Enterprise

Je moet een beetje lef hebben onder deze titel een artikel te schrijven als de succesvolle SOA implementaties nog op één hand te tellen zijn. Als ervaringsdeskundigen binnen IT-eye van een behoorlijk aantal SOA-projecten (volledig in productie), hebben we het toch maar gedaan. Dat we die projecten realiseerden op basis van de Oracle SOA Suite 10.1.3 is weliswaar relevant, maar de tien geboden zijn zoveel mogelijk neutraal opgesteld.

De tien geboden

Richtlijnen voor een succesvol SOA project

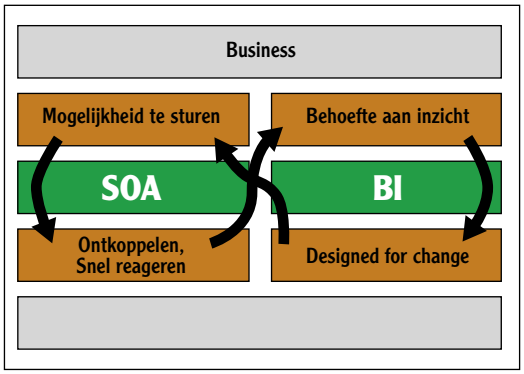
In de aanloop naar onze Top10 van geboden zijn er ruim dertig direct relevante onderwerpen aan de orde geweest. De discussies over de succesfactoren voor een SOA-project waren talrijk en leerzaam. We hebben gewikt en gewogen en uiteindelijk is een tiental geboden met stemrondes geselecteerd. Opvallend is dat relatief weinig geboden puur technisch zijn. Veel aandacht gaat uit naar de implicaties van de mindset die het toepassen van een service georiënteerd architectuur met zich meebrengt. De tien geboden staan niet op zichzelf maar moeten gezamenlijk bezien worden. Duidelijk moet zijn dat in een SOA omgeving nog steeds 'gewoon software wordt gemaakt' maar dat een aantal ingesleten patronen van de afgelopen jaren niet meer toereikend zijn. Het vermogen om goed op de veranderingen in te spelen is het meest bepalend of een project succesvol kan worden uitgevoerd. Om de leesbaarheid te vergroten zijn de tien geboden gegroepeerd rondom Scope, Architectuur, (Project)organisatie en Implementatie. En als een verborgen elfde gebod om vooral *agile* te werk te gaan hebben alle geboden van ons de kortst mogelijke naam gekregen.

Scope

1. *SOA needs BI*

Service oriëntatie gaat uit van communicerende services die zijn gemodelleerd als bedrijfsproces. Services worden generiek ontworpen zodat hergebruik in verschillende bedrijfsprocessen mogelijk is. Het resultaat is flexibiliteit om bedrijfsprocessen snel en goedkoop te kunnen veranderen. *Business Intelligence* is het ontsluiten, toegankelijk en beheersbaar maken van data en informatie die voortkomt uit de bedrijfsvoering. Het resultaat is continu inzicht in de operationele resultaten van *bedrijfsprocessen*.

In beide definities is bedrijfsprocessen het sleutelelement. BI verschaft inzicht in de resultaten van bedrijfsprocessen. De vaak apart genoemde BI verschijningsvorm Business Activity Monitoring (BAM) doet hetzelfde maar legt de focus op de procesuitvoering zelf. Door BAM en BI vroegtijdig onderdeel te maken van een SOA project wordt makkelijker ingespeeld op de informatiewensen van de gebruikers en wordt vooral het inzicht in de bedrijfsprocessen verbeterd. Acceptatiecriteria zijn onderdeel van de analyse en worden geïmplementeerd in meetinstrumenten. De meerwaarde geldt niet alleen ná oplevering van een project. BI levert al tijdens de ontwikkelingsfase een bijdrage in de systeemont-



Figuur 1

Jules de Ruijter
is managing consultant bij IT-eye

wikkeling, doordat het een andere blik biedt op de te ontwikkelen systemen en op het proces van systeemontwikkeling. Tot slot: waarom flexibiliteit bieden bij de ondersteuning van bedrijfsprocessen (SOA) als de resultaten niet eenduidig worden gemeten (BAM & BI)?

