ÖTP har inget specifikt mönster för fallet att en kommun själv vill utgöra ”annan part” åt någon aktör enligt den nedersta textbubblan i figuren, det får i så fall lösas på traditionella sätt. Rent konkret kan det i detta fall t ex behövas en större SHS-implementation ifall synkrona SHS-anrop ska kunna betjänas hos kommunen.
- Processtöd kan också ligga i en EAI/ESB-plattform, se separat kapitel.

3.7.2. Processtöd med hjälp av EAI/ESB

De flesta EAI-plattformarna innehåller processtöd (BPM - Business Process Management mfl namn). Här menas dels möjligheter att styra en verksamhetsprocess med hjälp av regelverk (workflow) i EAI-lösningen, dels att övervaka att processen fungerar som den ska, få varningar, nyckeltal, avstämningsstatistik etc (s k BAM).



Figur 16

I denna specifikation fokuserar vi på processtöd back-end (som naturligtvis indirekt kan tjäna även användargränssnittsnära workflow) eftersom front-end-lösningarna vanligen ligger inom andra tekniska plattformar. Icke desto mindre är det förstås viktigt att front-end- och back-end-processtöd kan agera i samklang och inte har för stora överlapp.

Se i övrigt bl a avsnittet *Styrlogik* nedan.

3.7.3. Styrlogik inom EAI/ESB

Med *styrlogik* nedan menas definitioner för formatöversättning, processtyrning, s k state-hantering, routing baserat på meddelandeinnehåll/typ, styrning av en-till-många respektive många-till-en, adaptrar, insticksmoduler, driftmiljöparametrar etc.

3.7.4. Processtöd med hjälp av några enklare workflow-funktioner

(Se även kapitlet *Process-aspekten*.)

Workflow tänks utredas noggrannare längre fram i tiden, men här inkluderas ändå några specifika funktioner som det framkommit behov av under ÖTP-arbetet (samt vid utredningsarbete hos vissa kommuner). Observera också att det separata Sambruksprojektet Dokument och ärendehantering har måst göra tydliga avgränsningar eftersom deras kravmassa varit mycket stor. De fokuserar framförallt på nämndadministration och inte på ärendeflödeshantering/workflow i sig.

De andra Sambrukspiloterna som funnits har framförallt sett behov av följande tre ”workflow-nyttigheter” idag:

- Rollbaserad **inkorg** till handläggargrupp för ”dispatch” av inkomna ärenden från e-tjänsten (i det fall inkorg inte sköts av verksamhetssystemet). Notera att själva ärendelagringen i sig dock först normalt har skett direkt via e-tjänsten mot verksamhetssystemet.
- Kunna ge **medborgaröverblick** över dennes ärenden hos kommunen – Mina Engagemang. Option: Att föda liknande info till Infratjänstens Mina Sidor
- Hantera “**timeout**” t ex att tidsgräns passerat utan att ärende blivit handlagt eller att medborgaren missat inge komplettering.
  Eller kravrutin – dock, lägg troligen hellre denna funktion där den hör hemma, i kundreskontran i ekonomisystemet

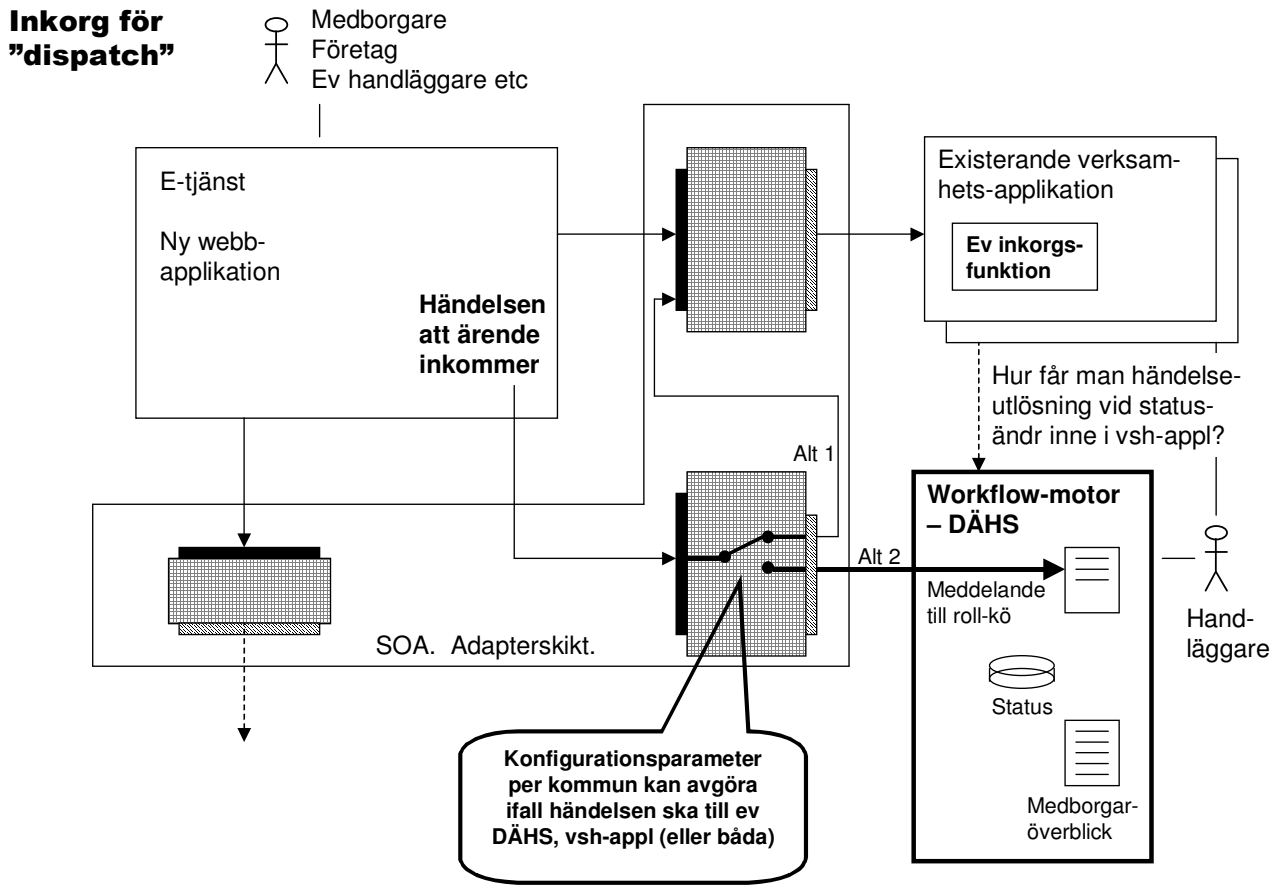
För kommuner som inte vill investera i ett helt dokument/ärende/workflowsystem skulle man kunna tänka sig att anordna endast ovanstående tre simpla funktioner, t ex att be om offerter för ”nerminskade” workflow-produkter, att använda någon Open Source-variant eller eventuellt att utveckla. En ännu enklare interimslösning (för endast inkorg) inkluderas även sist i kapitlet.

Se även avsnittet om *Ärendeknutpunkt* nedan.

Nyttomeddelandena ska förhoppningsvis vara så generella att webb-applikationen för e-tjänsten ska kunna bli oberoende av senare val av workflowlösning etc.

3.7.4.1. Enkel inkorg

Nedan följer en mallarkitektur för en enkel inkorg:



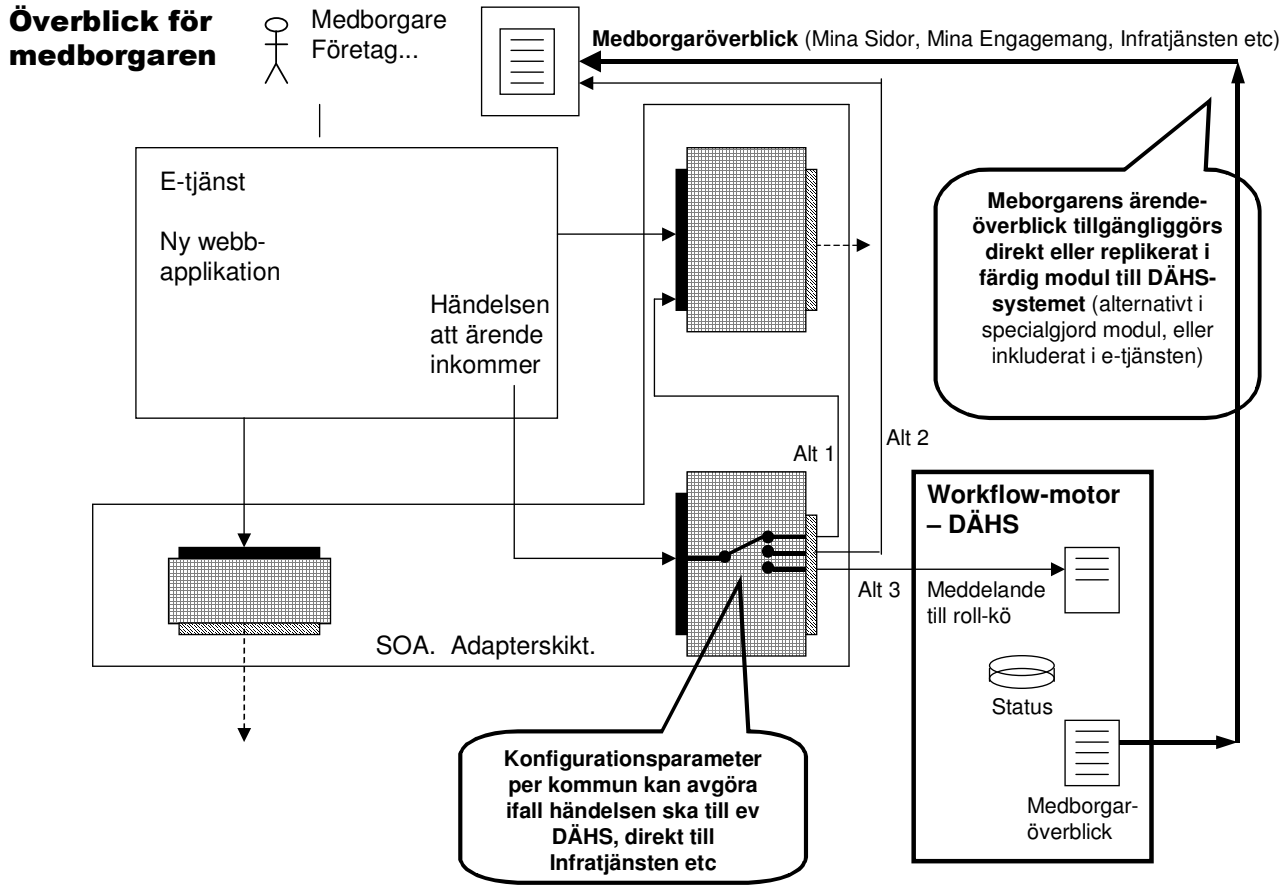
Figur 17

Kommentar vad gäller att få ”händelseutlösning” från existerande verksamhetsapplikationer, några lösningsalternativ:

- En mycket modern applikation kan tänkas ha ”krokar” för sådana händelser (eller sådana kan tillföras på beställning)
- Liknande teknik som används för metakatalog-snapshot/replikering kunde tänkas
- Man tillför SQL-triggers i databasen (vars övriga definitioner då kan lämnas helt oförändrade)
- Ofta har applikationerna exportmöjligheter till flata filer. Med dessa som utgångspunkt kunde skillnads-logik köras för att få fram vilka statusändringar som skett sedan förra körningen.

3.7.4.2. Medborgaröverblick, enkel Mina Sidor

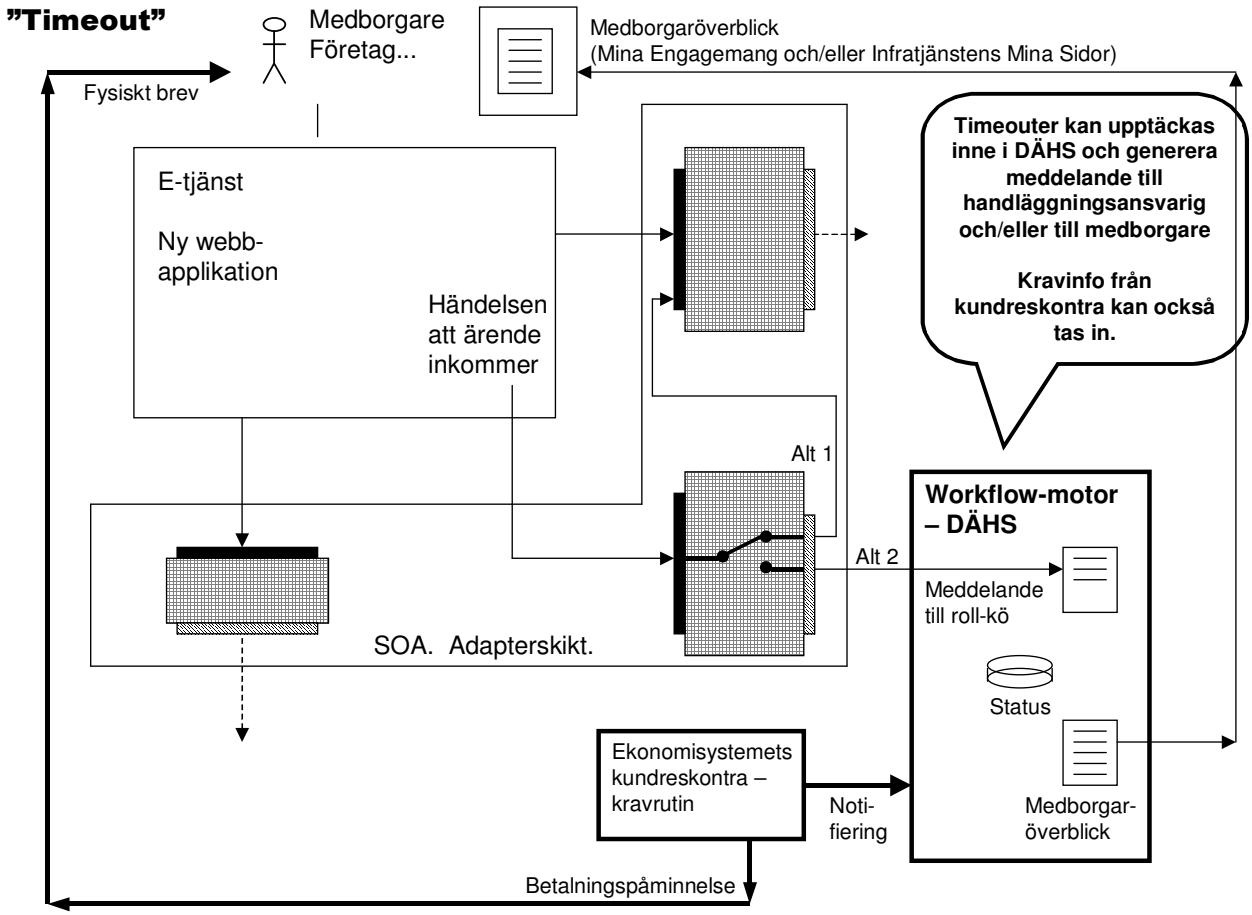
Nedan en mallarkitektur för medborgaröverblick:



