

Kan SOA gå på vandet?

Forfatter: Søren Lauesen, professor ved ITU

Serviceorienteret arkitektur (SOA) har vist sig nyttig til at opbygge nye systemer der trækker på data i store, gamle systemer. Programmørerne udvider de gamle systemer med et antal services der kan overføre data, og de nye systemer trækker på disse services. De der har prøvet det, ved at det ikke er smertefrit, men alligevel sikrere end den traditionelle løsning hvor nye systemer udføre SQL-sætninger direkte på de gamle databaser, som kun få tør røre ved.

Så langt så godt. Problemet kommer når SOA udråbes som fremtidens mirakel. Alt skal gøres serviceorienteret. Man skal opbygge alle systemer som moduler der hver tilbyder et antal forretningsorienterede services. Så kan modulerne let sættes sammen til nye løsninger, siges det. SQL-sætningerne skal skjules bag services. Det lyder jo forjættende, men som med så meget andet i it-verdenen, går det ikke helt som præsten prædiker. I praksis bliver der fx hele tiden behov for nye services i de eksisterende moduler, og det er langt mere besværligt at tilføje en ny service end at skrive en ny SQL-sætning.

Jeg vil illustrere problemerne med to eksempler fra virksomheder med store SOA løsninger (navnene er desværre fortrolige). Virksomhederne har typisk opbygget en SOA løsning bestående af 5-10 systemer (moduler) koblet sammen med services. Hvert modul tilbyder 30-40 serviceoperationer, som er defineret i et tæt samarbejde med dem der står for de andre moduler. Man forestiller sig at man nu har defineret alle de nødvendige serviceoperationer - man har jo tænkt sig grundigt om.

Eksempel 1: Virksomheden begynder at opbygge en ny anvendelse med nye skærbilleder. Det går ikke så let, for de eksisterende services passer alligevel ikke rigtigt, og man forhandler flittigt om hvem der har skylden. Er services defineret forkert eller kan de nye udviklere ikke finde ud af at bruge dem? Problemet stammer ofte fra brugergrænsefladen. Et typisk skærbillede bruger flere SQL-sætninger: en til hovedet af billedet, en til en tabel i billedet, et par stykker til drop-down lister. Og behovet varierer fra billede til billede. Disse SQL-sætninger svarer sjældent til de eksisterende services. Skal man holde sig til dem, får man enten dårlig brugerstøtte, eller også må den nye anvendelse overføre en masse overflødig data og selv uddrage det relevante (såkaldte client-level joins). Her har djævlens spillet os et puds ved at forbinde brugergrænsefladens detaljer med databasens detaljer - tværs over arkitekternes forretningsorienterede services. Den magi der ligger i en relationsdatabase kan ikke udføres på klient-siden.

Eksempel 2: Virksomheden vil købe et nyt standardsystem som skal kobles sammen med de eksisterende. Hvad kræver det af leverandøren? Det nye standardsystem har måske 500 SQL-sætninger og stored procedures der trækker på leverandørens egen database. De skal nu erstattes med kald af nogle services, men igen er der ikke lige

den rigtige service. Enten må man igen tilføje services eller lave client-level join. Men ligegyldigt hvilken løsning man vælger, er det ikke længere et standardssystem, så det skal vedligeholdes for sig selv og vil koste en formue. Det er fordi standardsystemet ikke er SOA-baseret, siger profeterne. Ak, nej. Dets services ville sandsynligvis være inkompatible med kundens og resultatet ville blive det samme.

Leverandøren vil evt. foreslå at hans system replikerer virksomhedens data og på passende tidspunkter synkroniserer de to databaser. Det kan han lettere lave som et tillægsmodul der ikke griber dybt ind i standardsystemet. Desværre er SOA-idealet nu revnet og dataaktualiteten ikke optimal, men det kan alligevel være den bedste løsning.

Nej, heller ikke SOA kan gå på vandet. Man skal omhyggeligt bygge bro hver gang man skal til en ny ø. Brug kun SOA hvor en traditionel database-løsning er umulig. Er SOA uomgængeligt, så gør det i det mindste let at tilføje nye services, og accepter lidt datareplikering.