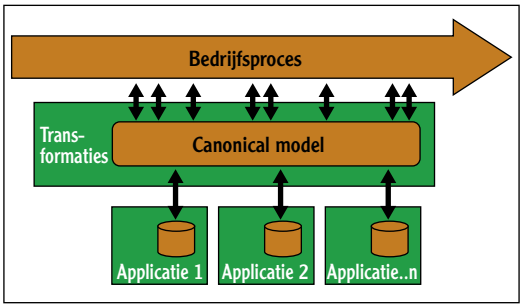
2. *Industriemodellen*
De eisen die aan een moderne automatiseringsoplossing worden gesteld zijn hoog, en wijzigen zo snel dat de markt gedwongen wordt om algemeen aanvaarde standaarden te ontwikkelen en te gebruiken. Technisch gezien is de technologiestack voor een SOA-implementatie hier een antwoord op. Hiermee worden de WS-* standaarden gevolgd. De techniek is maar een klein onderdeel van de uiteindelijke SOA-oplossing. Voor toepassing van deze technologie in de bedrijfsvoering is standaardisering van content nodig: procesdefinities en gegevensdefinities. Een groot deel van de kosten en tijd van een SOA-implementatie gaat zitten in de ontwikkeling en beheer van deze definities. Met de daarbij horende discussie tussen business en IT.
Her en der worden per *bedrijfstak* standaardmodellen voor processen en gegevens ontwikkeld. (bijvoorbeeld eTOM, HDN). Deze kun je als best-practice inzetten en naar eigen wensen customizen. Dit reduceert niet alleen de time-to-market van een oplossing, het leidt ook tot de juiste verhouding tussen business en IT: De standaard voor content wordt beheerd door de business, de standaard voor de technologie door IT.

Architectuur

3. *Architectuur*
Architectuur uitgangspunten bepalen hoe functionaliteit (voorheen een applicatie) wordt ontwikkeld. Dit is de kern van service oriented *architecture*. De architect bedenkt deze uitgangspunten in samenspraak met de business. Hierbij houd hij niet alleen rekening met de problemen van vandaag maar ook van morgen. De architectuur is 'designed for change' en biedt (tot bepaalde hoogte) vrijheidsgraden om ook niet voorziene ontwikkelingen te kunnen omarmen.
Voor veel IT-ers is werken onder een expliciete architectuur nieuw. De vorige generatie architecturen was vaak impliciet opgenomen in de ontwikkeltool of –methodiek.
Mede door de verschillende rollen binnen een project ontstaan discussies over concepten uit de klassieke wereld (bijvoorbeeld: Oracle was datadriven) en concepten de nieuwe wereld (J2EE patterns). Het is aan de architect om deze concepten in de juiste context te plaatsen en bepaalde keuzes te beargumenteren. Een architectuur hoeft niet groots en omvangrijk te zijn, een goede implementatie is veel meer bepalend voor de revenuen ervan. Want bedenk: de gerealiseerde helft bij een halfslachtig toegepaste architectuur betreft altijd de kosten, nooit de opbrengsten.

4. *Classificeer*
Er is een aantal typen software services te onderscheiden (proces-, business-, integratie- en applicatie services). Het loont zich om in een project vanaf het begin services te classificeren. Ieder type service heeft een eigen belang, complexiteit, beschrijving en lifecycle. Deze aspecten wegen mee bij het bepalen van (de schatting van) de ontwikkel- en beheerkosten. Verder geeft het in combinatie met de gegevens over beschikbare resources inzicht in de aanwezige en/of benodigde change capaciteit.
Vervolgens geeft deze classificatie duidelijkheid over communicatie en verantwoordelijkheden, vooral op het grensvlak tussen business en IT. Business services leveren de IT dienstverlening aan de business. Deze vormen dan ook de deliverables waarop je stuurt en met de business communiceert. Het is voor de business niet relevant welke services op applicatie- en integratieniveau worden gerealiseerd en beheerd. De classificatie werkt optimaal als de business zelf de verantwoordelijkheid neemt om de processen in te (laten) richten, gebruikmakend van die business services. Bereik dit door in een SOA implementatie altijd een (deel van een) bedrijfsproces in te richten.