Figur 18

3.7.4.3. Enkel ”timeout”

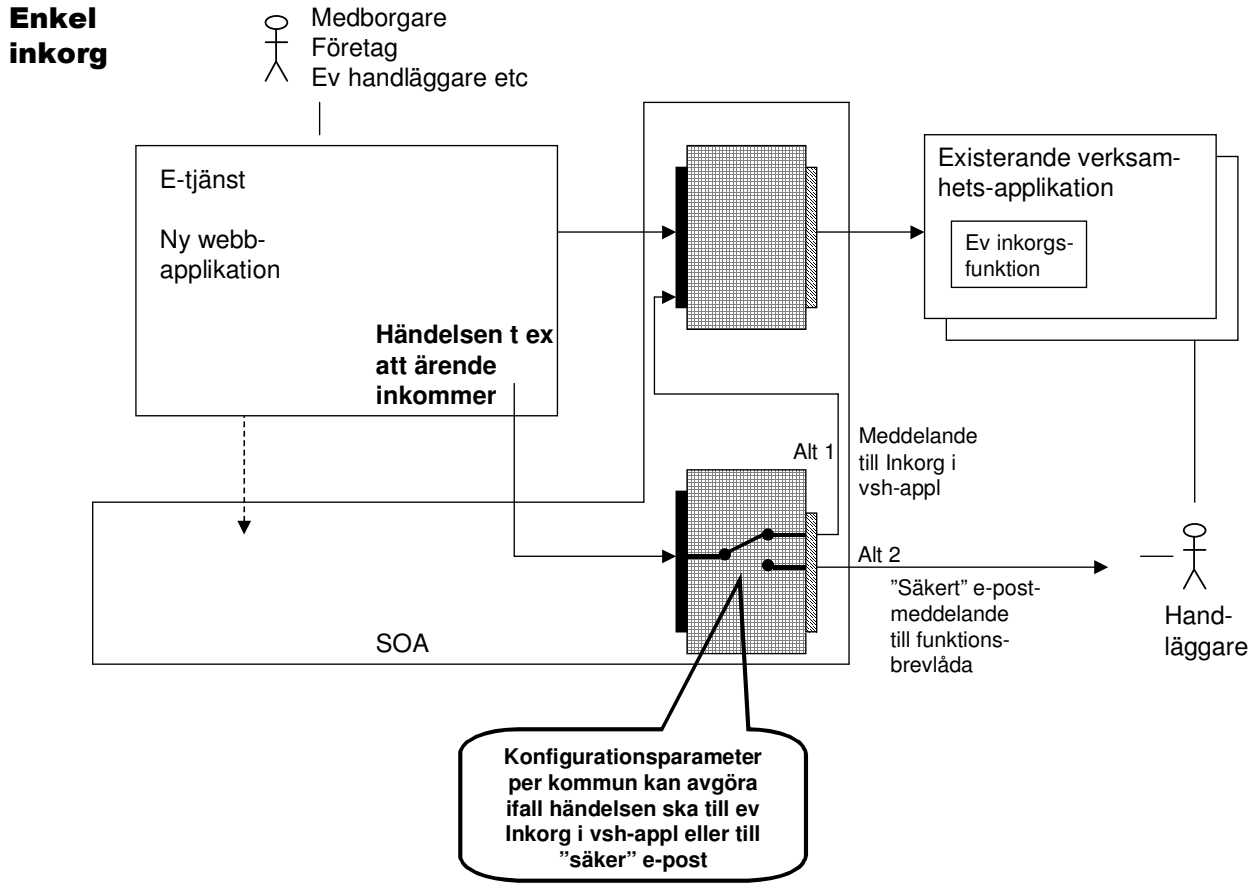
Och nedan en mallarkitektur för tidsutlösning (s k timeout):



Figur 19

3.7.4.4. Extraenkel, alternativ inkorg

Här inkluderas även en alternativ, synnerligen enkel mallarkitektur för endast inkorg:



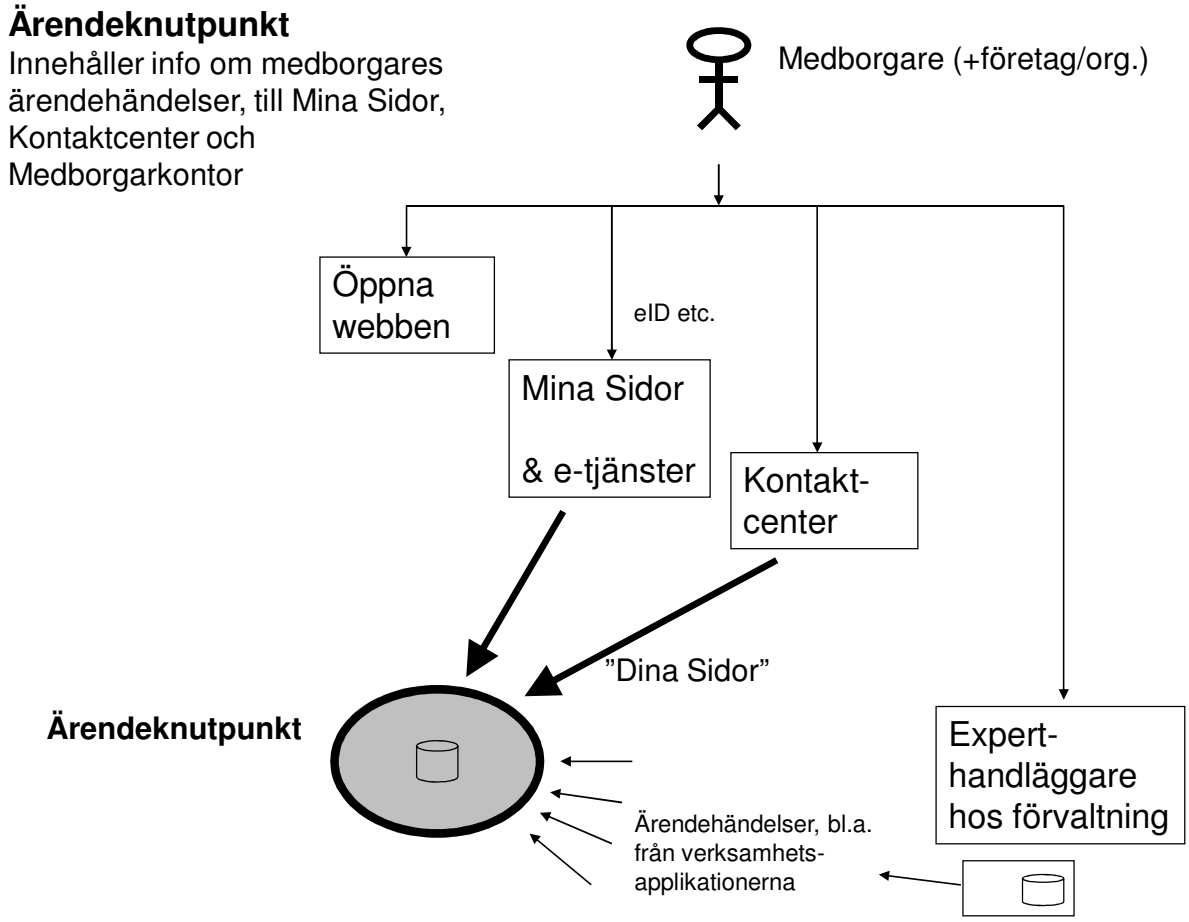
Figur 20

Kommentar: Med ”säker” e-post menas här att meddelandena aldrig färdats i osäkra nätzoner såsom Internet utan skickas direkt till funktionsbrevlådan, helt inom kommunens interna e-post-system.

3.8. Ärendeknutpunkt

Tre specifika företeelser leder till att en s k Ärendeknutpunkt skulle göra stor nytta:

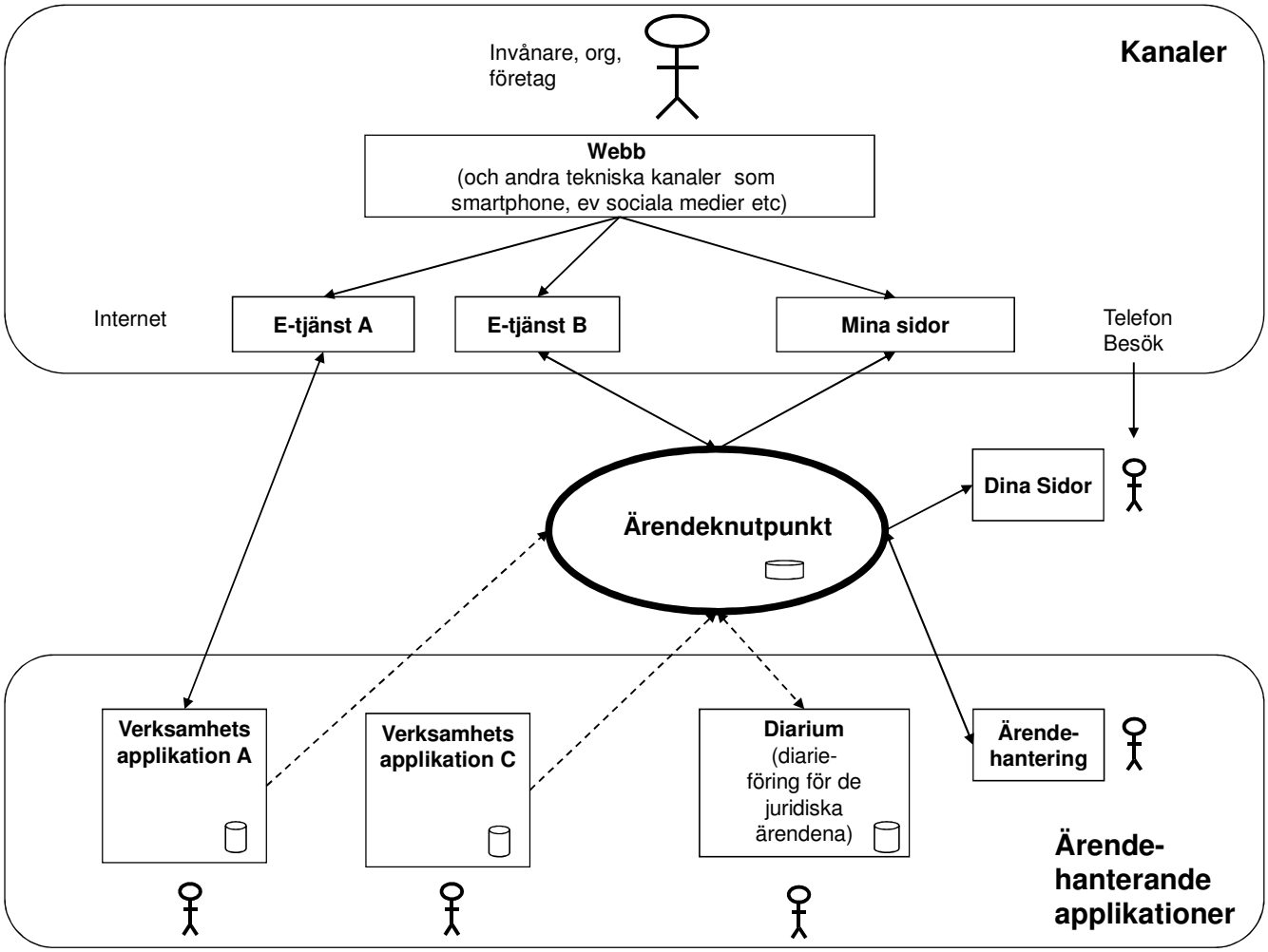
- Konktaktcenter
- Medborgarkontor
- Mina Sidor (för medborgare och även för företag, förening)



Figur 21

Ärendeknutpunkten samlar ihop information per berörd medborgare och även per ärende. Typiskt så hanteras ett bredare ärendebegrepp än bara de traditionella juridiska ärendena, här kan skötas även ren servicegivning, bibliotek, sport och simhall etc. Informationsklassning blir alltså mycket viktig för att veta vilka krav som ska ställas i Ärendeknutpunkten på vem som får se vilken information. Ramverk från Datainspektionen och olika lagar blir viktiga i samband med denna informationsklassning. Dock ska behörighetsregler kunna sättas så att i vissa sammanhang endast berörd medborgare kan se informationen på Mina Sidor, inte personal i kontaktcenter/medborgarkontor. Därmed blir funktionen till en ren service för medborgaren och lagrad information kan tänkas inte vara allmän handling i det skedet (här finns förebilder från Bolagsverket/Skatteverket mm).

Översikten för denna mallarkitektur blir:



Figur 22 Ärendeknutpunkt

I figuren förekommer två olika slags integrationer (markerade med pilar)

- Heldragna pilar: SOA-anrop (eller möjligen asynkron överföring)
- Streckade pilar: Händelseöverföring

vilka gås igenom i följande avsnitt.

Händelseöverföring har syftet att ge god överblick över hur ärenden fortskrider. Det behövs därför tätare/oftare händelseöverföring än vad som i de flesta fall sker idag (framförallt då idag ärendestatus ofta endast skickas till formellt diarium efter ärendeavslut). Orsaken är främst att kunna ge invånare, organisationer och företag god återkoppling om ärendehandläggning. Detta ökar upplevd servicegrad och minskar lasten av telefonsamtal till handläggare (och även till Kontaktcenter).

