Gijs zijn tijdloze IT wijsheden

voor studenten, stagiairs, afstudeerders en young professionals in de IT

"Is there anyone so wise as to learn by the experience of others?" - Voltaire

Inhoudsopgave

Inleiding 4
Past het bedrijfsproces wel bij het klantproces? 5
Outsource de business in plaats van de IT 7
Vertel de klant wat hij wil 9
Bring Your Own Business Process 11
Hoe borg je nu eigenlijk architectuur? 13
Verwijderingsbijdrage kan ook voor software 15
Meer “domme” klanten vinden of blijven innoveren? 17
You can have any color as long as it is black 19
Oplossing op zoek naar een probleem? 20
Rondrennende mannetjes en sla 22
SaaS en Scrum, vrienden voor het leven 23
Ook het bouwen van een luchtkasteel vergt een goede architect 25
Achteruitautomatiseren 27
Beschikbaarheid en schijnveiligheid 29
Een oplossing zonder beveiliging is geen oplossing 31
Usability – keep IT simple? 33
Is de cloud nog wel te bevatten? 35
Supportability first! 37
Durven Nederlanders nog wel te innoveren? 39
Staat de software ontwikkelaar op uitsterven? 41
Het einde van de silo applicatie en de roadmap 43
Pair architecting 45
Business Architect is ook een vak 47
De Apps tegen de Applicaties 49
API’s kijken 50
Anarchitectuur 51
Het einde van het document als bron 52
Probleren 53
Zelfrijdende auto als Paard van Troje 54

Disruptie door de gevestigde orde 55
Two and a half speed IT 57
Terug in je hokje? 59
Europa als digitale kolonie van Silicon Valley 61
Het draait allemaal om de business app 62
Digitaal bladgoud 64
Een IT'er is ook maar gewoon een uniek mens 66
Vertrouwen oogsten na agile zaaien 68
Over kannibalen en blockchains 70
Help, ik ben gedigitaliseerd! Wat nu? 72
Serverless maar niet mindless 74
Zelfsturende teams die wèl werken 76
Beleid automatiseren 78
Containers zijn gewoon IaaS lieve mensen 79
Digitale self-service als productiviteitskiller 81
De IT moet niet te goed worden 83
Voorbij de relationele database 84
Applicatieverlatingsangst 86
Datagedrevenheid 88
Het einde van monolitische platforms 90
Slimme data op een presenteerblaadje 92
Falen met software is een dure hobby 94
Teveel ballen in de lucht houden 96
Kantelen voor procesoptimalisatie 98
Frankenstein IT-oplossingen 100
Over Gijs 102

Inleiding

Voor je ligt een verzameling in boekvorm van mijn meest gewaardeerde columns van de afgelopen 7 jaar, die onder andere in Computable zijn gepubliceerd. Ze zijn in het boekje in chronologische volgorde opgenomen. Sommige hoofdstukken gaan misschien over technologie die al enigszins achterhaald is, maar de essentie van de verhalen klopt nog steeds.

Wat goed dat je dit leest! Dat betekent dat je als student, stagiair, afstudeerder of young professional in de IT leergierig bent en open staat voor de ervaringen van mensen met al wat IT-kilometers achter de rug zoals ik. Natuurlijk leer je het meeste van je eigen fouten, maar het zou best fijn zijn als we nóg sneller kunnen evolueren als IT mensheid door niet telkens weer dezelfde fouten te maken bij het ontstaan van elk nieuw IT paradigma. Want dat gebeurt echt nog steeds te vaak heb ik ervaren.

Ik hoop dat je wat aan mijn ervaringen hebt!

Feedback is altijd welkom. Stuur maar iets aan @gintveld op Twitter of gijs@indafield.com. Ben benieuwd!

Veel leesplezier!

Past het bedrijfsproces wel bij het klantproces?

Iedereen weet hoe het is om als burger te maken te hebben met bedrijven en instellingen waarbij het geautomatiseerde systeem niet echt aansluit op hetgeen je als eindgebruiker verwacht. Little Britain's 'computer says no' is erg grappig, maar helaas ook te vaak realiteit.

De meeste geautomatiseerde systemen van bedrijven zijn ontstaan met het primaire bedrijfsproces als uitgangspunt, netjes verdeeld over afdelingsspecifieke doeleinden. Deze systemen zijn na uitgebreide selectieprocedures en maatwerkprojecten in productie genomen. Te vaak zie je nog dat het primaire bedrijfsproces van een willekeurige instelling is ingebed in een monolitische applicatie, zelfs bij 'moderne' implementaties. Terwijl juist het monolitische (silo) karakter van deze applicaties vaak de aanstichter is van het leed dat is beschreven in de eerste paragraaf. Ook is het vaak zo dat er een wildgroei aan applicaties is ontstaan, mede veroorzaakt door overnames en fusies of omdat er voor een (tijdelijke) tactische oplossing is gekozen om een korte termijn uitdaging aan te kunnen.

