COLUMN CHRIS VERHOEF

Anticiperen op onvolkomenheden

Wat me telkenmale opvalt, is dat IT-intensieve systemen vaak alleen de happy flow implementeren; elke afwijking daarvan leidt direct tot fouten en falen van het systeemcomplex, meestal meteen na indienststelling van het systeem. De happy flow is de gang door een (bedrijfs)proces als alles volgens het boekje loopt. Een simpel voorbeeld uit de securityhoek is SQL injection. De happy flow is dat een gebruiker inlogt met naam en wachtwoord. Een alternatief is dat iemand SQL-code injecteert om zonder naam of wachtwoord binnen te komen. Als nooit is nagedacht over die mogelijkheid, is de kans groot dat er geen preventieve maatregelen genomen zijn om ongenode gasten te weren.
Het anticiperen op onvolkomenheden in systemen is al meer dan een halve eeuw onderdeel van een gedegen ontwerpproces. Althans, dat behoort het te zijn. Meer dan zeventig jaar geleden bedacht de Amerikaanse Defensie dit al: de FMECA. Dit staat voor Failure Mode, Effects and Criticality Analysis. In de militaire standaard MIL-STD-1629A worden de procedures beschreven voor het uitvoeren van een FMECA. Deze methode heeft het tevens geschopt tot internationale standaard: de IEC 60812 (Analysis techniques for system reliability – Procedure for failure mode and effects analysis).

Voorkom dat fouten je overkomen zonder dat je daarop geanticipeerd hebt

MECA is een methode om uit te vissen hoe een systeem allemaal zou kunnen falen, hoe erg dat is, en wat je kunt doen om falen te voorkomen, en welke maatregelen je allemaal moet nemen mocht het toch tot falen komen. De militaire standaard verwoordt het aldus: "This standard establishes requirements and procedures for performing a failure mode, effects, and criticality analysis (FMECA) to systematically evaluate and document, by item failure mode analysis, the potential impact of each functional or hardware failure on mission success, personnel and system safety, system performance, maintainability, and maintenance requirements. Each potential failure is ranked by the severity of its effect in order that appropriate corrective actions may be taken to eliminate or control the high risk items."
Dat is een mondvol maar er zit geen woord Spaans bij. Ter illustratie aangaande het SQL-injectionvoorbeeld: doe aan inputvalidatie waardoor je SQL-commando's afvangt. Werk met stored-procedures zodat willekeurige SQL-commando's überhaupt niet uitgevoerd kunnen worden, indien er toch ergens een gaatje blijkt te zitten. Vang de foutmeldingen van de database af zodat diagnostische informatie van de database-engine ongenode gasten niet in de kaart speelt. Kijk, daar heb je wat aan.
Een FMECA is geen statisch verhaal dat je een keertje kunt doen en dat is het dan. De FMECA voer je uit in verschillende stadia van de levenscyclus van een systeem. Bij de beginstadia, tot aan de exploitatiefase. Het is niet beperkt tot het systeem maar kan ook dienen om het proces de maat te nemen dat het systeem ondersteunt. En omdat tijdens het ontwerp zaken nog weleens veranderen, is aanpassing of herhaling van de FMECA dan nodig.

Het aardige van dit soort methodes en technieken is dat er reeds lang en breed over is nagedacht, dat het in standaarden is neergelegd, en dat dit precies is wat je nodig hebt als je IT-intensieve systemen wilt (laten) ontwerpen, bouwen en/of exploiteren. Er is maar een ding dat je niet moet nalaten: eisen dat het ontwerpproces geïntegreerd de FMECA-methodologie meeneemt. Daarna moet je uiteraard daadwerkelijk regie voeren en borgen dat de FMECA's worden uitgevoerd, en bij veranderende eisen of ontwerpen worden aangepast.
Het leven wordt niet gemakkelijk door FMECA's, omdat je daarmee het ontwerpproces uitbreidt. Omgekeerd: het wordt moeilijker, en uitvoerders gaan klagen en draaien om eronderuit te komen. Daarom is stringente regievoering noodzakelijk.
Uitbreiding met FMECA's zekert wel dat niet alleen de happy flow wordt uitgemodelleerd, en je systeem gewoon niet fit-for-purpose blijkt na oplevering. Daarnaast eis je dit ook voor onderhoud en beheer zodat je in latere fasen van de levenscyclus ook de fout- en faalmodi in kaart hebt en daar waar mogelijk (preventieve) maatregelen neemt. En dit voordat de fouten je overkomen zonder dat je erop had geanticipeerd. Het alternatief van laat-maar-waaien is bij vele systemen blijkbaar geen optie: veel te veel systemen blijken het meteen niet goed te doen bij oplevering. De kosten daarvan zijn vak inhibitief hoog ten opzichte van ontwerpkosten waar in de beginstadia meer dan het meest simplistische wordt ingecalculeerd.

