Methodology

IT-architectuur is in een jong vakgebied een nog jonger vakgebied, realiseert Cor Baars zich maar al te goed. Hij is al 22 jaar werkzaam in de IT en heeft zich de afgelopen jaren vooral beziggehouden met enterprise architectuur voor grotere organisaties. De afgelopen zes jaar was hij ook hoofd van de Master of Science opleiding IT Architecture bij DNV-Cibit. Hij heeft stevige en eigengereide ideeën over de 'nieuwe IT', die op heel andere uitgangspunten is gegrondvest dan de IT van weleer. Reden genoeg om eens naar hem te gaan luisteren op een door de nieuwe werkgever van Baars, Sogyo IT, georganiseerd seminar voor jonge IT-architecten.

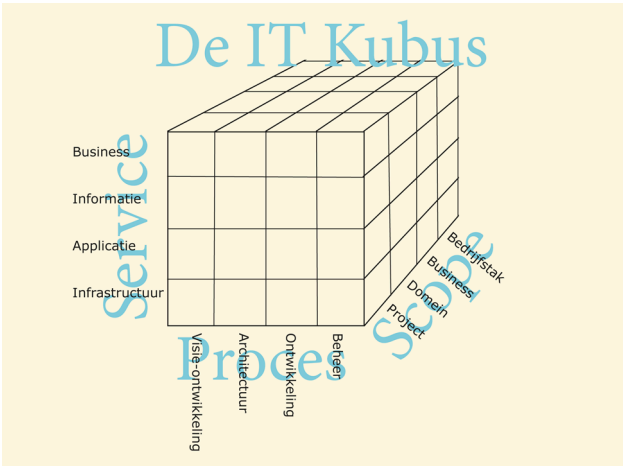
Wok-kok geeft SOA structuur

IT-kubus houdt business en IT 'aligned'

Tussen waterval en het nu in opkomst zijnde Event Driven Architecture is de afgelopen dertig jaar veel gebeurd. Er zijn veel fouten gemaakt en er worden nog steeds veel fouten gemaakt. Hierdoor is een grote hoeveelheid niet optimaal functionerende legacy opgebouwd, die je niet zomaar kunt laten verdwijnen. De oplossing moet langs twee wegen worden gevonden: verbeteren van bestaande software en voorkomen van fouten in nieuwe software. Cor Baars ziet goede mogelijkheden in SOA om beide problemen tegelijkertijd aan te pakken. Mits dat goed wordt toegepast.

Hij heeft een IT-kubus ontwikkeld, die er voor kan zorgen dat in zowel nieuwe als bestaande softwareprojecten alle neuzen in dezelfde richting wijzen. In deze kubus zijn de business-IT alignment, de tijdlijn en de scope van een project ondergebracht (zie figuur 1). Tweedimensionale modellen bestonden al, maar Baars heeft er de derde dimensie, de scope, aan toegevoegd. Op deze scope-as wordt de omvang van het proces weergegeven in de lagen project – domein – business – bedrijfstak.

Nadenken over architectuur kun je in verschillende scopes doen. Je kunt dat op projectniveau doen, op domeinniveau – bijvoorbeeld hypotheken bij een bank of schadeverzekering bij een verzekeringmaatschappij. Nog een stap hoger liggen de projecten op businessniveau en – heel stoer



Figuur 1: De IT-kubus van Cor Baars.

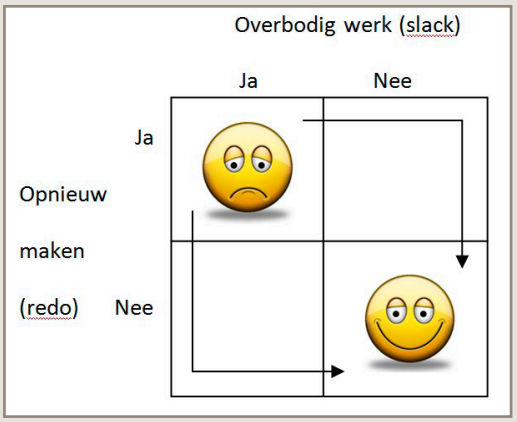
– architectuur ontwikkelen voor een hele bedrijfstak of de overheid.

De stelling van Cor Baars is dat je de kubus niet uit elkaar mag rekken. Dat de diverse activiteiten aligned moeten blijven. Het gaat fout met de infrastructuur van het project als de business er niet bij wordt betrokken. En als het proces te veel tijd kost, kan het zijn dat wanneer het project in de beheerfase komt, de visie inmiddels sterk is gewijzigd en je weer van voren af aan kunt beginnen. Het uittrekken van de kubus in de diepte veroorzaakt eilandvorming en het niet toepassen van

Robert de Ruiter
is hoofdredacteur
van Release.