Zaakgericht werken

Zaakgericht werken is hier een mooie oplossing voor, echter spelen er buiten de gestructureerde processen rondom de zaak zich ook veel ongestructureerde processen af. Ook is het vaak zo dat applicaties of platformen die zaakgericht werken mogelijk maken niet goed aan te sluiten zijn op andere, ook relevante backoffice-applicaties, omdat deze niet goed kunnen fungeren binnen zo'n flexibele architectuur. Zaakgericht werken is ook per definitie een fenomeen dat afdelingsoverstijgend is, en dat past toch vaak niet in bestaande applicatie-architecturen.

De best mogelijke oplossing voor dit alles is het inrichten van een meerlagen BPM (business process management) model in een SOA (service oriented architecture), waarbij de afzonderlijke (black-box) functies en diensten van de applicaties in het applicatielandschap beschikbaar worden gesteld als services die voldoen aan de service design principes opgesteld door Thomas Erl. Door de informatie- en procesarchitecten de gereedschappen te geven om processen flexibel in te richten, gebruikmakend van de services die ontsloten zijn binnen en eventueel buiten de eigen firewall zal een situatie ontstaan die het mogelijk maakt om snel op veranderende marktomstandigheden in te springen zonder dat daarbij monolitische applicatielogica behoeft te worden aangepast. Door ook nog gebruik te maken van de business intelligence (BI)-faciliteiten van het platform, kan intelligentie worden verzameld over de processen die vervolgens gebruikt kan worden om de processen weer te verbeteren.

Klant is koning

Het sluitstuk, of beter beginpunt (!) van dit alles is een universele presentatielaag die benaderbaar is vanaf elk willekeurig apparaat, elk willekeurig tijdstip en elke willekeurige plaats. Zo'n portaal of set van apps is bij voorkeur een platformomgeving waarin door de functionele beheerders (of door de power-users zelf) in samenspel met de informatie- en procesarchitecten op een flexibele manier gebruikersinterfaces samen te stellen zijn die perfect aansluiten bij de doelgroep en de betreffende processen. Door het zaakgericht werken in deze omgeving mogelijk te maken, ondersteund door de document management-faciliteiten van het platform, is een omgeving te implementeren die snel en wendbaar is en waarbij de gebruiker ook buiten de 'vaste' processen om, op een gemakkelijke manier kan samenwerken met collega's om op die

manier sneller en beter antwoorden uit de onderliggende logica en systemen te krijgen en dus efficiënter te kunnen communiceren met de uiteindelijke klant.

Het ultieme doel is dat een zichzelf continue verbeterend proces kan worden geïmplementeerd wat mogelijk maakt dat in plaats van dat de klant zich moet aanpassen aan het bedrijfsproces, het bedrijfsproces zich kan aanpassen aan het klantproces. En zo wordt de klant weer koning.

Outsource de business in plaats van de IT

Tijdens een zakelijke lunch na een meeting bij een financiële instelling was er een levendige discussie ontstaan tussen de mensen van IT-operations en R&D over de wrijving tussen 'de Business' en 'de IT'. Aan het einde van de lunch opperde iemand van R&D gekscherend 'Eigenlijk zouden we gewoon de business moeten outsourcen in plaats van de IT'.

Bij veel organisaties is een gezonde spanning waar te nemen tussen de business en de IT. De business wil graag snel op veranderende marktomstandigheden inspringen en de IT-afdeling is te star en traag. En andersom: de IT-afdeling wil graag zaken onder architectuur oplossen zodat de omgeving beheersbaar blijft en toekomstige ontwikkelingen ook goed en toch snel kunnen landen, terwijl de business vaak in silo -oplossingen denkt die alleen de korte termijn dienen.

Service oriëntatie

Toch is vaak de core competency van bedrijven die niet in de categorie brick-and-mortar vallen de unieke verzameling van IT-oplossingen en -diensten die ontwikkeld zijn. De kern van de geautomatiseerde bedrijfsprocessen van menig organisatie bevat het onderscheidende vermogen ten opzichte van de concurrentie. Investeren in een horizontaal, service georiënteerd platform waar slim opgedeelde services met daarin het intellectuele eigendom zijn ondergebracht, waarop flexibel processen kunnen worden gemodelleerd en actief gemeten en bijgestuurd, is het hoogste goed.

Voor bijvoorbeeld banken en verzekeraars, maar ook logistieke dienstverleners blijkt dat het grootste gedeelte van de intellectuele eigendom niet in de uitvoerende business zit, maar juist in de kennis geborgd in de geautomatiseerde systemen. En dan hebben we het niet over de standaard back-office systemen voor financiële administratie of HR, maar over de kern bedrijfssystemen zoals maatwerk ERP-systemen, schadeclaimsystemen, logistieke planningsystemen of handelssystemen. Het gaat hier meestal over bedrijven die als kerncompetentie het virtueel verplaatsen van data en informatie hebben die hiervoor veel geld hebben geïnvesteerd in maatwerkoplossingen waarin de kennis van het bedrijf is ondergebracht in de vorm van vele door een aantal procesarchitecten en bedrijfsanalisten bedachte business rules en complexe algoritmes.