5. *SLA*
Een service wordt vaak nog alleen gedefinieerd door zijn interface. Dit is weliswaar voldoende om een service in *technische* zin aan te kunnen spreken, maar absoluut onvoldoende om correct met de service te kunnen *werken*. Stel daarom voor iedere service een SLA (Service Level Agreement) op waarin het *gebruik* van service nader wordt beschreven.
De SLA dient de interface beschrijvend te definieren, voornamelijk ten aanzien van de functional en non-functional requirements. Aard en type SOA implementatie bepalen de inhoud van het SLA, maar al snel zijn de volgende items aan de orde:



Figuur 2

10 geboden voor een succesvol SOA project

Scope
1 SOA needs BI
2 Industriemodellen

Architectuur
3 Architectuur
4 Classificeer!
5 SLA

(Project)organisatie
6 Cross Functional Team
7 Processen
8 Integratietests

Implementatie
9 Canonical Model
10 Exception Handling

De aandacht voor de analyse van processen kan bijna niet groot genoeg zijn

- Eigenaarschap, organisatorisch
- Benodigde services, welke anere services worden benaderd voor het uitvoeren van de operatie.
- Beschikbaarheideisen.
- Performance-eisen, responsetijden.
- Rechten (autorisaties, rollen et cetera).
- Type interpretatie, bijvoorbeeld wat wordt bedoeld met een bepaalde code 1 of code 10.
- Gebruikte eenheden.
- Synchronisatie, voor wat betreft blocking, locking van bepaalde datastructuren.
- Infrastructuurbelasting, zoals geheugengebruik, netwerkbelasting of gebruikte poorten.
- Database acties, worden er insert, update, delete acties uitgevoerd op de dataset.

Projectorganisatie

6. *Cross Functional Team*

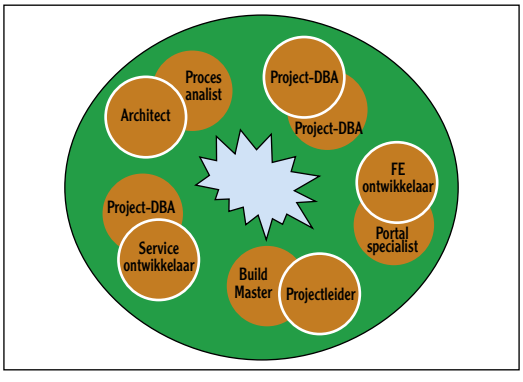
Een SOA project vereist inzet van een reeks verschillende rollen, die tevens variëren per fase van het project. Ter vergelijking: in een traditioneel project worden maar enkele rollen onderscheiden.

In een SOA project geldt ook dat veel rollen slechts kortstondig benodigd zijn. In theorie zet je dan veel mensen in. In de praktijk werkt dat niet: het project wordt een duiventil.

In een cross functional team vervult ieder teamlid *minimaal* 2 rollen. Er ontstaan hierdoor meer 'lijnen' van samenwerking tussen de teamleden onderling. Dit verbetert de samenwerking binnen het projectteam. In een cross functional team leren medewerkers meer van elkaar en worden vraagstukken sneller vanuit alle noodzakelijke invalshoeken belicht.

Door de service benadering neemt het aantal afhankelijkheden van programma units en dus van medewerkers/rollen snel toe. In een cross functional team dragen alle medewerkers mee aan het oplossen van dit planningsprobleem terwijl de kans op bottlenecks afneemt.