I hög grad är god tillgång till händelseöverföring en av de viktiga framgångsfaktorerna för e-förvaltning för medborgarna. Om det inte finns händelser att titta på i Mina Sidor för att få återkoppling så känns det inte så angeläget att alls börja använda e-tjänster (då uppfattas e-tjänsterna bara som ”svarta hål” – saker försvinner in, men inget kommer ut). Som tur är kan införandet av händelsesignallering ske allteftersom det är görligt, det måste inte vara en ”big bang”. Men det måste samtidigt betonas att arbete måste startas skyndsamt med tanke på ledtiderna. Likaså kan detaljeringen av händelsesignallering förbättras successivt. Tidigt kanske man börjar med att endast skicka ärendeslutpunkt i form av beslut (då kan man ofta återanvända äldre, befintliga meddelandeflöden ämnat för diarie). Därefter börjar man signallera även olika delsteg i handläggningen för att ytterligare ge medborgaren känsla av delaktighet och att ”det händer något”.

3.8.1. Ärendeknutpunkt – SOA-anrop

Heldragna pilar i Figur 22 motsvarar SOA-anrop i samma stil som i ÖTP:s huvudarkitektur, se t ex avsnitten 3.1. Tjänsteorienterad arkitektur, SOA och 3.2. Olika behov av anpassningslogik mot verksamhetsapplikation.

Ett exempel: En skolvalsapplikation (e-tjänst A i bilden) skulle kunna vara köpt tillsammans med en intern lösningför skoladministrering (verksamhetsapplikation A i bilden). Dessa två delar kan då tänkas ha ett proprietärt maskingränssnitt mellan sig. Det är dock naturligtvis önskvärt ifall gränssnittet är publicerat och öppet anropbart från annan part under korrekta säkerhetsbetingelser, då skulle det följa ÖTP:s SOA-mall. Maskingränssnittet kan även tänkas vara asynkront.

E-tjänst B i bilden tänks istället vara en enkel e-blankett som lagrar in ett ärende direkt i Ärendeknutpunkten via SOA-anrop (eller möjligen asynkront). Oavsett vilket bör det följa specifikationen för Nyttomeddelanden.

Mina Sidor/Dina Sidor anropar troligen Ärendeknutpunkt direkt för att se ärendestatus (anrop enligt ÖTP:s SOA-mall), liksom då ärendehantering sker direkt mot Ärendeknutpunkt (nere till höger i figuren).

3.8.2. Ärendeknutpunkt – händelseöverföring

Streckade pilare i Figur 22 ovan motsvarar händelseöverföring.

Med detta menas att en viss applikation spontant ska ge ifrån sig ett meddelande då händelsestatus har ändrats inne i den applikationen. För exemplet med skolapplikation A ovan skulle följande sekvens vara typisk:

1. Ärenden etableras direkt i verksamhetsapplikation A i ett detta alternativ. Årendeetablering kan tänkas ske dels via handläggare direkt i verksamhetsapplikationen, dels av medborgare via e-tjänsten.
2. För att medborgaren ska kunna få överblick över sina olika ärenden hos kommunen via Mina Sidor, liksom att personal i kontaktcenter/medborgarkontor ska kunna se ärendeinformationen vid telefonsamtal eller besök (ifall informationsklassningen så tillåter) så måste ärendehändelser spontant genereras ifrån verksamhetsapplikation A till Ärendeknutpunkten vid ärendeetableringen.

3. Detsamma bör ske när ärendestatus har ändrats på grund av att ärendehandläggningen fortskridit ett steg eller beslut fattats och ärendet avslutats.

Samma mönster gäller för andra verksamhetsapplikationer, oavsett hur ärendet först etablerats.

De blir alltså viktigt att ställa krav på verksamhetsapplikationer inte bara att de är SOA-anropbara, utan också att de kan leverera spontana ärendehändelser i samband med att ärendestatus ändrats inom dem.

Sambruk har specificerat ett generellt Nyttomeddelande för Ärendehändelse. Detta bör prioriteras vid specifikation av lösningar.

(Se även avsnittet

3.7.4.1. Enkel inkorg, för några kommentarer om händelsutlösning.)

Flera sorters händelseöverföring finns:

- I de fall som ett ärenden först har etablerats i Ärendeknutpunkten, men därefter vidareförts till experthandläggning, så kan senare en verksamhetsapplikation hos en förvaltning skicka en ärendenotifiering om att ärendet har uppnått en ny status, att beslut fattats, etc. Då bör denna notifiering innehålla den ärendeidentititet som Ärendeknutpunkten tidigare etablerat.
- I de fall som en verksamhetsapplikation hos en förvaltning först har etablerat ett ärende, så måste notifieringen innehålla den ärendeidentititet som verksamhetsapplikation tidigare etablerat. Annars går inte händelsen att korrelera till rätt ärende vid visning i Mina Sidor.
- Ansatsen är att till huvuddiarium förs endast slutbehandlade ärenden, efter ärendebeslut. I övrigt utgör Ärendeknutpunkten diarium för inkomna handlingar och information om handläggningssteg. Ärendeknutpunkten är enda arkiv för inkomna handlingar och information om handläggningssteg för serviceärenden (dvs icke-formella ärenden). Kommentar: I vissa handläggningssammanhang, t.ex. inom socialsidan, utgör verksamhetssystemet inom förvaltningen enda diarium.
- Ifall ärenden etablerats direkt i huvud-diariet, bör ärendenotifiering skickas till Ärendeknutpunkten.
- Varje ärende ska vid varje tillfälle ha en och endast en ansvarsägare.
- Följande gäller för att klara informationstillgång till Mina Sidor och för att Kontaktcenterpersonal ska kunna svara på frågor om ärendestatus:
  I de fall Kontaktcenter direkt vidareförmedlar ett ärende, dvs förmedlar ett ärende för vidare handläggning inom en förvaltning, bör vi i de flesta fall anse att därefter kan det komma in ärendenotifieringar till Ärendekuntpunkten. Ärendet får alltså inte göras helt inaktivt i Ärendeknutpunkten, utan ska ha en status enligt beskrivningen ”Vidareförmedlat: Ägarskapet överfört men ärendet fortfarande aktivt för att kunna motta ärendenotifieringar”.
- En av de viktiga händelsnotifieringarna är Avslut av ärende. Det är den part som har ärendeägarskapet som skickar en sådan händelse. En sådan notifiering har en specialställning i och med att den ”släcker larm” som kan finnas uppsatta i Ärendeknutpunkten för vid vilken tidpunkt som varning till ansvarig ska skapas ifall ärendehandläggning avstannat pga sjukdom e.dyl. Se mer info i specifikationen för Nyttomeddelande för ärendehändelse.

3.8.2.1. Teknik för händelseöverföring

Ifall översättning ska ske mellan olika meddelandeformat eller olika överföringstekniker så görs detta enligt ÖTP:s normala adapterprincip, se 3.2. Olika behov av anpassningslogik mot verksamhetsapplikation. Se även 3.5. Applikationsintegration EAI, ESB och 3.6. SOA understött av EAI/ESB, liksom 3.13. Notifiering.

Händelseöverföring är i sin grundkaraktär asynkron. En vanlig ansats är då att använda en kölösning. De nämnda ÖTP-kapitlen förespråkar Web Service-gränssnitt (WS) *till* och *från* en kö, då sådan används, enligt följande:

Händelsesändare >
WS-gränssnitt >
Sändningsadapter >
Kö >
WS-gränssnitt >
Mottagningsadapter (ev)
Händelsemottagare

En av alla fördelar blir att detta följer black box-principen, ger isolation och inkapsling samt kan få god utbytbarhet för kö-produkten/lösningen. Vid inkapslingsgränssnittet ska Nyttomeddelanden användas vilket kan kräva ett skikt till i kedjan ovan.

Ett modernt alternativ till traditionell kö-produkt är RSS/Atom och med sådan använd som kö blir kedjan:

Händelsesändare (ofta en verksamhetsapplikation) >
WS-gränssnitt >
Sändningsadapter >
RSS/Atom >
Händelsemottagare

Händelsesändare i form av äldre verksamhetsapplikationer kommer ofta, åtminstone i början, inte att klara att skicka händelsesignaler via Web Services. I vissa fall kan vi återanvända signallering som sker i dag till andra system, ofta till diarium. Denna kan vara filbaserad och/eller hanterad via Decapus/TEIS e.dyl, här får man anpassa sig till situationen och bygga en adapter för varje fall. Så tidigt som möjligt i kedjan bör man dock översätta till det generella Nyttomeddelandet som definierats för ärendehändelser.

Nyttomeddelandet måste dock inte skickas via web service ifall signalleringen vid källan är helt filbaserad. Resonemanget i 5. Principer för Nyttomeddelanden innebär ett oberoende av transportteknik och täcker därför att överföring kan ske via fil. Se även 3.9. Mönster för batchuppdatering. Därmed ger det tillräcklig inkapsling att ha följande kedja för sändning av filinfo till Ärendekuntpunkten:

Händelsesändare (ofta en verksamhetsapplikation) >
Proprietär fil >
Sändningsadapter >
RSS/Atom >
Händelse-mottagare

Sändningsadaptern översätter även till Nyttomeddelande ifall så behövs (detta blir ett vanligt fall, i början).

I något fall skulle det kunna bli så att generering av händelser måste ske genom att periodiskt läsa direkt i en verksamhetsapplikations databas, och jämföra med läget vid förra läsningen s k snapshot-samt-delta-generering (skillnadsinformationen utgör händelserna). Då utgår skiktet Proprietär fil som alltså ersätts av databasaccess.

Observera att händelsesignallering till Ärendeknutpunkt bör kombineras med notifiering direkt till medborgare om att det finns ny info att logga in och se på personens Mina Sidor. Se 3.13. Notifiering.

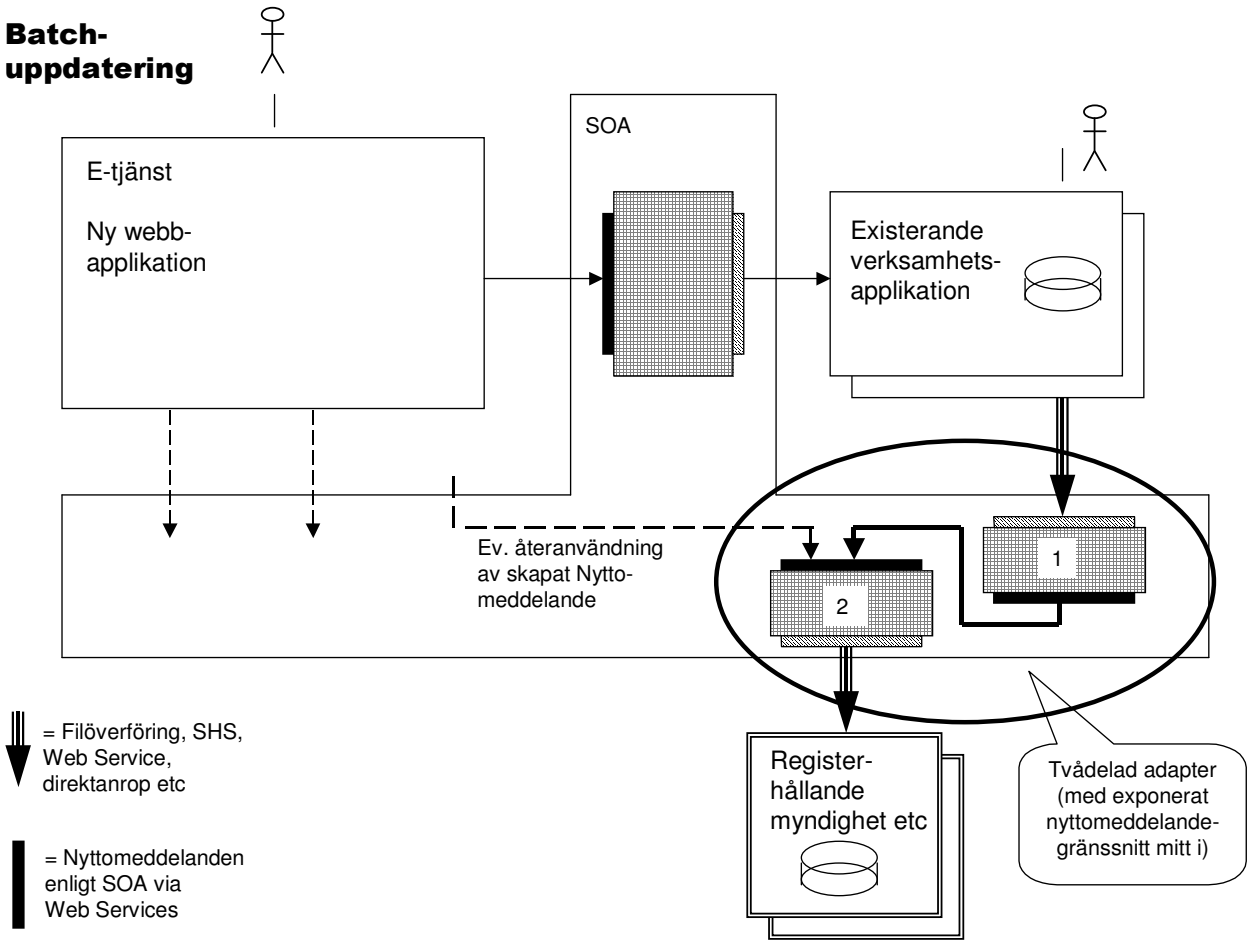
3.9. Mönster för batchuppdatering

För många e-tjänster blir det naturliga att online hämta information via Nyttomeddelanden (från en adapter mot verksamhetssystem). Detsamma gäller för uppdateringar. Därmed kan användbarhet/ergonomi och både medborgarnytta och verksamhetsnytta inom kommunen bli hög (se även kapitlet *E-blankett vs interaktiv webbapplikation*).

I andra sammanhang är det inte lämpligt med online-koppling bakom e-tjänsten. Det kan t ex röra sig om en e-tjänst med stor anstormning av användare vid speciellt tidpunkt (ungefär som e-deklarationen har årsvis eller Internetbankerna har vid månadsskifte) där det inte är rimligt att dimensionera driftmiljön till bakomliggande verksamhetssystem för denna höga last. Det kan röra sig om verksamhetssystem där det inte av olika skäl blir genomförbart att anordna en online-adapter. Det kan handla om att man skulle få hårda beroenden till ett större antal samtidiga online-källor vilket försämrar chansen till god total stabilitet. Det kan också handla om att verksamhetslogiken behöver ”snapshot-data” från en specifik tidpunkt (så är fallet för vissa kopplingar till Bistånd).