Prof. dr. Chris Verhoef is hoogleraar Informatica aan de Vrije Universiteit Amsterdam

Een servicegeoriënteerde architectuur (SOA) moet een organisatie flexibeler maken. Maar om dat te realiseren, is een besturingsproces nodig dat meegroeit met de architectuur. Het is lastig het juiste governancemodel te kiezen. **Hidde Andriessen cum suis** gaan in op de IT-doelstellingen van een SOA, op de risico's en de te nemen maatregelen.

Serviceoriëntatie lastig te managen

Groot aantal governancemodellen zijn drempel voor organisaties

Serviceoriëntatie is een manier om IT beter aan te laten sluiten op (veranderende) bedrijfsdoelstellingen. De organisatie van vandaag moet zich steeds sneller aanpassen aan een veranderende markt, steeds meer transparantie bieden aan haar klanten en steeds meer integreren met ketenpartners. Serviceoriëntatie wordt ingezet als middel om te zorgen dat op de juiste punten flexibiliteit bestaat. Hierdoor zijn veranderingen in de producten- en/of dienstenportfolio, de bedrijfsprocessen of de organisatiestructuur eenvoudiger door te voeren.
Veel organisaties vergeten dat voor een succesvolle implementatie van een servicegeoriënteerde architectuur (SOA) wel een besturingsproces nodig is. De beoogde doelstellingen worden namelijk niet vanzelf bereikt (zie kaders).
Tegelijkertijd is het een uitdaging om de invoering van een SOA pragmatisch te houden, zodat ook op kortere termijn al concrete winstpunten behaald worden. Een groot voordeel is dat SOA gefaseerd ingevoerd kan worden. Hiervoor is een besturingsproces nodig dat mee kan groeien met de SOA, oftewel evolutionaire SOA Governance.

Er bestaan veel governancemodellen op verschillende organisatieniveaus waar een SOA-governanceproces van afgeleid zou kunnen worden, zoals COSO (Corporate Governance) en CobiT (IT Governance). Tevens hebben onder andere IBM en Oracle modellen ontwikkeld voor SOA Governance. Het nadeel van al deze modellen is echter dat ze nogal overweldigend kunnen overkomen.
Een governancemodel moet aansluiten bij de mate van complexiteit van datgene wat bestuurd wordt. Het heeft weinig zin om een corporate-governancemodel in te richten voor een eenmanszaak. Wanneer te snel gestart wordt met grote modellen, bestaat het risico dat er meer tijd in het inrichten van het governanceproces wordt gestoken dan nodig is, waardoor

Een corporate-governancemodel voor eenmanszaak heeft geen zin

het proces te zwaar is en losstaat van de dagelijkse praktijk. Het feit dat veel modellen volwassenheidsniveaus onderscheiden, neemt niet weg dat toch het hele model doorgrond moet worden. In de praktijk betekent dit dat organisaties deze modellen, hoe essentieel ook, als een drempel ervaren.
Vanaf de start van het opzetten van een SOA is echter wel een minimale besturingsvorm nodig om te voorkomen dat het initiatief uit de hand loopt. Als te laat gestart wordt met SOA Governance is de urgentie wel duidelijk, maar is het de vraag of zaken nog onder controle te krijgen zijn.

Een veel gebruikte managementmethode is risicoanalyse. Zij vormt ook de basis van het COSO Enterprise Risk Management (ERM). In een risicoanalyse worden voor een specifiek onderwerp de belangrijkste risico's in kaart gebracht. Daarbij wordt gekeken naar de kans en de impact van de risico's en worden maatregelen gedefinieerd om die risico's te reduceren (zie kader).
Risicoanalyse kan ook gebruikt worden bij het inrichten van een governanceproces. Hierdoor kan bottom-up gekeken worden wat initieel de belangrijkste maatregelen zijn die genomen moeten worden. Door het governanceproces te herhalen, kan men relatief eenvoudig opstarten en sturing uitoefenen op een SOA. Naarmate de SOA groeit, kunnen steeds meer onderdelen uit een standaardmodel worden geïntroduceerd om uiteindelijk volledig aan te sluiten bij een standaardmodel.

