Veckans teknikspaning

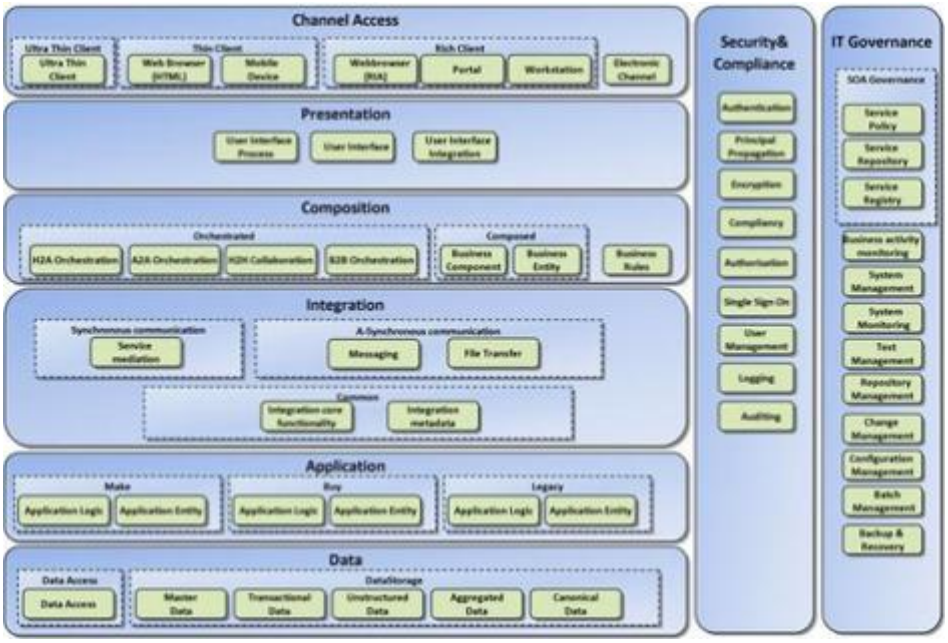
Instabilt med sammansatta tjänster?

Att sätta ihop kräver eftertanke

2012-12-21: Sven-Håkan Olsson

KOMPOSITTJÄNSTER Det är tacksamt att skapa mer avancerade tjänster genom att sätta samman ett antal enklare. Men risken är att det blir på bekostnad av både stabilitet och prestanda.

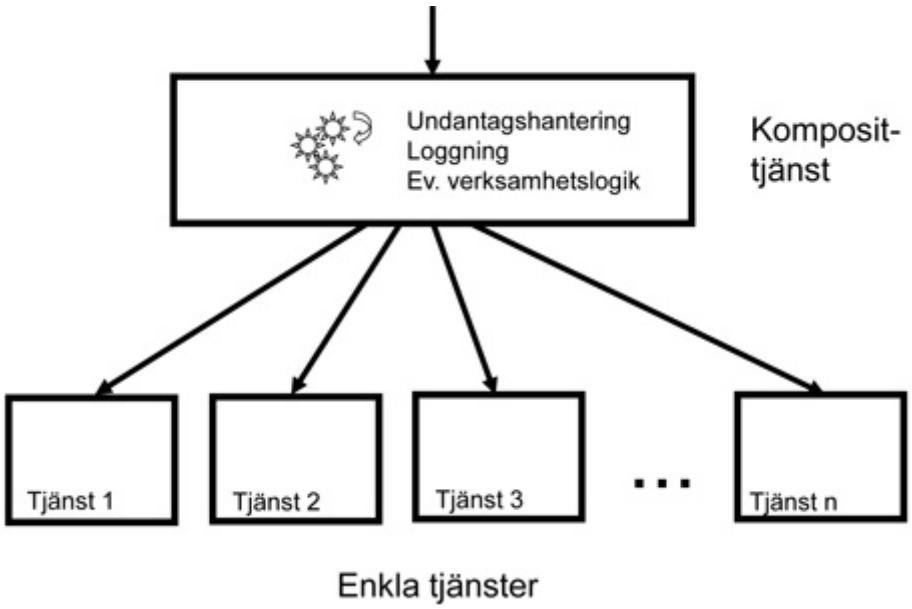
Nästan varenda referensarkitektur inkluderar en företeelse som brukar kallas sammansatta tjänster eller kompositttjänster (*composite services*). Titta exempelvis på modeller från IBM, tidigare BEA och Sun eller den oberoende CORA-modellen.



Källa: coramodel.com

Det finns en stor potential i att återanvända på en högre nivå genom att skapa kompositttjänster som utgår från enklare tjänster. Exempelvis kan man ofta hitta användningsfall där ett antal enklare tjänster alltid anropas samtidigt.