I fall då online-koppling inte är lämplig behöver information överföras tidvis och satsvis, det som brukar kallas batch.

Det kan vara värt att nämna att ÖTP inte specifikt berör alla de befintliga batch-kopplingar som finns för verksamhetssystemen i en typisk kommun (dessa hanteras ofta med Decapus/TEIS, SHS eller liknande lösningar). ÖTP fokuserar istället på e-tjänster mot medborgare/företag/föreningar etc. Däremot kan vi behöva hantera ett mönster där kommunicerande part ändå är en verksamhetsapplikation och där vi behöver tillföra ett batch-flöde för att överhuvudtaget kunna förverkliga en e-tjänst, se följande skiss:

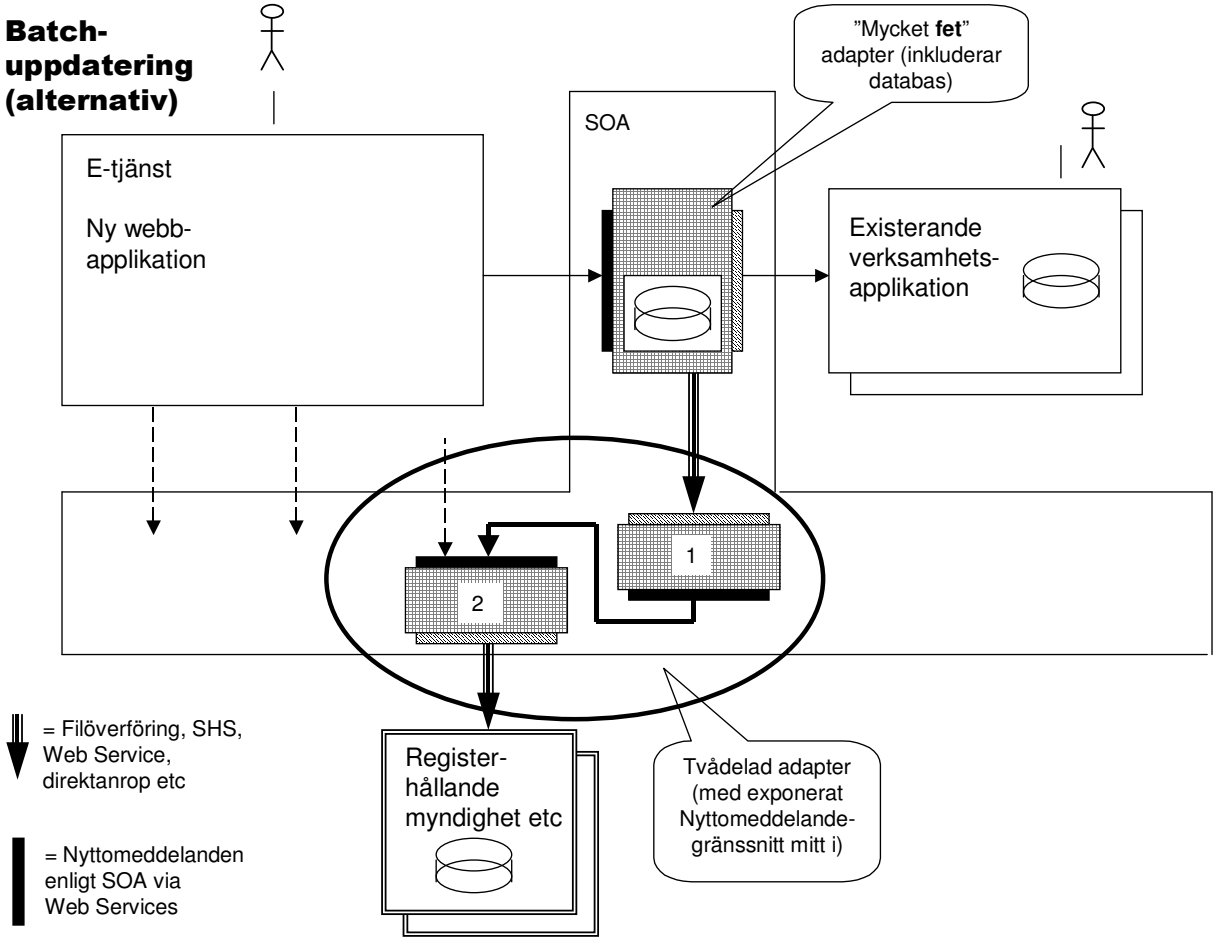


Figur 23

Observera att batchadaptern bör delas i två halvor som åtskiljs av Nyttomeddelande(n) mitt i. Därvid syftas till en ökad öppenhet och återanvändningspotential. Detta brukar kallas för ”kanonisk datamodell”. Det är dock värt att notera att även om ett Nyttomeddelande återfinns som återanvändningskandidat så kan fortfarande andra implementationsskillnader förekomma, t ex synkront/asynkront (se *Kommunikationsprofiler*).

Det är också värt att notera att man bör väga ovanstående konstruktion mot kostnad/nytta/inlåsning med en lösning som Decapus, BizTalk eller motsvarande. Om man redan använder en sådan lösning kan marginalkostnaden att tillföra ett nytt flöde vara låg. Hybrider är också tänkbara där Decapus etc skulle kunna sköta en andel av arbetet hos någon av adapterhalvorna.

För tydlighets skull följer en skiss som visar fallet när det inte är lämpligt (eller går) att kommunicera med verksamhetsapplikationen, utan istället en ”mycket fet” adapter tyvärr måste skapas (vilken innehåller databas):


Batch-uppdatering (alternativ)
E-tjänst
Ny webb-applikation
SOA
"Mycket fet" adapter (inkluderar databas)
Existerande verksamhets-applikation
1
2
= Filöverföring, SHS, Web Service, direktanrop etc
= Nyttomeddelanden enligt SOA via Web Services
Register-hållande myndighet etc
Tvådelad adapter (med exponerat Nyttomeddelande-gränssnitt mitt i)

Figur 24

3.10. Webb-integration

En av de stora utmaningarna är att få delar som är utvecklade på olika håll och i olika tekniker att uppföra sig på ett enhetligt sätt gentemot webb-användaren. Medlemskommunerna har valt helt olika tekniker för sina huvud-webbar (olika Content Management-system – CM, olika teknikplattformar). Olika stuprörssystem som redan har webb-gränssnitt är också utförda på helt olika sätt. Det klassiska är också att vissa webb-lösningar kanske kan tänkas inkludera information från andra webbar (vara ”master”), men få brukar vara förberedda att inkluderas (vara ”slav”).

Webb-integrationen (eller yt-integrationen som man också kan kalla det) har ett antal olika aspekter:

- Ett eller flera webb-fönster
  Ska de olika delarna samsas i samma fönster eller är det godtagbart att andra fönster startas hos användaren?
- Form och färg
  Kommunens form-profil ska användas i de olika delarna
- S k WAI-anpassning för funktionshindrade
  Kan t ex fungera sämre vid användning av s k frames.
- Driftbarhet på olika ställen
  T ex ska Sambruks e-tjänster kunna samdriftas, medan kommunens huvud-webb ofta driftas i egen regi eller i annan outsourcing.
- Sammanhållen inloggningssession – single signon (SSO)
  Ska användaren automatiskt vara inloggad i de olika delarna om man redan är inloggad i en del, eller kan det krävas ny inloggning? Se separat kapitel.
- Sammanhållen katalog
  Även om inte SSO skulle stödjas, att slippa komma ihåg olika användarnamn/lösenord etc till olika webb-delar. Se separat kapitel.
- mm mm

För samtliga aspekterna måste man göra en bedömnning av ambitionsnivå där man beaktar faktorer som nytta, kostnad, inlåsning/öppenhet, komplexitet och kalendertid.

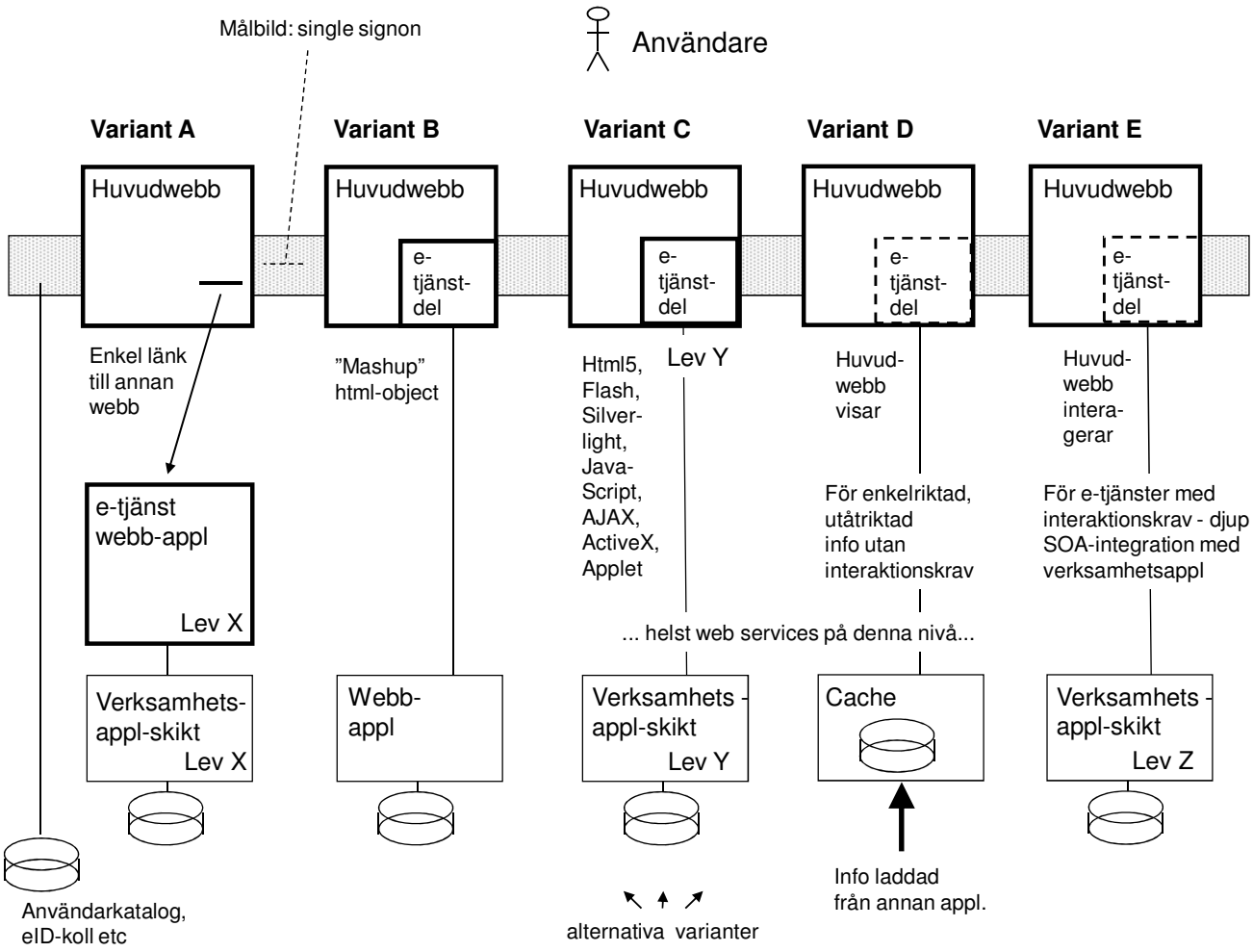
Teknikläget för webb-integration är tyvärr inte tillfredsställande idag. Inom Java-världen finns det en standard som kallas portlets och som är till för att t ex en e-tjänst ska kunna ”sticka fram” sitt användargränssnitt under kontroll av en huvud-webb (portal server). Emellertid finns det oss veterligen inga sådana portlets ingående i verksamhetsapplikationer för kommuner idag, till stor del eftersom få av dem är skrivna i Java.

Microsoft har ett liknande koncept som heter webparts och som används i deras Sharepoint Portal Server. Dock är detta på motsvarande sätt som för Java begränsat till Microsofts utvecklingsmiljöer. Inte heller här känner vi till att det finns färdiga sådana webparts för verksamhetsapplikationer idag.

Det finns en ansats som heter WSRP som är tänkt att kunna länka ihop de här inkompatibla teknikmiljöerna bl a så att en portlet-portal ska kunna visa upp info från en webpart och vice versa. I dagsläget verkar det dock tveksamt om denna standard verkligen får stöd i praktiken. Eftersom ÖTP bör vara konservativt vad gäller nya standarder och endast använda brett

spridda och väl beprövade tekniker så är vi än så länge avvaktande. Området bör dock bevakas.

I nedanstående skiss visas ett antal tänkbara alternativ för webb-integration som alla har för- och nackdelar. Vad bilden framförallt vill visa är att det trots allt finns lösningar som går att använda redan idag, även om de inte är så eleganta som portlets/webparts/WSRP skulle varit om detta varit mer öppet, väl beprövat och väl spritt.



Figur 25

Det är ofta troligt att vi på grund av den dåliga tekniska standardiseringen får tänka oss ett av de två vänstra alternativen, kanske allra mest troligt det vänstra (för att ändå minimera ett antal tänkbara tekniska krångligheter som också följer av att Sambruks e-tjänster ska kunna samdriftas rationellt – s k ASP-drift). En ökande användning av variant C har docks märkts de senare åren. Den framväxande standarden Html5 är extra intressant eftersom den minskar inlåsning i proprietära miljöer som Flash. I detta sammanhang bör också nämnas den starka framväxten av ”appar” för smarta telefoner och läsplattor.