Slack of redo?

Als de schijven in de IT-kubus horizontaal of verticaal of in de diepte uit elkaar worden getrokken dreigt het ontwikkelproces uit de hand te lopen. Six Sigma gaat er vanuit dat je het maar beter meteen goed kunt doen (geen redo's, omdat het herstellen van fouten vele malen duurder is dan meer kwaliteit stoppen in het initiële ontwikkelproces. Lean daarentegen probeert zoveel mogelijk 'waste' uit een proces te halen (geen slack). Het herstellen van een fout kan tot wel vijftig keer zoveel kosten dan wanneer het werk in een keer goed wordt gedaan. Lean daarentegen probeert zoveel mogelijk 'waste' uit een proces te halen (geen slack). Slack die wordt veroorzaakt door het overbruggen van de overgang van de business naar de ontwikkeling en van de ontwikkeling naar infrastructuur en beheer. Je kunt je daar de afgebeelde matrix bij voorstellen.



Het is wel duidelijk dat het vak linksboven niet in orde is. Dat is een ontwikkelproces, waar én heel veel werk is gaan zitten én dat we ook niet goed doen, want er moet veel aan worden gerepareerd. Dat telt dus dubbel in de min en is daarmee vaak de oorzaak dat men zich niet voor kan stellen dat er iteratief wordt ontwikkeld omdat men dan niet 1 keer maar meerdere keren de zojuist genoemde overhead tegenkomt. Als je in deze situatie belandt, is het goed om na te denken hoe je die oplost. Iedereen wil het liefst via de rechte lijn van linksboven naar rechtsonder, maar dat is niet erg realistisch. Hoe los je dit dan het beste op? Ga je eerst kijken hoe je de overbodige zaken eruit kunt halen of ga je nieuwe requirements schrijven om het opnieuw en goed te doen. In de meeste gevallen zal men proberen om eerst het specificatieproces op orde te brengen, zodat in het vervolg wordt gebouwd wat bedoeld is. Het verminderen van de 'slack', de overbodige elementen, is vaak lastiger, omdat dit met veel afdelingen en organisatiebrede zaken te maken heeft.

bedrijfstak- bedrijfsbrede of domeinspecifieke principes en standaarden.

Twee fouten

Over het algemeen kun je twee dingen fout doen: je kunt foute functionaliteit maken, maar dat is meestal goed te voorkomen. Je kunt ook goede functionaliteit maken die niet aan de kwaliteitseisen voldoet. Grote organisaties maken vaak hele mooie software, maar soms is die onvoldoende beschikbaar of schaalbaar. Stel je ontwerpt een prachtig programma voor helpdeskmedewerkers van een organisatie. In de praktijk blijkt het niet te werken boven de vijftien gebruikers, maar de organisatie heeft 300 helpdeskmedewerkers in dienst… Dan is duidelijk een gat ontstaan tussen de inframensen en de ontwikkelaars. Waarmee niet gezegd wil zijn dat de inframensen hun werk niet goed hebben gedaan, want ook de ontwikkelaar kan heel veel aan schaalbaarheid bijdragen. Sterker nog, het is gemakkelijker om de schaalbaarheid en de flexibiliteti van een applicatie als ontwerper/ programmeur te verpesten dan als infraarchitect.

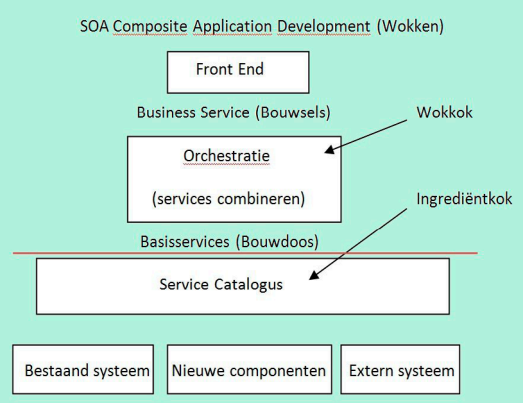
Het doel van de IT-kubus is duidelijk te maken dat een individu in het ontwikkelproces altijd in zes richtingen moet 'alignen': boven (business), beneden (infrastructur), links (visie), rechts (operatie), voor (project) enachter (bedrijf of bedrijfstak),. Als je rol in de kubus is bepaald, is duidelijk waar je de informatie vandaan haalt en naar wie je terug moet koppelen.