Het lukraak combineren van rollen is geen aanbeveling. Hanteer vijf rollen als uitgangspunt en maak combinaties tot het project voldoende is toegerust: Architect, BPEL ontwikkelaar, Front-end ontwikkelaar, Service ontwikkelaar, Projectleider. De combinatie van een cross functional team met Scrum als projectmanagementmethodiek is nog beter. De open communicatie en korte opeenvolgende iteraties (sprints) zorgen ervoor dat beide instrumenten elkaar versterken.



Figuur 3

7. *Processen*

De aandacht voor de analyse van processen kan bijna niet groot genoeg zijn. Het realiseren van de beloften van een SOA implementatie is hier voor een groot deel van afhankelijk. Een gebod om de processen te analyseren is een open deur. Dit gebod geeft wel een reeks tips om de processen juist te analyseren.

- Maak een onderscheid in bedrijfsprocessen en usertask processen.
- Besteed aandacht aan ondersteunende processen. Hoewel deze niet de core business vormen zijn ze wel bepalend voor de flexibiliteit in de SOA implementatie.
- Beschrijf de processen en geef aandacht aan eisen aan het proces en knelpunten die benoemd worden.
- Betrek zowel de opdrachtgever als 'de werkvloer' bij de procesanalyse. De laatste groep voert het werk dagelijks uit en hun werkwijze kan mogelijk verschillen met wat het management denkt.
- Analyseer meer dan alleen de 'sunshine flows' of 'happy flows' (zie ook 10).
- Overzie de resultaten van de procesanalyse en destilleer daaruit generieke processen, dit verhoogt de herbruikbaarheid van services van alle classificaties (zie ook 4).

8. *Integratietests*

Voor traditionele homogene applicaties volstaan unit tests en systeem tests. Deze applicaties worden in isolatie getest op eigenschappen zoals beschikbaarheid, veiligheid en functionele correctheid. Voor een SOA volstaat dit niet: een andere aanpak is noodzakelijk.

Een SOA omgeving is heterogeen en bestaat uit losse applicaties en systemen die door middel van services verbonden zijn om een business proces te ondersteunen. Dat alle losse services in isolatie zijn getest, wil nog niet garanderen dat deze ook correct functioneren in compositie. Onvolkomenheden komen aan het licht ná compositie, doordat een aanroepende service een aanname doet van een service die hieraan niet kan voldoen, en vice versa. Integratietests zijn noodzakelijk om te garanderen dat de services in compositie correct gedrag vertonen.

Neem integratietests op vanaf het begin van het project. Zo wordt voorkomen dat het achterhalen van de oorzaak en het verhelpen van de fout veel

tijd in beslag neemt. Gebruik mock-up services om service interfaces te testen. Integreer tests in een build server zodat de tests dagelijks / continu uitgevoerd worden.

Implementatie

9. *Canonical Model*

Een SOA aanpak is bij uitstek geschikt om functionaliteit over verschillende heterogene systemen heen te realiseren. In de praktijk heb je dan snel met een mismatch van datamodellen of met interpretatieverschillen van data te maken. Ieder systeem of zelfs service heeft namelijk zijn eigen visie op de wereld. Overlappende functionaliteit, bijvoorbeeld gerealiseerd in de vorm van een BPEL proces, moet eenheid creëren in deze heterogeniteit.

Maak voor ieder business proces een canonical model (zie kader) dat alle gebruikte entiteiten beschrijft. Het is niet de bedoeling dat dit een soort Enterprise Data Model wordt, waarin alle mogelijke attributen gespecificeerd zijn. Het voldoet als bekend is om welk type en id het gaat.

Aan de hand van deze gegevens kan dan voor het aanroepen van een service de benodigde data opgehaald worden. Ook kan de data indien nodig getransformeerd worden naar de door de service benodigde vorm. De belangrijkste winst van de toepassing van canonical models is een betere scheiding tussen proces en service. Het onderscheiden van specifieke doelstellingen zoals proces en service en deze separaat implementeren is een mindset die hoort bij SOA.