Hidde Andriessen (hiddea@infosupport.com), Raimond Brookman (raimondb@infosupport.com), Bas Meerman (basm@infosupport.com) en Alexander Molenkamp (sanderm@infosupport.com) zijn IT-architecten bij Info Support. Dit artikel is een bewerking van de whitepaper 'Evolutionaire SOA Governance'.

Voor reacties en nieuwe bijdragen van deskundigen: Henk Ester (h.ester@sdu.nl, (070) 378 03 97).

ILLUSTRATIE: JOS THOMMASSEN

Maatregelen

Een lichtgewicht SOA-governanceproces voor startende SOA-implementaties bestaat uit zeven maatregelen. Deze maatregelen moeten in de volledige governancecyclus worden opgenomen (bijvoorbeeld aan de hand van de kwaliteitscirkel van Deming: Plan, Do, Check, Act).

1 Bepaal SOA-strategie
Het is noodzakelijk vast te stellen wat de doelstellingen zijn en te beoordelen of SOA inderdaad de beste manier is om deze doelstellingen te realiseren. De doelstellingen dienen 'smart' gemaakt te worden, in lijn met de bedrijfsstrategie en bedrijfsdoelstellingen.

2 Definieer een SOA-referentiearchitectuur
Op basis van de SOA-strategie dient een SOA-referentiearchitectuur opgesteld te worden, waarin principes en richtlijnen voor de inrichting worden bepaald. Deze principes en richtlijnen moeten aansluiten bij de bedrijfsinrichting en -doelstellingen. Op hoofdlijnen moeten hiervoor de bedrijfsfuncties, bedrijfsprocessen en informatiedomeinen bekend zijn, idealiter vastgelegd in een enterprisearchitectuur.

3 Bestuur het ontwikkelproces
Maak expliciet welk proces er gevolgd moet worden voor het modelleren en ontwikkelen van services. Minstens zo belangrijk is ervoor te zorgen dat het proces ook daadwerkelijk wordt gevolgd. Bedenk prikkels om het volgen van het proces te bevorderen. Gebruik hiervoor een mix van controlemechanismen en beloningen.

4 Bepaal metrieken
Het is essentieel om metrieken te verzamelen om de toegevoegde waarde van de SOA te bewijzen. Gebruik hiervoor metrieken die al aanwezig zijn, zoals het aantal incidenten, de verwerkingstijd per change et cetera. Bepaal welke verbetering er wordt verwacht op basis van de SOA-doelstellingen. Voeg metrieken toe waar nodig, maar meet niet te veel. Dat is onnodig, duur en kan zelfs averechts werken.

5 Stel een servicecatalogus samen
Besturing van een SOA kan niet zonder een overzicht van alle services. Services hebben verschillende levensfasen (definitie, ontwerp, bouw, test, productie tot en met end-of-life). Er dient in minimale vorm een servicecatalogus aanwezig te zijn. Begin niet direct met het inzetten van uitgebreide tools; aanvankelijk kan bijvoorbeeld een spreadsheet volstaan.

6 Vergeet de marketing van de SOA niet
Stel een SOA-marketingplan op waarin wordt vastgelegd hoe de beoogde en behaalde doelstellingen worden besproken met de rest van de organisatie. Het is belangrijk om voldoende draagvlak te creëren en te behouden door onder andere behaalde successen te vieren.

7 Richt een SOA-competence-center op
Begin geleidelijk met het borgen van opgebouwde kennis in een SOA-competence-center. Deze kan dan dienen als vraagbaak voor projecten en kan ingezet worden voor de handhaving van de governanceregels.

IT-doelstellingen SOA

1 Verhogen flexibiliteit (veranderbaarheid) van IT
Een goed ontworpen SOA maakt het mogelijk om functionaliteit toe te voegen of te wijzigingen zonder dat hiervoor het hele systeemlandschap aangepast hoeft te worden.

2 Verhogen herbruikbaarheid van IT
Een goed ontworpen SOA bestaat uit een aantal gemeenschappelijke services die door verschillende bedrijfsfuncties of bedrijfsprocessen gedeeld kunnen worden.