3.11. Användbarhet

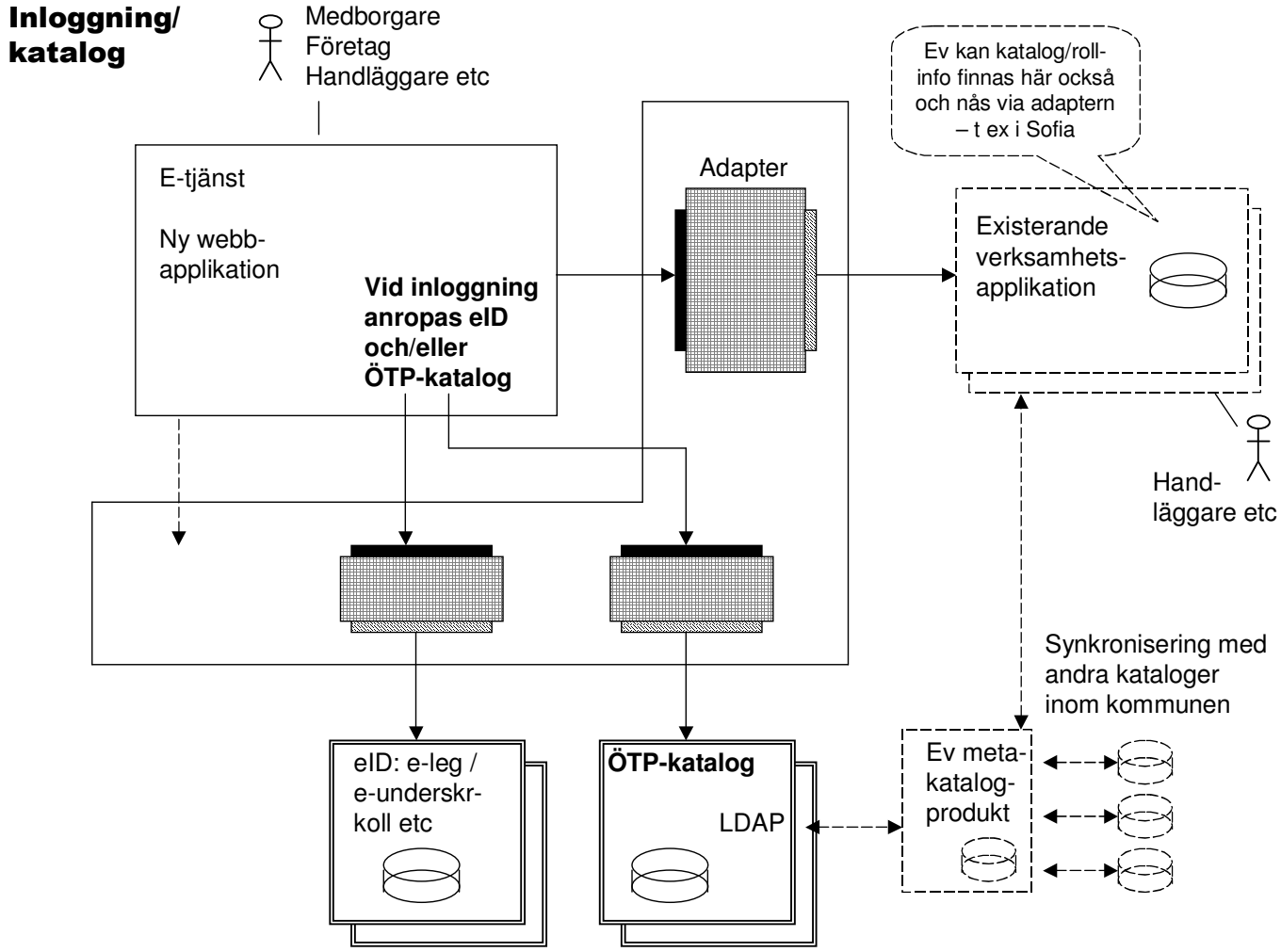
Användbarhetsområdet är egentligen mycket stort. I ÖTP:s konkreta anskaffandeunderlag finns vissa grundläggande krav, se bl a [ÖTP-Upphandl]. Värt är dock att notera att design av god användbarhet startar redan när verksamhetsprocesser utformas

Ett annat råd är att undersöka ifall det behövs flera parallella användargränssnitt mot samma funktionalitet. Detta är fallet då man har stor skillnad mellan sällananvändare och expertanvändare. Dessa två kategorier har fullständigt olika krav. Sällananvändarna måste få utförliga textförklaringar, förklarade koder (eller helst inga alls), hjälptexter och tips. Oftaanvändaren behöver istället snabbtangenter, kondenserade bilder som kan tillåtas visa icke-förklarade koder, tät integration med ordbehandling och kalkyl mm.

3.12. Sammanhållen inloggning/användarkatalog

Sammanhållen inloggningssession – single signon (SSO) och användarkatalog är två viktiga och svåra teknikområden (ibland tyvärr också kostnadsdrivande).

Sambruks ansats i detta läge är att ha en något konservativ attityd eftersom området både är snårigt och kostsamt. Vi kan därmed inte ha ambitionsnivån att klara SSO gemensamt med alla de tänkbara webbinloggningar som kommunerna redan idag har eller skaffar, var för sig. Däremot måste ansatsen vara att alla nya e-tjänster som tillverkas efter nuvarande specifikationsfas ska ha chans till SSO, enligt nedanstående skiss:



Figur 26

Det är värt att poängtera att SSO och integrerad (eller synkroniserad) användarkatalog är två separata frågor att ta ställning till från fall till fall. SSO kan i vissa fall tänkas utföras utan integrerad/synkroniserad användarkatalog liksom att det sistnämnda utan SSO trots allt ger

en stor bekvämlighetsökning för medborgaren genom att man endast behöver hålla ordning på en omgång användarnamn/lösenord (även om man måste logga in upprepat).

Den äldre Borlängepiloten för Bistånd har verksamhetskrav på eID-inloggning, liksom att roll-info ligger i verksamhetsapplikationerna. De nyare piloterna har däremot endast verksamhetskrav på användarnamn/lösenord varför SSO ändå inte skulle göra nytta gentemot Bistånd.

I fallet SSO kan vi inom den närmaste framtiden välja två ansatser: Att låta utveckla e-tjänsterna webb-applikationer för samma teknikplattform varvid denna torde klara SSO. Nackdelen bleve att lägga flera ägg i en korg. Ska man med disparata teknikmiljöer ändå klara SSO får man skapa ett sätt att skicka med ”credentials” emellan applikationerna. Flera sådana sätt finns tillgängliga även om de medför ett visst krångel.

En ytterligare anledning till att inte i dagsläget göra dyra åtaganden vad gäller stora SSO-produkter är att teknikutvecklingen fortgår relativt snabbt. SSO-standarden SAML2 har slagit igenom men man bör beakta att även om en part följer SAML2 så finns det risker för olika inställningar i headrar etc vilket kan förhindra interoperabilitet. Mekanismer som Google Accounts och Live ID förs fram samt vidareutvecklingen av operativsystem mm pågår. Sociala medier som Facebook och Linkedin kan också komma att bli en möjlighet för enklare sorters SSO i framtiden. Särskilt för servicegivning som bibliotek, simhall, teater räcker ofta den sortens säkerhetsnivå. Området bör alltså bevakas noga.

Observera att den relativt nya Nyttomeddelandespecifikationen för Ärendehändelse har kategoriserat upp inloggningskvalitet eftersom man kan behöva veta detta relaterat till informationsklassning och visning för olika parter.

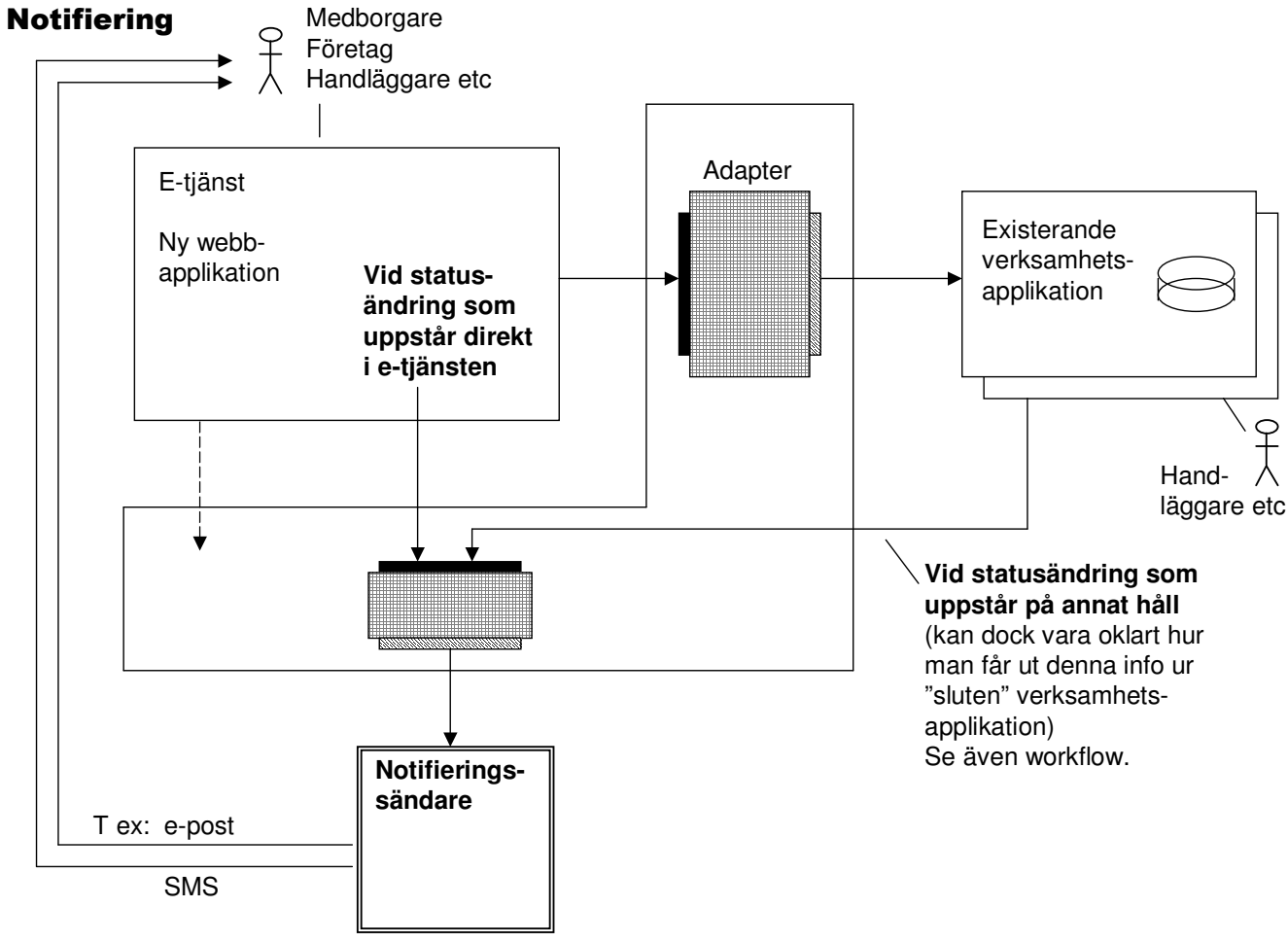
3.13. Notifiering

Med notifiering (kallas även ibland avisering) menas här att skicka någon slags meddelande till en medborgare som innebär att en status som är relevant för medborgaren har förändrats, t ex att handläggningsbeslut i ett ärende nu är fattat.

Notifiering kan tänkas skickas via osäkra kanaler varför ingen känslig information ska inkluderas i meddelandet i sig – i sådana fall är notifieringen endast en signal om att medborgaren bör logga in i e-tjänsten, Mina Sidor etc och där på ett tillräckligt säkert sätt kunna ta del av statusändringen.

Typiska kanaler idag är vanlig e-post och SMS till mobiltelefon. I många fall kan det vara önskvärt att notifieringstexten innefattar en URL till e-tjänsten.

Se även kapitlet *Processtöd med hjälp av workflow*.



Figur 27

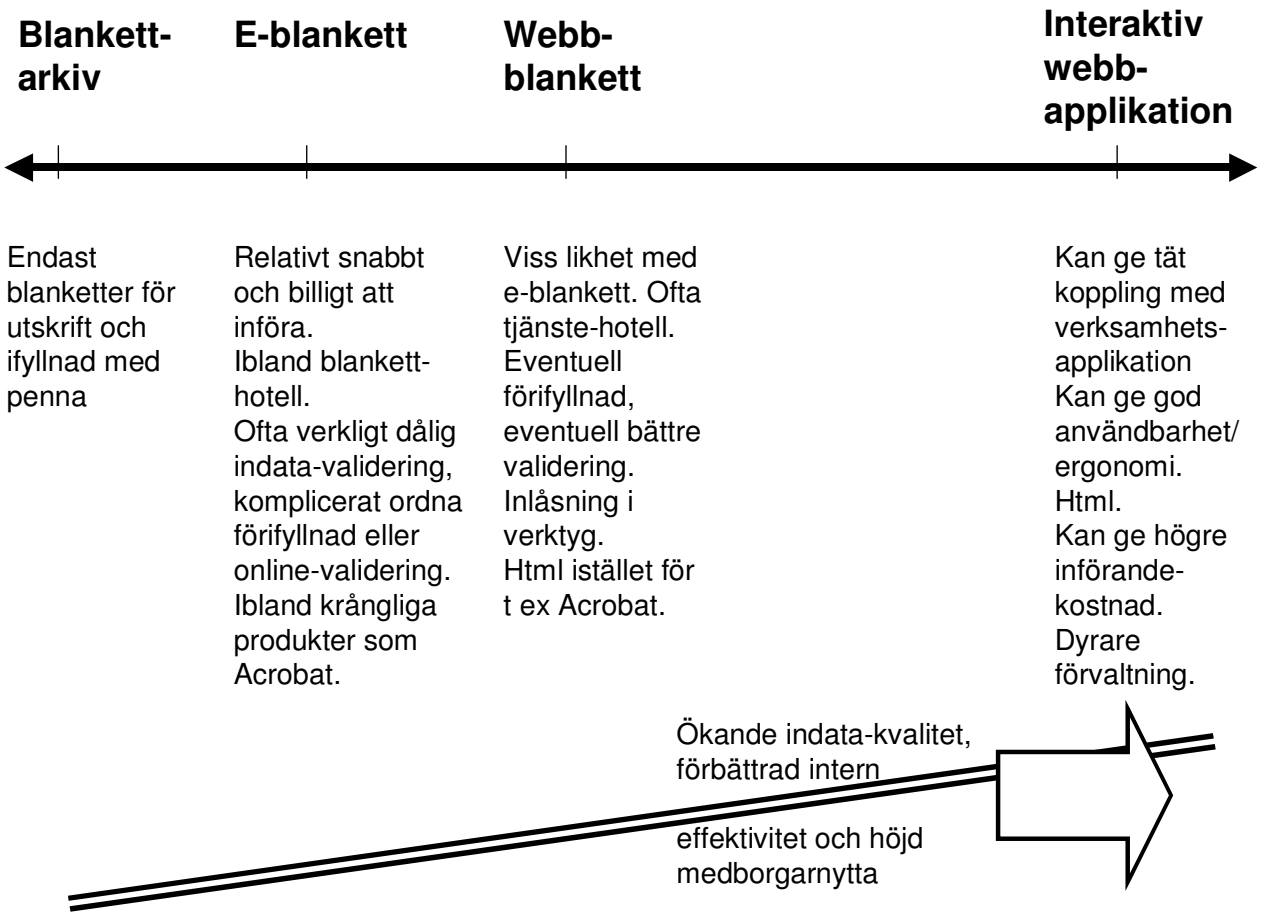
Se även här den relativt nya Nyttomeddelandespecifikationen för Ärendehändelse där en definition av notifieringskanaler ingår samt föreslagen hantering av ouppdaterad och/eller krockande mobilelefonnummerdefinitioner inom olika delar av kommunen, samt liknande hantering för e-post.