Wokken

Belangrijk is dat voor het ontwikkelen je voldoende inzicht krijgt in de context en de beschikbare (half) fabrikaten. Cor Baars heeft hiervoor een metafoor bedacht rondom een Wokrestaurant. In zo'n restaurant zijn de ingrediënten heel belangrijk. Je hebt daar dus speciale koks, die de ingrediënten klaarzetten. Daarnaast heb je de wok-kok, die agile werkt, de communicatie met de klant verzorgt en deze naar wens kan bedienen. Dit model past goed in een SOA-ontwikkelstraat met een divers team. De wok-kok bedenkt daarbij welke ingrediënten hij nodig heeft om zijn gerecht – het op te leveren product – te kunnen maken. Vervolgens gaan de ingrediëntkoks aan de slag om hun bakken met salades, vlees, vis, groenten en dergelijke – de services – klaar te zetten. Hiermee wordt het probleem dat de 'voorkanters' op de 'achterkanters' moeten wachten voorkomen. In veel ontwikkelprocessen zit de back-end op de specificaties van de frontend te wachten, terwijl later in het proces de frontend moet wachten op de producten van de backend.

In het wokrestaurant van Baars kunnen de ingrediënten ook worden gefaked, zodat de wok-kok kan doorwerken. Dat is vervelend, omdat de use cases op enig moment kunnen zijn afgewerkt, maar omdat bepaalde services ontbreken het programma niet – of niet geheel – kan worden opgeleverd. Het is dan ook zaak om voldoende aandacht aan de ingrediëntkoks

Het maken van half-fabrikaten is wezenlijker, omdat deze in meerdere projecten worden gebruikt.



De plaats van de koks bij application development.

te geven, omdat hun werk belangrijker is dan dat van de wok-kok. In de metafoor van Baars wordt dat zo uitgelegd: "Als de wok-kok een fout maakt, zit er één klant op de wc, maar als de bak mosselen van de ingrediëntkok niet goed is zit het hele restaurant op de wc".

Baars: "Het wordt vaak niet beseft, maar als je halffabrikaten maakt is dat veel wezenlijker, omdat deze in verschillende projecten worden gebruikt. Ik heb meegemaakt dat voor een vijftal projecten werd besloten om de autorisatie niet zelf te ontwikkelen, omdat binnen drie maanden een kant-en-klare autorisatieservice uit zou komen. Die functiepunten werden uit de projecten geschrapt. Maar die oplossing kwam pas na een half jaar. Toen lagen er dus vijf projecten stil. Veel projectleiders hebben hieruit de conclusie getrokken dat zij iets met SOA moesten gaan doen om dit soort problemen te vermijden."

In eigen beheer iets ontwikkelen werkt veel sneller dan kant-en-klare producten afnemen bij bestaande leveranciers.

Ombouwen

Een restaurant tot wokrestaurant ombouwen is niet eenvoudig. Je moet ingrediënten hebben. Je ziet dat de bedrijven, die succesvol zijn met SOA, veelal met een gekochte verzameling ingrediënten zijn begonnen en niet eerst stapje voor stapje hun ingrediëntenpakket hebben uitgebreid. Je kunt bijvoorbeeld je eigen frontend ontwikkelen en voor de businesslogica gebruik maken van bijvoorbeeld SAP. Je kunt wel kiezen voor een stapsgewijze benadering, maar dan haal je één voor één delen uit de keuken van het restaurant. Je kunt bijvoorbeeld eerst een saladebar inrichten en als dat is gebeurd stap je over op wat de Britten 'Carving' noemen: je zet het lam, de ham en de biefstuk met een kok erbij in het restaurant en laat dit in het restaurant snijden. Later ontwikkel je dit tot self-service en dan is het vlees de keuken uit.

Er ontstaat dankzij de ontwikkelingen rond de cloud trouwens ook een enorme markt voor halffabrikaten, waardoor je bepaalde delen van het proces relatief snel kunt ombouwen. Leveranciers als Salesforce, SAP, Oracle (AIA-services) bieden voor de backend prima oplossingen aan. Daar zal de komende tijd ook intensief gebruik van worden gemaakt, verwacht Baars. "Maar je moet als architect natuurlijk wel goed kijken of de backend die je wilt aanschaffen past bij de organisatie waar je deze wilt implementeren. Aan de andere kant kun je ook zeggen dat binnen een bepaald kostenplaatje nu eenmaal niet alles bereikbaar is. Het is een beetje flauw om in een wokrestaurant te roepen dat er geen Jacobsschelpen zijn. Als je die perse wilt, moet je naar dat veel duurdere wokrestaurant om de hoek."