10. *Exception Handling*

Vaak wordt bij procesmodellering uitgegaan van de zogenaamde 'sunshine flow': het proces wordt beschreven alsof het in de ideale wereld wordt uitgevoerd. Helaas is de praktijk anders en is er behoefte aan patronen voor foutafhandeling waarmee robuuste processen kunnen worden gemodelleerd die bestand zijn tegen de werkelijkheid. In traditionele software ontwikkeling is foutafhandeling gemeengoed; iedere ontwikkelaar heeft kennis over hoe fouten afgehandeld moeten worden. Voor procesmodellering is dat nog niet het geval. Een proces is een orkestratie is van services, waarbij het niet altijd inzichtelijk is hoe een service zich gedraagt binnen een transactie op procesniveau. Indien een service een onverwacht antwoord teruggeeft aan het proces, moet het proces in staat zijn het onverwachte antwoord te kunnen verwerken. Er is nieuwe aandacht vereist voor exception handling.

SOA exceptions zijn te categoriseren in een vijftal verschillende types[1]: (1) falen eenheid van werk, (2) verstrijken van deadline, (3) niet beschikbaar zijn van middelen, (4) externe gebeurtenissen, (5) overtreding criteria.

Canonical Model

Een canonical model is een onafhankelijk en neutraal 'data'model dat onafhankelijk is van elke applicatie of toepassing. Een canonical model is samengesteld uit entiteiten die een rol spelen in specifieke bedrijfsprocessen. De inzet is dat het zorgdraagt voor de translaties van data. Door de data van elke in het bedrijfsproces geraakte toepassing te vertalen van en naar het canonical model fungeert het als integratielaag voor data. Een canonical model hoeft geen persistentie in een database te hebben, het gaat om de translaties. De toepassing van een canonical model leidt dus tot een reeks translaties van data. Deze kunnen uitstekend worden gerealiseerd met een Enterprise Service Bus (ESB), een onderdeel dat je ook om andere redenen graag inzet in een SOA omgeving.

Voor elk type exception kunnen patronen worden opgesteld binnen de uitvoering van het proces, dan wel gerefereerd worden in de uitvoering van het proces. Het is een kwestie van de juiste patronen definiëren, specifiek voor de business waarin de processen uitgevoerd worden. Ieder patroon is een proces op zichzelf, en daarmee wordt exception handling een bedrijfsproces.

Hierdoor wordt het ook een modelleringvraagstuk, waarbij business kennis vereist is. Desondanks zijn er een aantal generieke patronen voor ieder type exception op te stellen.

Eén belangrijk patroon is het *compensatie proces*. Hierin wordt het proces teruggedraaid om de integriteit en operationele consistentie te kunnen waarborgen. In een compensatie proces worden services aangesproken die compensatie handelingen uitvoeren in de onderliggende systemen. Het is daarbij van belang dat iedere service die aangesproken kan worden in een proces self-contained is, en daarmee in staat is zijn eigen compensatie te implementeren en deze als operatie beschikbaar te stellen. Dit stelt specifieke eisen aan het serviceontwerp.

Daarmee is dit tiende gebod eigenlijk een set van twee: exception handling op het niveau van processen en eisen aan services die functionaliteit in zich dragen om hun transactie terug te draaien.

We realiseren ons dat de beschrijving van de tien geboden te kort is om de volledige betekenis en impact weer te geven. Op de weblog www.it-eye.nl/weblog/category/it-eye-focus/soa-patterns worden de komende weken alle geboden uitgebreider beschreven, toegelicht en bediscussieerd.

Samengesteld door een aantal SOA specialisten van IT-eye: Andrej, Bert, Ingmar, Rene, Ronald, Theo en Tom.

Referenties

1. N. Russell, W.M.P. van der Aalst, A.H.M. ter Hofstede. Workflow Exception Patterns.