3.14. E-blankett vs interaktiv webbapplikation

Detta avsnitt handlar främst om avvägningen mellan pappersblankett-efterliknande lösningar och mer interaktiva webbapplikationer (i stil med Internetbanker och webbutiker).

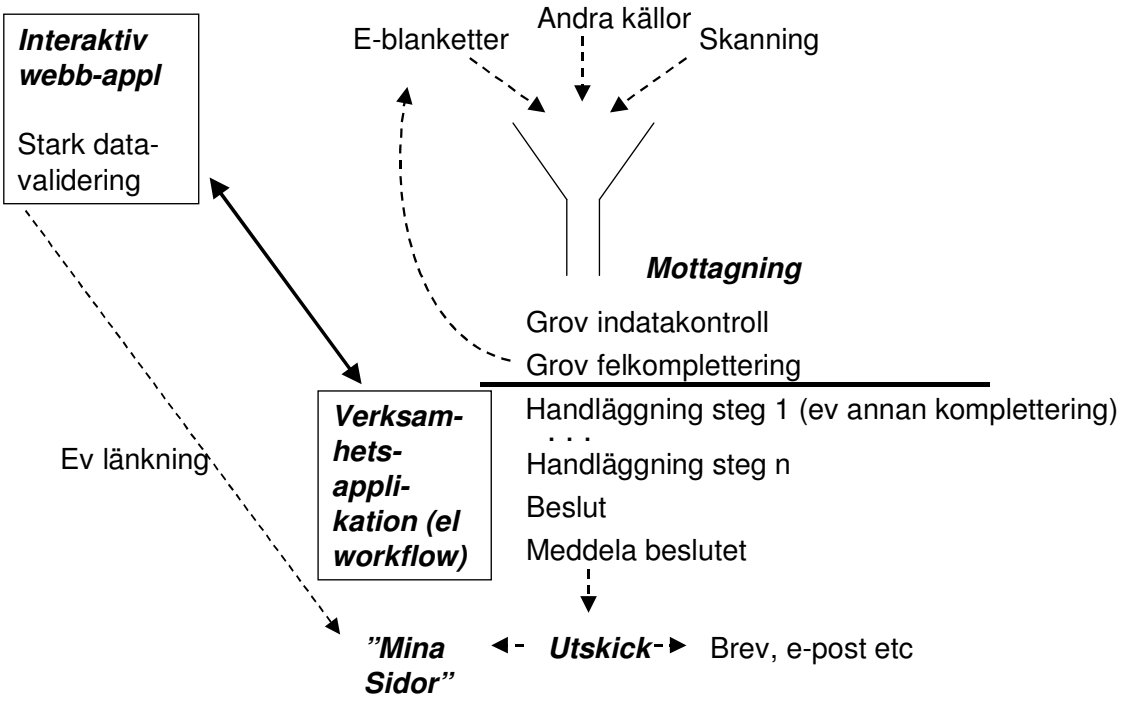
Det kan vara värt att tydliggöra möjligheterna att i varierande grad uppfylla verksamhetskrav på effektivitet och nytta, liksom hur väl man kan betjäna medborgaren.

- Problembild vid alltför simpla formulär
    - Troligen ej anpassning av användardialog successivt efter vad som tidigare kryssats i (dvs ”logiskt döda fält” fortfarande ifyllbara)
    - Troligen ej presentation av strikt korrekta värdeförråd (t ex drop-down-listor) – vem ska översätta fritt inmatat ”Gymnasiet i Lidköping” eller ”De la Gardieskolan” till ”Lidköpings gymnasium” som det står i databasen, ger risk för manuellt extraarbete
    - Troligen ej verksamhetslogik som tätt samverkar med användargränssnittet
    - Troligen ej stark indatavalidering, risk för många kompletteringar


Blankett-arkiv
E-blankett
Webb-blankett
Interaktiv webb-applikation
Endast blanketter för utskrift och ifyllnad med penna
Relativt snabbt och billigt att införa.
Ibland blankett-hotell.
Ofta verkligt dålig indata-validering, komplicerat ordna förifyllnad eller online-validering.
Ibland krångliga produkter som Acrobat.
Viss likhet med e-blankett. Ofta tjänste-hotell.
Eventuell förifyllnad, eventuell bättre validering.
Inlåsning i verktyg.
Html istället för t ex Acrobat.
Kan ge tät koppling med verksamhets-applikation
Kan ge god användbarhet/ ergonomi.
Html.
Kan ge högre införande-kostnad.
Dyrare förvaltning.
Ökande indata-kvalitet, förbättrad intern
effektivitet och höjd medborgarnytta

Figur 28

Interaktiv webb-appl vs andra källor



Figur 29

Kommentarer till ovanstående bild:

- Den högra varianten med "tratten" är den som brukar förekomma inom stora statliga verk etc.
- Dock riskerar man där att missa indata-kvalitetsproblemet, att ärendehandläggning kan bli så oerhört mycket effektivare ifall data verkligen är korrekt och antalet kompletteringar kan hållas lågt eller obefintligt.
- T ex ifall en e-blankett-lösning innehåller viss validering så kastar man ner den högre kvaliteten i samma röra av skannade blanketter med mycket lägre datakvalitet.
- En interaktiv webb-appl som har en stark indatavalidering (jmf Internetbank) har mycket större möjlighet att lågt kompletteringsbehov, men så måste också det datat föras in på rätt kvalitetsnivå i handläggningsstegen.
- Vissa legala aspekter måste tillgodoses. Även om en Internetbank kan ha gjort sig av med blankett/dokument/ärende-begreppet fullständigt kan en kommun t ex behöva lagra undan en ärendelogg där exakt utseende på ifylld ansökan etc i undantagsfall ska kunna sökas fram vid efterhandsutredning.

Inom Sambruk fokuserar vi hittills på interaktiva webb-applikationer men vi måste också i framtiden hitta mönster för att avgöra när en viss ärendetyp förtjänar en investering i en avancerat interaktiv e-tjänst, eller när det totalt sett blir vettigare med en enklare lösning. Troligen ska frekventa ärendetyper hanteras med mer avancerade lösningar, medan sällanärenden kanske får tas om hand med e-blanketter e dyl.

3.15. Var datalagring sker

Beroende på verksamhetskraven för e-tjänsten och på kapabiliteten hos använd verksamhets-applikation kan det uppstå behov av datalagring på fler ställen än i verksamhetsapplikationen i sig.

Dessutom, beroende på återanvändningsaspekter och när i tiden olika delar implementerats kan datalagringsbehov uppstå både i e-tjänste-applikationen och i anpassningsskiktet.

Man bör dock vara noggrann med att definiera ett tydligt ”Master”-förhållande och manuella processer för att undvika data-diskrepanser.

Frågan om var datalagring sker är mycket parallell med resonemangen i kapitlet om process-aspekten.

3.16. Stora, asynkrona dataflöden

Stora asynkrona dataflöden bör behandlas i särskild ordning. Arkitekturbilderna i denna skrift behandlar mest online-fallet och i viss mån asynkron fråga-svar.

Samtidigt finns det traditionellt sett ofta befintligt stöd för filöverföring och batchuppdateringar. Att välja att gå ifrån dem bör göras ur ett cost/benefit-perspektiv.

Asynkrona flöden har naturligtvis fördelar (lösare koppling ger bättre chans för bra tillgänglighet, svarstider är inte problematiska etc) men också nackdelar (dålig info-aktualitet, risk att processer inte kan utformas så att de utförs i ett enda svep vilket vanligen är billigast för verksamheten etc).

ÖTP har avsnitt om EAI/ESB-perspektivet (vilket ofta lyfts fram när man vill modernisera gamla fil-flöden). Dock kan man även med framgång använda SHS för vissa former av EAI/ESB. Här bör dock nämnas att Web Service-gränssnittet till SHS (version 1.2) som anpassningsskiktet främst tänks använda, troligen endast klarar ca 1 MB per asynkront meddelande. Vill man skicka större mängder utan att segmentera datat så kan man installera en SHS-satellit/klient/nod i samband med integrationsskiktet så att de äldre SHS-API:erna som klarar större datamängder per meddelande kan användas. Detta alternativ stöds också av Infratjänsten. Se även kapitlet *Principer för Nyttomeddelanden*.

Om man inte skulle välja SHS och inte heller traditionell filöverföring kan man gärna överväga temporärlagring i relationsdatabas. Som front-API kan man t ex använda Web Services. Därmed har man skapat en mycket drifttålig kö för asynkront bruk som vanligen inte kräver några extra plattformsinvesteringar (såsom MSMQ, IBM MQ etc)

Man bör också notera att gammaldags avstämningsrutiner e.dyl. oftast är nödvändiga för att kvalitetssäkra asynkrona flöden.

Se även kapitlen *Mönster för batchuppdatering* och *Applikationsintegration EAI/ESB*.

3.17. Återanvändning vid ytterligare e-tjänst

Återanvändningsaspekten är en av kärnfrågorna i visionen för Sambruk. Lyckas man med detta kan kostnader och ledtider sänkas rejält per kommun. Samtidigt måste man vara realist, en stor mängd förväntningar på hög återanvändningsgrad under 1990-talet kom på skam. Ofta berodde det på ett överteoretiskt förhållningssätt, flummiga förväntningar och framförallt på organisatoriska aspekter (vem ska ansvara, förvalta, missionera, svara för flexibilitet) ego, ”not-invented-here” och bristande ekonomimodeller.

Nedan följer några återanvändningsnoteringar per del av någon lösning enligt ÖTP:

För e-tjänstens **webb-applikation**:

- Återanvändning av webb-applikation bland många kommuner borde vara ganska lätt, förutsatt att:
    - Processen och detaljutförandet som stöds av utvecklad e-tjänst-applikation kan accepteras ”ego-less” av återanvändande kommuner utan modifieringar.
    - IT-infrastrukturen som valdes vid programutvecklingen godtas av de återanvändande kommunerna.
    - Samma verksamhetsapplikation används av återanvändande kommun, eller att den verksamhetsapplikation man använder är rimligt ”typisk” så att anpassningslogik-skiktet har en chans, och inte e-tjänsten måste modifieras.

För en viss **anpassningslogik** (mellan en viss e-tjänst och en viss **verksamhetsapplikation**):

- Återanvändning av anpassningslogiken bör vara rimligt lätt om samma Nyttomeddelanden efterfrågas och samma version av verksamhetsapplikationen används.

För en viss **anpassningslogik** (mellan de allra flesta e-tjänster och **centrala register**):

- Återanvändning bör vara lätt eftersom troligen samma Nyttomeddelanden efterfrågas i många e-tjänster.

För en viss **anpassningslogik** (mellan de allra flesta e-tjänster och **e-leg/e-underskrift**):

- Återanvändning bör vara lätt eftersom troligen samma Nyttomeddelanden efterfrågas i många e-tjänster. Dock förutsatt att samma Infratjänsteleverantör och leverantörer av e-leg/e-underskrift används.

Fallet att en kommun inte anser att existerande Sambrukslösning helt skulle räcka till relativt upplevd behovsbild är värt en liten genomgång. Ifall Sambruk äger programkoden eller Open Source eller liknande principer används är det förstås möjligt att en återanvändande kommun gör modifieringar, men då blir naturligtvis den totala förvaltningssituationen sett över hela Sambruk mycket mindre optimal. Förvaltning kommer ändå att bli en komplicerad utmaning, även utan en mängd kommun-specifika varianter.

En annan lösning är att designa e-tjänste-applikationen till att vara parameterstyrbar för att kunna anpassas till varierande kommunkrav. Detta kan fungera väl om mängden parametrar hålls lågt, i annat fall blir varje införande dyrt och komplext (jämför med den ofta mycket höga införandekostnaden för stora, parameterstyrbara ERP-system såsom SAP).

4. RÖRLÄGGNING

Detta kapitel berör val som gjorts inom ”Öppen teknisk plattform” vad gäller grundläggande kommunikationssätt och integrationsteknik.

En av de faktorer som gjort ”Öppen teknisk plattform” överhuvudtaget rimlig att börja arbeta med är tekniken som för några år sedan kommit att etableras och som ger stor interoperabilitet mellan olika tekniska plattformar, nämligen enkla Web Services enligt SOAP/http. Alla de olika kommunerna kan inte förväntas göra exakt samma teknikval och dessutom finns många investeringar redan gjorda i verksamhetsapplikationer utvecklade/driftade i diverse tekniker. För Sambruk tänker vi oss att parter som kommunicerar via Web Services är ”kända för varandra” och exekverar i kontrollerade, interna driftsmiljöer. Därmed bör inte säkerhetsfrågor ge problem. Därmed finns det heller ingen direkt anledning att idag använda en aktiv UDDI (en Web Services-katalog). Möjligen kunde man använda UDDI som en ren dokumentations-katalog.

Här kan beröras den process/objektmodellering som nämns i kapitlet *Process-aspekten.* Den hypotetiska mycket höga teoretiska ambitionsnivån som där nämns skulle ha krävt modeller, utvecklingsverktyg, driftmiljöer mm för tämligen fingranulära objekt samtidigt med välfungerande objektkommunikation mellan disparata teknikmiljöer. För detta fanns det stora förhoppningar i början av 1990-talet, men de infriades huvudsakligen aldrig.

Istället har vi alltså valt att i plattformen ”kapsla” in verksamhetsapplikationer relativt grovgranulärt, utgå ifrån deras inneboende möjligheter och i e-tjänstens webbapplikation lägga till det som sedan inte visade sig räcka till.