Vad vore bättre då än att slå ihop anropen så att konsumerande program bara behöver göra ett enda anrop? Det uppstår förstås också en bra möjlighet att baka in återanvändbar verksamhetslogik som styr hur de enkla tjänsterna anropas och därmed få kompositttjänsten ännu mer nyttig.



Så långt är allt gott och väl. Men jag har två invändningar:

- **Datakvalitet:** Om det sker uppdatering inom fler än en av de ingående enkla tjänsterna så får vi ett besvärligt problem med datakvaliteteten. Antingen får vi risk för halva uppdateringar eller också får vi problem med datafärskhet. Det finns mönster för att minska problemet och för att optimera, beroende på aktuella förutsättningar, men det finns ingen hundraprocentig lösning. Se gärna min genomgång i den tidigare trendspaningen Användarna förtjänar data som är korrekt.
- **Stabilitet och prestanda:** När antalet ingående tjänster i en kompositttjänst ökar, så försämras stabilitet och prestanda. Resten av artikeln ägnas åt just denna invändning.

Dålig stabilitet

Då en kompositttjänst endast behöver anropa några få enkla tjänster så påverkas inte stabiliteten nämnvärt. Men om man har haft god nytta av en kompositttjänst inkluderande få tjänster så det är lätt hänt att man går vidare. Och helt plötsligt har man skapat en kompositttjänst som behöver anropa tiotalet ingående tjänster.

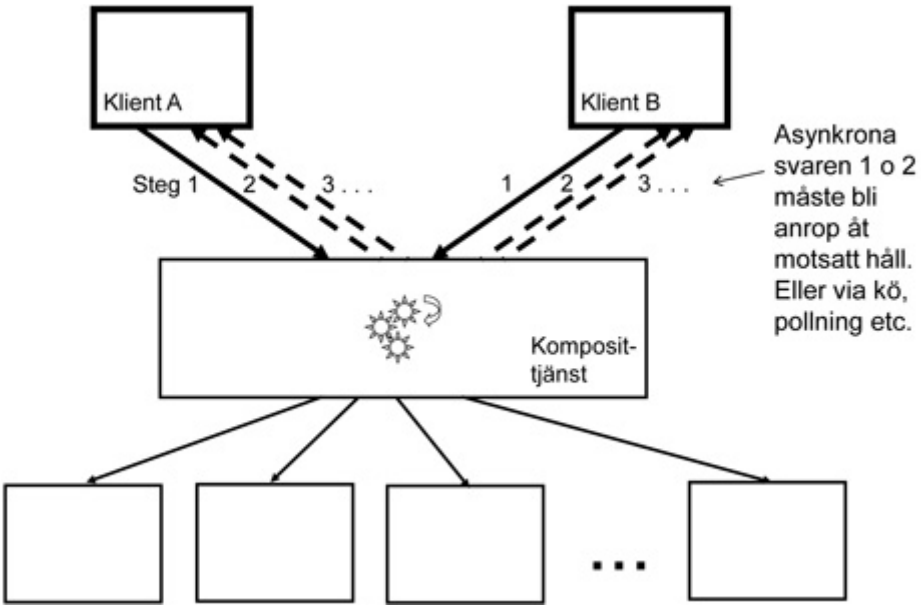
Här är sannolikhetsläran obeveklig. Förenklat uttryckt så måste man multiplicera ihop sannolikheterna för att vardera av de enkla tjänsterna är igång för att få fram hur stabilt kompositttjänsten som helhet beter sig.

Om vi är lite pessismistiska för räkneexemplets skull och antar tio enkla tjänster som vardera levererar 99 procents upptid (*uptime*) så måste man ta 0,99 upphöjt till tio, vilket ger förskräckande låga 90 procents upptid för kompositttjänsten. 90 procent betyder att den skulle riskera att stå still **fyra timmar** en typisk arbetsvecka!

Även om vi höjer till 99,5 procent för de ingående tjänsterna (vilket är en ganska typisk nivå för molntjänster) blir resultatet 95 procent vilket motsvarar två hela timmar under veckan. Skulle vi lägga pengar och arbete på att komma upp till 99,9 procent ger det 99 procent, vilket motsvarar **24 minuter** – inte vidare populärt bland användarna, det heller. Visst kan man komma högre än 99,9 procent men det är genuint svårt och dyrt samt många gånger blir tekniklösningarna så komplexa att den mänskliga faktorn mm tar över som upptids-risk. Läs gärna min trendspaning När hög tillgänglighet inte blir hög.

