To SaaS or not to SaaS

Hieperdepiep. We hebben er weer een hype bij. SOA is nog zeker niet uit, ESB wat Microsoft betreft wel, maar het werd toch weer tijd voor iets nieuws. En dat is SaaS: Software as a Service. Is het oude wijn in nieuwe zakken? Deels wel, deels niet, is het te verwachten diplomatieke antwoord. In elk geval bekt SaaS lekker weg; en dat is al een pre ten opzichte van de eerdere 'service'-hypes.

Wat is SaaS? Voor een goede omschrijving kun je tegenwoordig niet meer om wikipedia heen. Daar wordt SaaS als volgt omschreven: "a model of software delivery where the software company provides maintenance, daily technical operation, and support for the software provided to their client".
Wie goed op de definitie let en al acht jaar of langer in het vak zit, moet bij het lezen wel een "déjà vu" hebben gehad. En inderdaad: SaaS lijkt erg op het ASP-model van het einde van de vorige eeuw. ASP als in Microsoft Active Server Pages? Nee, ASP als in Application Service Provider. Het ging daarbij om applicaties 'uit de muur' die veel sneller en efficiënter konden worden geboden door gespecialiseerde ASP's dan dat je als bedrijf die spullen zelf zou gaan installeren en beheren. Het was het begin van het tijdperk van ICT-outsourcing. En daar waar het uitbesteden van de ICT-ontwikkeling en de ICT-operaties later een grote vlucht hebben genomen, is het ASP-model gestruikeld. De ASP-hype had zijn hoogtepunt in 1999 toen er alleen al in de VS zo'n slordige 3,6 miljard dollar in geïnvesteerd werd en er niet minder dan 2000 bedrijven in verschillende ASP-rollen actief waren. Dat was een paar jaar later zo goed als voorbij. ASP kon niet doen wat het beloofde. Gaat SaaS het dan wel redden?

Wat ging er fout bij ASP? Het is een mix van vooral twee zaken: ten eerste, de door de ASP aangeboden applicaties waren qua architectuur in de regel niet geschikt voor dat werk en ten tweede, de staat van de technologie en de 'mindset' van de architect waren er nog niet klaar voor. Ik zal beide punten nader toelichten.

De ASP-ers hadden haast om hun businessmodel op de markt te zetten en vergaten te kijken naar de geschiktheid van de aangeboden applicaties om uit de muur door meer klanten tegelijk geconsumeerd te kunnen worden. Vaak werden bestaande twee/drielagen client/server-applicaties gebruikt; applicaties die gemaakt waren om binnen een enkel 'trust'-domein en een LAN-omgeving te gebruiken. En die werden nu van een webbrowser-laagje voorzien en zo aan de klanten aangeboden. Als het al werkte, dan ging het zeer traag en moest de ASP voor elke andere klant een nieuwe instantie van software/hardware installeren: weg schaalvoordeel. Ook wilden gebruikers vaak aanpassingen aan de aangeboden applicatie en dat betekende veranderingen in de broncode en dus het onderhouden van meer codebases: een ware nachtmerrie. Als men het geheel dan toch aan het werk kreeg, kwam de volgende tegenvaller om de hoek zetten. De klant wilde de ASP-applicatie integreren met de applicaties die hij al in huis had. Als het integreren van applicaties in het eigen ICT-landschap al lastig was, schier onmogelijk werd het om de elders gehoste en met anderen gedeelde applicaties te integreren.

Het tweede punt sluit hierop aan. De staat van de technologie was nog niet zover dat integratie of samenwerking tussen applicaties goed te realiseren was; zeker niet volgens goede architectuurprincipes. De afgelopen paar jaren heeft het SOA-gedachtegoed echter flink om zich heen gegrepen. Ik heb in deze kolom verscheidene malen over dit boeiende onderwerp geschreven. Nu is de staat van de technologie er wel rijp voor - bijvoorbeeld via webservices - en snapt de architect ook beter waarop hij of zij moet letten bij het uitbesteden van softwarecapaciteiten die nodig zijn om de business te ondersteunen. Immers, als je een bepaalde softwarecapaciteit uitbesteedt, moet je vooral ook letten op de integratiepunten met wat je nog wel binnen houdt. Want je moet als enterprise ICT-architect wel het totale ICT-landschap in de gaten houden. Ik denk dat ICT-architecten dit inmiddels een stuk beter snappen dan een jaar of zeven geleden.

Maar wat is er dan anders aan de SaaS-applicaties zelf? SaaS-applicaties worden in de regel nu goed volgens de architectuurprincipes van serviceoriëntatie ontworpen: een smart en/of thin cliënt-deel dat tegen een set van processervices aanpraat, die op zijn beurt weer onderliggende shared businessservices aanroept. Dat betekent dat de SaaS-applicaties bij uitstek geschikt zijn om over het internet te werken. Maar dat is niet genoeg om een dergelijke applicatie ook aan meer klanten tegelijk aan te bieden. Daarvoor is meer nodig. Ik noem hier slechts kort drie belangrijke aspecten.

Ten eerste moet een goede SaaS-applicatie *configureerbaar* zijn, dat wil zeggen dat de klant – de SaaS-afnemer – de applicatie naar zijn smaak moet kunnen aanpassen zonder dat de broncode daarvoor moet worden aangepast: op het vlak van de gebruikersinterface, in de inrichting van de bedrijfsregels en qua gegevensstructuur. Ten tweede moet een goede SaaS-applicatie meer klanten naast elkaar in dezelfde applicatie-instantie kunnen huisvesten. Lukt dat namelijk niet, dan moet de SaaS-provider voor elke klant een aparte installatie doen en dan gaat het schaalvoordeel grotendeels verloren. Dan wordt het allemaal een stuk duurder en daarmee verdwijnt een belangrijk voordeel voor de klant. Ten derde moet de SaaS-applicatie geweldig goed kunnen opschalen, want het succes staat of valt met het aantal gebruikers. En dat kunnen er zomaar een paar miljoen worden. Je komt er alleen maar met beproefde schalingontwerpprincipes, zoals 'scale out' met load balancing, een stateless ontwerp van de applicatieserver, asynchrone I/O met interrupts, het delen van dure resources als threads en databaseconnecties en het gebruik van optimistic locking. Dat alles hebben we geleerd van het falende ASP-model. SaaS is dus in wezen niets anders dan ASP 2.0.

SaaS is hot en je kunt je er snel in verliezen, net zoals bij SOA en ESB. Pas dus goed op en zorg dat je het juiste abstractieniveau behoudt. Zoals je gemerkt hebt, heb ik hier vrijwel uitsluitend gesproken over het SaaS-paradigma en de applicatiearchitectuur. Dat je met de webservicesarchitectuur (ws-*) en met het .NET Framework prachtige SaaS-applicaties kunt bouwen, is een conclusie die je pas mag trekken als je eerst weet wat een goede SaaS-applicatie moet bevatten. En hopelijk heeft deze kolom daar een beetje aan toe bijgedragen.

Dik Bijl
Architect Advisor Microsoft Nederland