Integrationen mellan e-tjänsten och verksamhetsapplikationen blir därmed mera av ”Service Oriented”-slag (SOA) med relativt lösa kopplingar (t ex utan den distribuerade ”object life cycle management” som ställt till stora problem i tätt kopplade arkitekturer) och med Web Services som grundprincip.

ÖTP har fokuserat på SOAP som transport i samband med Web Services. Detta är troligen fortfarande huvudalternativet, men den enklare varianten för "Web Services" som kallas REST kan ofta övervägas som alternativ. Eftersom Nyttomeddelanden är tänkta att vara transportoberoende ska detta inte vara ett problem. Dock måste man i faktiskta införanden avväga hur ortodoxt man ska använda REST.

5. PRINCIPER FÖR NYTTOMEDDELANDEN

Nyttomeddelanden infördes redan i ÖTP V1.1 (till Borlängepiloten för Bistånd). Vi valde under det spec-arbetet (år 2004) att vara mycket konkreta och även skriva XML-definitionsfiler för att markera att skapande/införande av en lösning kunde ske tämligen omedelbart.

I och med att vi i de senare projekten skapar flera krav-specar samtidigt får vi ett utökat strukturbehov för hur vi ska hantera Nyttomeddelande-definitioner, varför vi i ÖTP v1.2

förfinade sättet att specificera meddelanden. Samtidigt har vi lärt oss av erfarenheter och framförallt har nu Statskontorets/E-nämndens/Vervas ”05:01 Riktlinjer för utveckling av standardmeddelanden för förenklat informationsutbyte med elektroniska standarddokument” getts ut i februari 2005. Sambruks inriktning är att så långt möjligt följa dessa riktlinjer och vad vi förstår utgör alla hittillsvarande Nyttomeddelanden tillika Standardmeddelanden.

Det nya sättet att speca består av följande:

- Ett dokument innehållande **Begreppsmodell**.
  Här definieras naturligtvis begrepp (semantik).
  Detaljdefinitioner inkluderas av paket, grupper, typer och termer, se figur nedan.

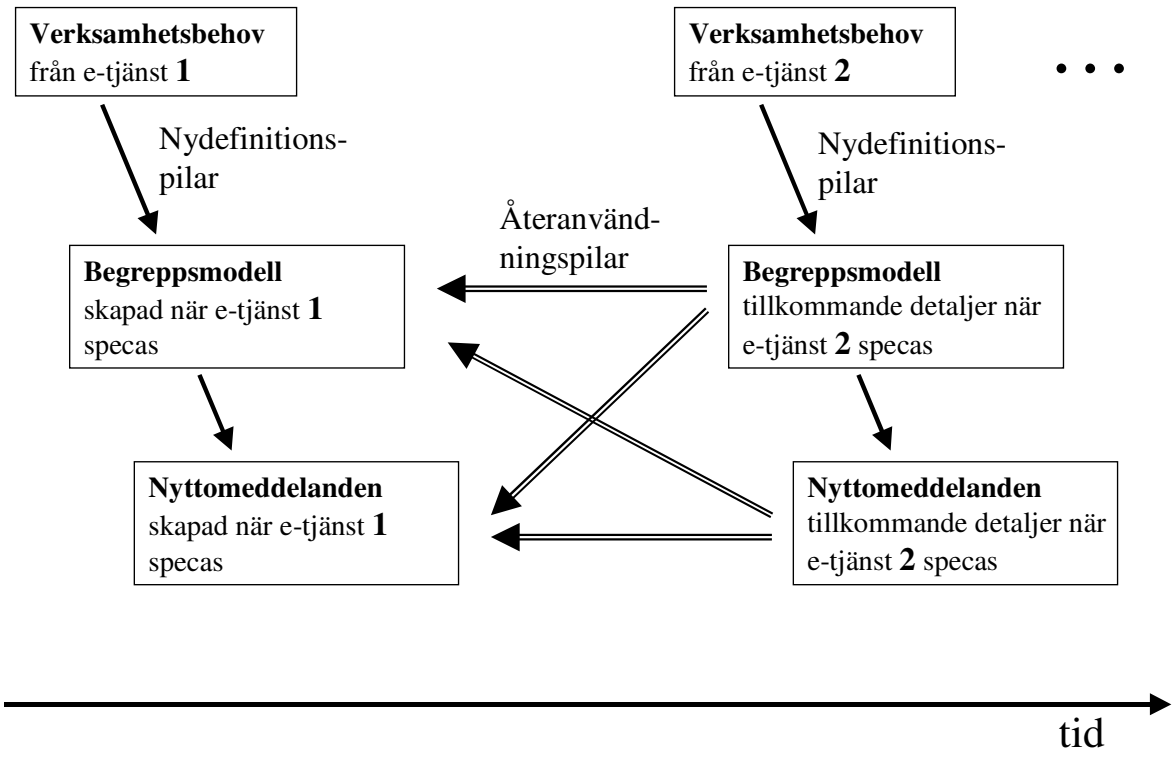
- Ett dokument innehållande **Nyttomeddelanden**.
  Här beskrivs syfte och användning av meddelanden, som i sin tur pekar ut detaljdefinitioner specade i Begreppsmodellen.
  Dokumentet definierar också de anrop som använder Nyttomeddelandena (begäran samt svar).

Framgent har vi höga förväntningar på att en mycket god återanvändningsgrad av Nyttomeddelanden ska uppstå. Återanvändning ställer som bekant mycket stora krav på missionering, struktur, ordning, versionshantering, förvaltning och ekonomiska incitament. Vi kan därför räkna med att sättet att definiera och underhålla Nyttomeddelanden kommer att behöva utvecklas vidare över tiden. På sikt kanske uppdelning i ett stort antal del-filer behövs, liksom användning av ett configuration management-verktyg samt i övrigt strikta versionshanteringsprocesser och tydligt utpekade ansvar.

Återanvändning förväntas ske på alla nivåer i skissen; Anrop, Nyttomeddelanden, paket, grupper och typer. Detta får bli beroende av respektive verksamhetsbehov. Ju högre upp som återanvändningen sker, desto bättre återvinningseffekt, men ibland kommer det inte att bli möjligt. Då kanske återanvändningen t ex ofta sker på Paket-nivån.

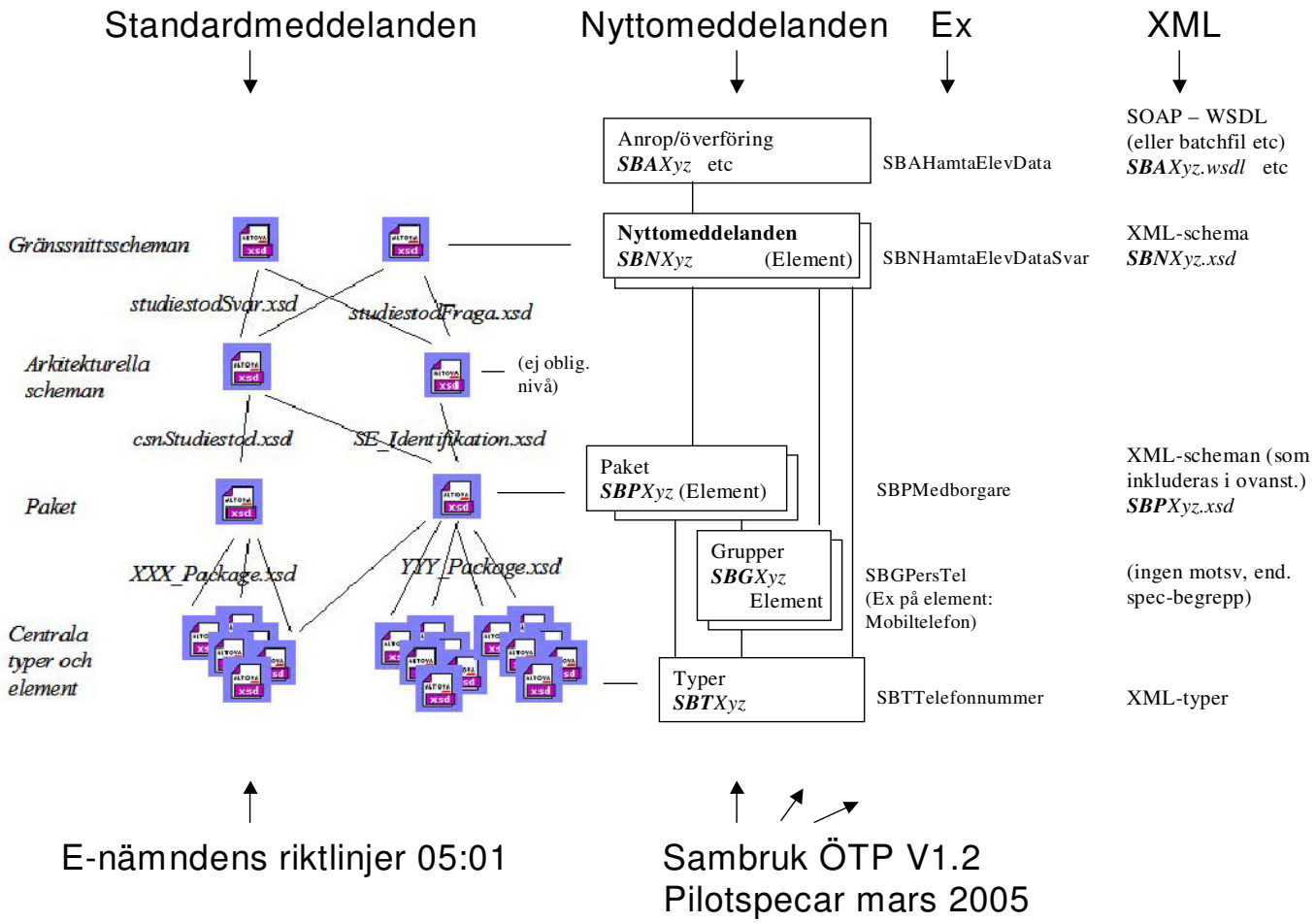
Som man förstår av följande bild blir det synnerligen viktigt att ha god ordning på återanvändning, configuration management och kravspårbarhet:

Successiv återanvändning av nyttomeddelanden/begrepp över tiden



Figur 30

Följande skiss belyser hur Nyttomeddelanden förhåller sig till Standardmeddelanden samt ger även kopplingen till konkret XML:



Figur 31

En anmärkning är att Nyttomeddelanden förväntas kunna transporteras inte bara av SOA-Web Services via SOAP, utan lika väl av REST, SHS eller enkla filer etc. Detta är anledningen till uttrycket "Anrop, överföring" i översta boxen i figuren.

En viktig aspekt av SOA, Web Services och modelleringen av storleken på de Nyttomeddelanden som förmedlas är avvägningen av det som brukar kallas ”chatty vs chunky”, dvs ska maskingränssnittet småprata ofta med små datamängder i taget eller mer sällan och istället med större datamängder i taget. All erfarenhet pekar på att man inte bör ha SOA av chatty-karaktär, det skulle strida mot tankar på självständighet/inkapsling, att SOA har mer teknisk overhead än t ex SQL-protokollen, andra prestandaproblem mm. Att endast kapsla in ett existerande client/server-gränssnitt av småprats-karaktär i Web Services fungerar alltså vanligen inte bra.

Mot detta ska ställas några aspekter som talar emot alltför stora chunky-meddelanden. Det kan gälla tiden det tar att presentera första bilden för användaren. Det kan gälla att alltför mycket information mellanlagras (”cacheas”) som sessionsdata i webbservern vilket i sin tur kan ge prestandaproblem över långsamma förbindelser till slutanvändaren (t ex mobilt bredband) genom att sessionsdatat i många lösningar forslas fram och tillbaks mellan webbläsare och webbserver. Alternativt, ifall sessionsdatat vidmakthålls i webbservern kan man få problem med clusterlösningar (för lastdelning respektive feltolerans). En annan aspekt är att cacheat data kan riskera att bli ofärskt jämfört med det som står i den egentliga databasen (vanligen i verksamhetsssystemet) så att användaren fattar beslut utgående från felaktiga premisser. En ytterligare risk är att sannolikheten ökar för att uppdateringar av samma information ska krocka. Det finns knappast aldrig någon möjlighet att hålla hårda lås i databasen under läsning/visning/inmatning/uppdatering. Inte heller kommer det med tanke på verksamhetssystemens slutenhet att gå lätt att ordna en strikt s k ”optimistic locking”. Alla dessa saker pekar alltså på att inte ha alltför stora meddelanden som utväxlas alltför ofta.

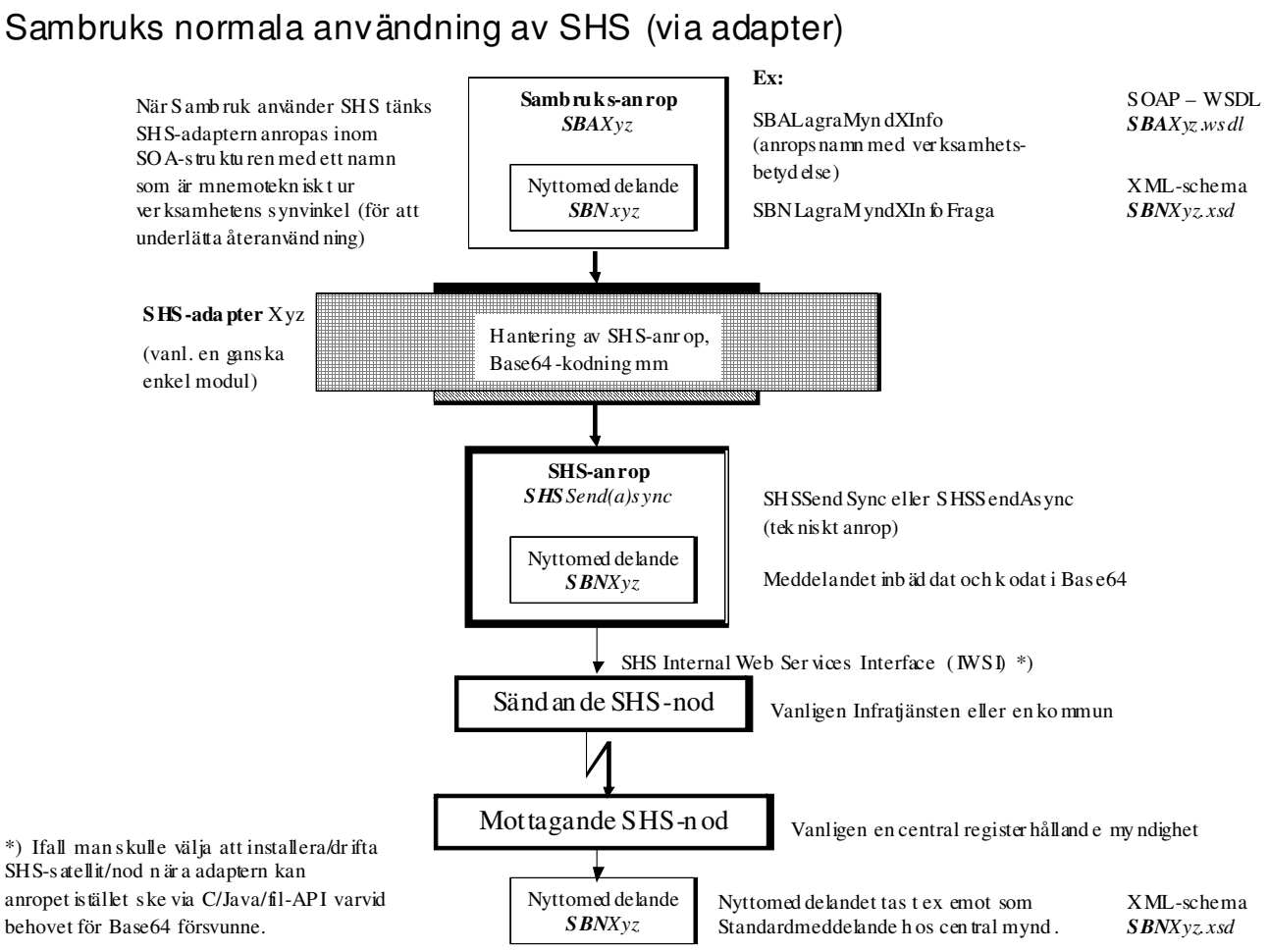
Som i så många sammanhang blir det alltså frågan om göra en vettig avvägning och en optimering vad gäller storlek på Nyttomeddelanden och hur färskt datat ska vara i webbservern.