Asynkron lösning

Ovan har vi förutsatt ett online-scenario där kompositttjänsten gör synkrona anrop till de ingående tjänsterna och därefter synkront levererar det sammansatta svaret. Den typen av programmering är enkel att utföra och lätt att förstå samt ger en tämligen enkel undantagshantering. Men synkrona anrop ger en tät koppling som resulterar i den multiplikation av upptidssannolikhet som beskrivs ovan.



Så hur gör vi då? En lösning blir att leverera svar från kompositttjänsten asynkront. Tekniskt är det mycket mer komplext både vad gäller teknikinfrastruktur och programmeringsmönster i klient, app eller webb. I förstone kan det se omöjligt ut; till många klientmiljöer finns det av säkerhetsskäl ingen ”back-kanal” (den skulle öka risken för attacker).

Kanske måste klienten polla kompositttjänsten för att få de olika delsvaren − men pollning förbrukar tid och bandbredd samt ökar last. Eller också inför man en kö-lösning som visserligen kan vara elegant men ökar komplexiteten. I app-världen finns förvisso vissa notifierings-tjänster som kan användas, men som ger ett beroende till en central tredjepartsaktör.

Det kanske blir så att det asynkrona istället behöver ligga i klienten så att den kan presentera informationen allteftersom den anländer från de olika källorna. Användaren kan då börja titta på det som först svarades och sedan fortsätta med det som kommit senare. Eller så är man kanske är nöjd med det första datat denna gång.

Jag har byggt sådana sammansatta lösningar och de har visat sig mycket stabila. Men då kanske inte kompositttjänsten kan ligga i ett separat mellanskikt utan behöver exekvera inom klientmiljön. Det ger i sin tur andra problem såsom komponentdistributions-trassel och inkompatibilitet mellan språkmiljöer som Java och Dotnet. Fjärrmässigt är det alltså i det här fallet de enkla tjänsterna som anropas direkt.

Asynkron leverans är dock inte av så stor nytta om anropen till de olika ingående tjänsterna bygger på varandra, eller om det sammansatta svaret slås ihop från de enkla tjänsterna innan resultatet blir meningsfullt. Då blir det i alla fall den trista sannolikhetsmultipliceringen som gäller.

Ett annat sätt att öka kompositttjänstens upptid är att ha relativt korta tidsgränser (*timeouts*) i anropen till de enkla tjänsterna. Då får konsumenten i alla fall svar efter

en stund, även om allt data kanske inte finns med. Detta fungerar förstås inte om anropen till de olika enkla tjänsterna bygger på varandra. Om svarstiden hos de enkla tjänsterna dessutom fluktuerar mycket, riskerar man att inte få det data man skulle kunnat få, eftersom en kort tidsgräns redan har löst ut.

Problemet med sammansatt *prestanda* liknar problemet med upptid om man använder synkron leverans respektive tjänsteanrop som bygger på varandra – eftersom svarstiderna adderas. Skulle kompositttjänsten dessutom innehålla en loop kan svarstiderna gå upp betänkligt.

Annan lagringssamverkan

Ett helt annat sätt att tackla problemet är att inte göra de där anropen till fjärrtjänster utan att se till att datat som behövs redan finns tillgängligt nära konsumenten istället. Lösningar som replikering, avisering, händelsedriven arkitektur (*EDA, Event Driven Architecture*) etc, kan fungera väl.

Men man ska vara medveten om att priset ofta är sämre datafärskhet genom att replikeringen skedde för en stund sedan, eller att en avisering kanske endast sker en gång i veckan. Det finns också en risk att mycket stora datamängder måste lagras nära konsumenten vilket i sin tur kan leda till både stora lagringskostnader och integritetsproblem.

Sammanfattning

Sammanfattningsvis kan man säga att kompositttjänster många gånger kan vara kraftfulla och ge utmärkt nytta – men inte i alla lägen. Framförallt bör man vara mycket försiktig om ett stort antal underliggande tjänster ska anropas samtidigt. Och asynkrona lösningar kan vara nyttiga men är samtidigt svårare att skapa.

Sven-Håkan Olsson sysslar just nu med en läsplattetjänst för bolagsstyrelser och nämnder. I övrigt är han oberoende konsult som särskilt arbetar med att kombinera verksamhetsnytta med teknikhöjd. Han har en lång karriär bakom sig sedan 70-talet som it-konsult (applikationsarkitektur, systemdesign, programmering, reviewer, utredningar, kursledning). Sven-Håkan är medgrundare till KnowIT där han också var teknikchef 1990 - 2003. Han utsågs till en av "Sveriges topputvecklare" av Computer Sweden. Sven-Håkan håller regelbundet kurser åt

Dataföreningen. Läs gärna mer på hans blogg definitivus.se samt på styrelsemote.se.

Sven-Håkan Olsson