3 Verbeteren business/IT-alignment
Een goed ontworpen SOA kent een opdeling in services die herkenbaar is voor de business. Deze opdeling is idealiter direct te relateren aan de services die op bedrijfsniveau zijn gedefinieerd. Hierdoor ontstaat een gemeenschappelijk vocabulaire waardoor de afstemming tussen business en IT verbetert.

4 Ontvlechting van ontstane spaghetti van koppelingen tussen systemen
Door het uiteenrafelen van 'een-op-een'-systeemkoppelingen naar services met duidelijke verantwoordelijkheden wordt het systeemlandschap inzichtelijker en beter beheersbaar.

5 Integratie van systemen
Via services kan de informatie uit diverse systemen op een uniforme manier ontsloten worden. Dat maakt het integreren van deze informatie eenvoudiger en maakt integratie van systemen mogelijk.

6 Verbeteren van gegevenskwaliteit
SOA helpt hierbij doordat binnen een SOA duidelijke afspraken gemaakt worden over het eigenaarschap van diensten en bijbehorende informatie. Services, verantwoordelijk voor bepaalde gegevens, zorgen voor eenmalige vastlegging en meervoudig gebruik.

Risico's

■ **Onvoldoende businessbetrokkenheid**
Veel SOA-implementaties worden geinitieerd vanuit de IT-afdeling. Echter, om de doelstellingen omtrent flexibiliteit en in mindere mate de vierde, vijfde en zesde doelstelling (zie kader) te behalen, moet samen met de business bepaald worden wat de meest effectieve services zijn voor de organisatie. Zonder voldoende businessbetrokkenheid is dat onmogelijk en worden de verkeerde keuzes gemaakt.

■ **Inrichting sluit niet aan bij de bedrijfsdoelstellingen (SOA is niet effectief)**
Een SOA die zich richt op IT-doelstellingen die niet bijdragen aan bedrijfsdoelstellingen is niet effectief. Een veel voorkomend risico is dat gestreefd wordt naar meer flexibiliteit, maar dat niet concreet wordt gemaakt waar die flexibiliteit nodig is. En dat terwijl SOA op veel verschillende manieren invulling kan geven aan flexibiliteit.

■ **Bestaand systeemlandschap past niet bij een SOA-inrichting**
Wanneer SOA wordt toegepast op een bestaand systeemlandschap is het noodzakelijk dat de systemen ontsloten kunnen worden zonder dat dit invloed heeft op het servicecontract. Tevens moeten de systeemgrenzen zich in voldoende mate kunnen conformeren aan de bedrijfsfuncties en informatiedomeinen die gebruikt worden om de services van af te leiden.

■ **Services worden gerealiseerd op basis van verkeerde bedrijfsfuncties/informatiedomeinen/bedrijfsprocessen**
Randvoorwaarde voor een succesvolle SOA is dat de bedrijfsinrichting voldoende bekend is. Als bijvoorbeeld de bedrijfsfuncties niet bekend zijn of niet goed ontworpen zijn, heeft dit direct invloed op de mate waarin een service zal bijdragen aan flexibiliteit en herbruikbaarheid. De SOA-inrichting zal dan niet effectief zijn.

■ **Cultuur past niet bij SOA-werkwijze**
Een succesvolle SOA vereist een hoge mate van samenwerking tussen de betrokken afdelingen om voor de organisatie als geheel de meest optimale services te bepalen. In een cultuur waarin afdelingen te veel voor eigen belang strijden is dit niet haalbaar.

■ **SOA is van slechte kwaliteit (SOA is niet efficiënt)**
Een standaardrisico voor elk softwareontwikkelingstraject is dat de ontwikkelde services van slechte kwaliteit zijn. Het gevolg is dat de SOA niet efficiënt is en daardoor niet of slechts minimaal zijn doelen zal bereiken. Hier kunnen verschillende oorzaken aan ten grondslag liggen, zoals het niet volgen van principes en bestpractices en het gebrek aan kennis, vaardigheden en/of draagvlak.

■ **Technologieplatform draagt niet bij aan de SOA**
Er zijn veel technologieplatformen (BPM, WFM, DMS, portals et cetera) die standaardfunctionaliteit beschikbaar stellen om services mee te realiseren. Deze platformen moeten dan wel ondersteuning bieden voor serviceoriëntatie. Is dit niet het geval, dan zal veel extra maatwerk nodig zijn en/of zal het resultaat zeer beheerintensief zijn.