En annan problematik är storlek på meddelanden. I och med att vi önskar en mycket bred användbarhet hos diverse olika utvecklingsverktyg och exekveringsplattformar och parallellt också i och med att vi angivit en konservativ användning av XML-standarder (t ex att Nyttomeddelanden ej ska skickas som attachments) så ansluter vi oss till bedömingen gjord i SHS V1.2 att Nyttomeddelande skickat i ett web service-anrop kan vara max ca 1 MB stort. I föreliggande piloter har vi inte sett behov av segmentering eller ”fjärrbläddring” utan vi kan helt enkelt sätta maxantal för förekomster (per definierat Nyttomeddelande). Självklart ska en returkod efter anrop kontrolleras, vilken då anger ifall svaret trunkerats. Användargränssnittet får därvid visa upp en informationstext och t ex användaren får ange fler sökvillkor så att sökresultatet inte blir så stort.

Sambruk planerar också att anordna en slags testbänk, "validator", för Nyttomeddelanden, tillgängligt från Internet. I så fall kan viss testning också utföras gentemot denna. Testbänken avses kunna ge viss syntaxkontroll samt kunna visa/skicka dummydata. Denna möjlighet torde också kunna ingå i acceptanstest.

5.1. Nyttomeddelanden via SHS

I samband med att en e-tjänst inom Sambruk anropar SHS för informationsutbyte, t ex gentemot en central myndighet, tänker vi oss följande meddelandestruktur:



Figur 32

6. REGLER, KONVENTIONER, DRIFT

6.1. Informationsöverföring

För att konkretisera vad som i referensarkitekturen förstås bl a med Web Services och för att tillförsäkra interoperabilitet specificeras bruk av vissa standarder och mekanismer.

För att verkligen åstadkomma praktisk interoperabilitet ska endast enkla Web Services användas. De nyare och ofta mycket komplexa standarderna för Web Services (ibland sammanfattade WS-*) bör undvikas om det inte kan tydligt påvisas att alla potentiellt interagerande programvarudelar verkligen är interoperabla enligt dessa standarder.

Specifikt rekommenderas de enkla Web Services-standarderna SOAP 1.1 och 1.2, se [ÖTP-Upphandl]. De nya, ännu enklare Web Services-konventionerna REST[5] (vilka direkt baseras på http-standarderna) kan i många fall vara användbara. Såsom beskrivs i kapitlen *Översikt referensarkitektur applikationsintegration generellt* och *Principer för Nyttomeddelanden* så tänks Nyttomeddelanden specificeras transportoberoende, varför olika val som SOAP, REST, filöverföring, RSS/Atom, SHS etc kan vara tänkbara parallellt för att ge så stor öppenhet och interoperabilitet som möjligt.

Området informationsskydd är mycket stort. Här inkluderas endast några konkreta avsnitt som har extra relevans relativt ÖTP. (Se även säkerhetsutredningen Saekklassn_v10_050120.pdf tillgänglig på www.sambruk.se.)

Web Services-anrop måste utföras på ett tillräckligt säkert sätt relativt respektive överförd informations sekretesskrav , skyddsbehov mot manipulering etc. Detta kan åstadkommas på många sätt, t ex kan en viktig lösning vara SOAP/https för att ge högkvalitativ SSL tillsammans med både klient- och server-certifikat, eller så kan VPN-tunnlar användas punkt-till-punkt, eller noggrann nätsegmentering med VLAN-teknik, mm. Liknande teknik kan också användas med annan kommunikation, t ex filöverföring. Vid samdrift kan det också uppstå behov av datakommunikation från samdriftplatsen och till en verksamhetsapplikation i en kommuns egna driftplats och denna följer troligen något specifikt protokoll såsom DCOM, RMI eller Transact-SQL vilket måste påverka vald säkerhetslösning. Allt detta ska förstås kompletteras med normal säkerhet för datahallar, data-kommunikation, personal etc.

Se även avsnittet *Dataintegritetet/säkerhet inom EAI/ESB*.

Elementdata för Nyttomeddelanden ska i normala fall vara enligt teckenuppsättingen 8859-1 eller Unicode, vilket måste specificeras noga. Undantag specificeras och motiveras i Nyttomeddelanden och Begreppsmodell.

Återförsök efter en tid (s k retry-on-timeout) bör programmeras in i anropande ända för att skapa tillförlitlighet i lösningen (Web Services, framförallt t ex via en tunnel genom Internet,

[5] REST: REpresentational State Transfer

har ju viss risk att aldrig komma fram). Vid retry-on-timeout kan dubletteliminering behövas, enligt respektive verksamhetskrav, s k idempotency. I de fall anropet direkt t ex ska förse ett användargränssnitt med information må det vara tillåtet att inte göra retry-on-timeout programmatiskt, utan istället ge användaren en förståelig feltext och låta denne försöka igen.

6.2. Driftsegenskaper och infrastruktur

Kraven från en IT-lösning på klientdatorer etc brukar inte vara i fokus i samband med e-tjänster, icke desto mindre inkluderas viss kravbild i kravmatrisen för mer allmänt upphandlingsbruk i en kommun.

6.3. Kommunikationsprofiler

Kommunikationsprofiler är till för att gruppera ihop ett antal vanliga kombinationer av icke-funktionella egenskaper hos integrations-gränssnitten. Därefter kan specifikationen för Nyttomeddelanden referera till någon befintlig kommunikationsprofil (eller åstadkomma en ny om så behövs).

Syftet är tvåfaldigt: Dels slipper Nyttomeddelande-specen bli så rörig och voluminös, dels är det mindre risk att man glömmer specificera alla icke-funktionella egenskaper.

Icke-funktionella egenskaper kan vara t ex:
Latenstidskrav
Genomströmningskrav för olika stora datamängder
Tillgänglighetskrav
Säkerhetsnivåkrav (här menas framförallt intrångsskydd etc.)
Aktualitetskrav (hur färskt datat behöver vara)
Asynkron eller synkron kommunikation
Krav på transaktionshantering (ACID, kö med leveransskydd, Långa transaktioner etc.)

Det bör noteras att de stora SLA-frågorna (Service Level Agreement) såsom tillgänglighetskrav, volymer/frekvenser/svarstider i webbapplikationene etc inte täcks här, de bör specificeras av varje användande kommun enligt respektive behovsanalys. Vissa egenskaper i kommunikationsprofilerna kan tänkas gränsa till SLA-parametrar, men kommunikationsprofilerna ska istället i sådana fall ses som att mer ungefärligt ge en inriktning.

I denna version av ”Öppen teknisk plattform” inkluderas här av praktiska skäl några kommunikationsprofiler som initialt kan behövas för piloter. Senare bör profilerna brytas ut till ett separat förvaltat dokument.

I annat kravdokument i samband med eventuell anskaffning förtecknas exakt vilka Nyttomeddelanden som ska använda vilken kommunikationsprofil.

6.3.1. Kommunikationsprofil SBR_KPOL

Huvudsakligt användningsområde:
- Online-access mellan e-tjänst och verksamhetsapplikation, centrala register, e-leg/e-underskrift etc.

Latenstidskrav:
- Medium webb online, dvs typiskt sekund – halvminut

Genomströmningskrav/datamängder:
- Små till medium

Tillgänglighetskrav:
- Medium för e-tjänst, dvs efterliknande Infratjänstens servicenivå B, god kl 6-24 + övrig tid"best effort" 24*7 (tillåtet med planerade batch/service-fönster på efternatten t ex). Reservrutiner av manuell art ska finnas inom kommunen till e-tjänsten.

Säkerhetsnivåkrav:
- Inga större krav än normalt inom en datahall. Dock ej tillåtet gå okrypterat över publika linjer eller Internet. Viss integritets-känslig info kan transporteras, men ej kvalificerad sekretess.

Aktualitetskrav för informationen:
- Online

Asynkront/synkront:
- Synkront

Transaktionshantering (ACID):
- Nej. Felupptäcktsrutiner och felavhjälpsrutiner måste istället finnas.

6.3.2. Kommunikationsprofil SBR_KPBA

Huvudsakligt användningsområde:
- Bakgrunds-access mellan e-tjänst och centrala register

Latenstidskrav:
- Medium bakgrund, dvs typiskt minut – timme

Genomströmningskrav/datamängder:
- Medium, kan bli över 1 MB (skulle man överstiga 1 MB så måste antingen undantagshantering lösas ut eller informationen segmenteras i flera meddelanden)

Tillgänglighetskrav:
- Medium för e-tjänst, dvs efterliknande Infratjänstens servicenivå B, god kl 6-24 + övrig tid"best effort" 24*7 (tillåtet med planerade batch/service-fönster på efternatten t ex). Reservrutiner av manuell art ska finnas inom kommunen till e-tjänsten.

Säkerhetsnivåkrav:
- Inga större krav än normalt inom en datahall. Dock ej tillåtet gå okrypterat över publika linjer eller Internet. Viss integritets-känslig info kan transporteras, men ej kvalificerad sekretess.

Aktualitetskrav för informationen:
- Månadsfärskhet

Asynkront/synkront:
- Asynkront

Transaktionshantering (ACID):
- Nej. Felupptäcktsrutiner och felavhjälpsrutiner måste istället finnas.

7. INFÖRANDEPROJEKT

Huvudsakligen är införandeprojektaspekten utanför scope för ÖTP. Emellertid har det visat sig vara till nytta med en enklare grundprocedur för acceptanstest, framförallt i de fall leverantörerna inte har så mogna sådana rutiner:

7.1. Sambruks grundprocedur för acceptanstest

Acceptanstest:

- Före acceptanstest ska modultest separat för varje ingående modul ha gjorts av leverantören, med lyckat resultat (enstaka småfel av klass 3 och 4 må få finnas på publicerad restlista).
- Före acceptanstest ska också system- och sammanhangstest ha gjorts av leverantören, med lyckat resultat (enstaka småfel av klass 3 och 4 må få finnas på publicerad restlista).
- Acceptanstest utförs gentemot de funktionella kraven. Testprotokoll ska föras som justeras av både kommunen och leverantören.
- Acceptanstest utförs gentemot de icke-funktionella kraven. Testprotokoll ska föras som justeras av både kommunen och leverantören.
- Enstaka småfel av klass 3 och 4 må få finnas på restlista efter att kommunen godkänt acceptanstesten men då ska det finnas en överenskommen handlingsplan för att åtgärda dessa i närtid.
- Lösningen får inte tas i normaldrift förrän acceptanstesten godkänts av kommunen.

Varje testperson som vid tester får ett avvikande resultat från vad som förväntats, skriver direkt en felrapport, i möjligaste mån enligt mall som tillhandahålls elektroniskt.

- Felrapporten lämnas snarast till kommunens testansvarige
- Testansvarig sammanställer alla fel eller avvikelser i ett testprotokoll.
- Felet ska klassas av aktuell testperson och kommunens testansvarige, hur allvarligt de anser det vara, i en skala från 1 – 4 enligt följande klassning:

1. = Systemstoppande t ex en hängning som gör att man inte kan jobba vidare och som hindrar arbetet
2. = Allvarligt fel t ex felaktig maskinell behandling av informationen som kan få allvarliga följder
3. = Mindre allvarligt fel t ex fel som går att ”gå runt” och inte får allvarliga konsekvenser
4. = Irriterande / Skönhetsfel t ex stavfel i ledtexter .

Klassningen görs utifrån sannolikheten att felet uppstår, hur ofta det uppstår och hur många som blir drabbade när det uppstår.

- Testprotokollet skall kontinuerligt uppdateras och gås igenom tillsammans med leverantören enligt överenskommelse som beskrivs i testplanen. Avsikten är att koppla en problemägare till varje fel-/avvikelse samt en plan för åtgärd och trolig tid för när omtest ska kunna ske.

- Testprotokollet ska ligga åtkomligt för information för alla inblandade i testerna, men får enbart uppdateras av system- och testansvarig.

Dokument som ska framställas under testarbetet är följande:

Testplan	- beskriver närmare hur testerna skall genomföras
	- beskriver vem som har ansvar att testa vad och när det måste vara klart
	- beskriver samtliga testområden, deras prioritet och när de beräknas vara färdigtestade
Testfall	- beskrivning av händelser som ska testas
	- beskriver förutsättningar, vad som skrivas in i systemet och vilka funktionstangenter och liknande knappar man trycker på.
	- innehåller även förväntat resultat (testfacit).
Felrapport	- dokumentation för rapportering av fel /avvikelse från testfacit.
Testprotokoll	- sammanställning av fel som rapporterats i samtliga tester