Proces

Het SOA ontwikkelproces ziet er in de ogen van Baars uit als in figuur 2. Aangegeven is waar de activiteiten van de koks plaatsvinden.
In de ideale situatie zou het onderscheid tussen business en IT bij de rode lijn moeten liggen. Dat de business dat kleine beetje techniek wordt bijgebracht, zodat ze zelf kunnen wokken zonder van de IT afhankelijk te zijn. De echte IT-ers zorgen dan dat de services beschikbaar komen. De keus is dan tussen zelf ontwikkelen of externe systemen van bestaande leveranciers afnemen. Baars ziet in de laatste optie grote voordelen, mede omdat sneller kan worden gewerkt dan wanneer je alles van scratch af gaat ontwikkelen.

Samenstellen van de Service Catalogus is overigens een kunst op zich. Baars heeft tijdens een workshop bij een gemeente een aantal domeindeskundigen (architecten) in een zaal gezet en hen gevraagd om in een dag tijd de benodigde backend-services te bedenken waarmee vier use cases - trouwen, scheiden, geboren worden en overlijden – konden worden gerealiseerd. Aan het eind van de dag hadden zij 25 services bedacht. Sommige daarvan kwamen in één use case voor; andere werden in alle vier use cases gebruikt. Die exercitie is later nog eens op grotere schaal overgedaan. Toen rolden er 80 services uit de bus, waarmee 80 procent van de burgerzaken konden worden geautomatiseerd.

"We hebben dan dus nog geen geld uitgegeven, maar met pen en papier bedacht hoe de ideale catalogus eruit zou zien. Dat geeft mooi richting aan wat je moet produceren. Vervelend was wel dat de services moesten voortkomen uit de bestaande systemen. Want er zat een enorm verschil tussen wat je kon maken met de backend van nu en wat je wilde realiseren. Je kunt echter van hieruit wel proberen stapje voor stapje dichter bij het ideaal te komen door onderdelen in de bestaande catalogus te vervangen." Je hebt een punt op de horizon en dat maakt de afweging of bepaalde services die uit projecten ontstaan eventueel tot domeinniveau of bedrijfsniveau moeten worden opgeschaald eenvoudiger en doelgerichter.

Risico's

Het grootste risico van SOA ontwikkeling is dat op enig moment onvoldoende services beschikbaar zijn, zoals wanneer een serviceleverancier platgaat. Daarbij komt nog dat dat des te erger is in het geval van een service die veel in applicaties wordt gebruikt: het is in een wokrestaurant erger wanneer een zeer geliefd ingredient op is dan een ingredient waar toch al niet zoveel vraag naar was.

Maar er zijn meer valkuilen, zo erkent Baars. De harmonisatie van de gegevens is daar een van. SOA is geen Esperanto, dus de services moeten wel op elkaar zijn afgestemd. Het eigenaarschap vormt ook nog wel eens een punt van discussie. Van wie zijn de services? En dan is er nog de kwestie van vraag en aanbod. Er bestaat nu eenmaal geen gouden gids, waaruit je maar van alles kunt bestellen.

Tot slot: de performance. Zoals overal elders in de IT is dit ook bij een SOA een 'long and winding road'.

Als SOA zoveel risico's inhoudt, waarom zou je er dan aan beginnen? Cor Baars zegt daar gekscherend op: "Omdat iedereen het doet." "Maar", voegt hij eraan toe, "SOA heeft ook veel voordelen. Als de ingrediënten zowel functioneel als kwalitatief beschikbaar zijn en als de wokkok in het SOA-restaurant zijn werk goed doet, is het proces snel, interactief, communicatief, klantgericht en (beperkt) aanpasbaar."

Tot slot

Tot slot nog een reden waarom hij kiest voor SOA: "Omdat Event Driven Architecture nog gevaarlijker is dan SOA. Je weet bij EDA niet waar de events landen, hoe ze worden geïnterpreteerd en wanneer ze worden verwerkt. Met een beetje pech krijg je op grote schaal hetzelfde probleem van de 'flash crash' die enige tijd geleden het gevolg was van een typefout in een beurstransactie en de onmiddelijke reactie van de geautomatiseerde beurshandelssystemen. Vergt SOA al veel coordinatie, dat geldt nog meer voor EDA." «

'Event Driven Architecture is nog veel gevaarlijker dan SOA.'

Cor Baars.