Bij implementatie van BPM en SOA-architecturen, waarbij door middel van service oriëntatie en middleware voor integratie, procesorkestratie, presentatie, document- en zaakmanagement een horizontaal platform wordt geboden waarop 'klantintieme', applicatie overstijgende oplossingen kunnen landen is vaak het grootste probleem: welke business unit of afdeling gaat voor deze horizontale services betalen? Hoe verdelen we de kosten van de ontwikkeling van horizontale diensten over de verschillende business owners? Wie betaalt voor welke business requirement? Aangezien de business alleen geïnteresseerd is in de snel te realiseren silo oplossingen en vaak geen budget heeft voor de langere termijn, zijn strategische investeringen in het platform en het financieren van zo'n service georiënteerde omgeving al snel een bijna onneembare hobbel.

IT als core business?

Toch is het neerzetten van zo'n platform een vereiste om snel op veranderende marktomstandigheden in te springen. Het lijkt er dan meer en meer op dat de IT-afdeling eigenlijk de core business is van het bedrijf

waar alle strategische ontwikkeling plaatsvindt, en de uitvoerende business eigenlijk de 'commodity' is die geïmplementeerd is op de slimme, flexibele op BPM en SOA gebaseerde applicaties.

We praten dan niet meer over het outsourcen van de ondersteunende, secundaire bedrijfsprocessen zoals HR of financiële administratie onder de noemer business process outsourcing (BPO), maar echt over een verschuiving van de kerncompetentie van dit soort bedrijven naar IT. Het ontsluiten van deze slimme IT-omgevingen richting eindgebruikers als klanten is dan iets wat door 'iedereen' kan worden gedaan, zelfs in de vorm van Business Process as a Service (BPaaS) in de cloud of in de vorm van franchise constructies. Het betreft hier dan de strategische kernprocessen van een bedrijf wat zich bezig houdt met virtuele diensten, zoals bijvoorbeeld verkoop en afhandeling van hypotheken. Het verdienmodel voor deze BPaaS oplossingen zit hem dan in het pay-as-you-go model gebaseerd op zorgvuldig uitgekiende business modelllen.

In plaats van het outsourcen van de IT zullen we de business gaan outsourcen en zullen meer en ook kleinere organisaties in staat zijn om professionele, goed doordachte en bewezen diensten te kunnen gaan leveren aan hun klanten. De BPaaS leveranciers zullen hier natuurlijk wel bij varen, maar het algemene service niveau van de bedrijven die gebruik maken van deze oplossingen om diensten te kunnen leveren zal sterk stijgen met als resultaat een hogere klanttevredenheid en grotere totale welvaart. Maar allereerst zal er toch geïnvesteerd moeten worden in de platformoplossingen, zonder grote budgetten vrijgemaakt door de business. Wie durft?

Vertel de klant wat hij wil

Tezamen met 'slecht projectmanagement' vormt 'slechte requirements' de top 2 van redenen waarom softwareprojecten vaak uitlopen of helemaal mislukken. Bij het van de grond af ontwikkelen van nieuwe software oplossingen voor organisaties is het altijd een uitdaging om eisen en wensen op zo'n manier in kaart te brengen dat het opgeleverde systeem op basis daarvan geaccepteerd of afgewezen kan worden en het resultaat goed bruikbaar is.

Het verzamelen van de business-, user-, functional- en non-functional (quality of service) requirements is een karwei wat altijd onderschat wordt op de volgende aspecten:

• Weet de klant wel wat hij eigenlijk wil?

• Bepalen de juiste mensen de requirements?

• Is de lijst van requirements volledig?

• Kunnen requirements ondubbelzinnig worden gespecificeerd?

• Hoe maken we requirements testbaar?

De laatste twee aspecten zijn relatief eenvoudig te adresseren. Door gebruik te maken van een deugdelijk requirements gathering tool (veelal Excel op basis van een goede template) en in vervolgdocumenten zoals functioneel ontwerp, technisch ontwerp en testplan een requirements index op te nemen kan veel geautomatiseerd worden. Requirements moeten zo veel mogelijk SMART gedocumenteerd worden. Het blijft natuurlijk wel monnikenwerk, maar het is in het algemeen goed te doen. Uiteindelijk kunnen dus bij het uitvoeren van het testplan tijdens de acceptatietests de requirements afgetikt gaan worden en bepaald worden of het opgeleverde aan de eisen voldoet.

Tegenstrijdige requirements

Op de vraag 'Is de lijst van requirements volledig' is vaak moeilijker antwoord te geven. Wel helpen de juiste tools ook hier bij. Vooral voor de lijst van non-functional requirements (zoals performance en beschikbaarheidseisen) is snel te bepalen of het een volledig overzicht is omdat het vaak een standaard lijst is. De moeilijkheid zit hem hier vaak in tegenstrijdige requirements zoals performance versus security en flexibiliteit versus beheersbaarheid. Gebruik requirements versioning om ook veranderende requirements tijdens de uitvoer van het project goed te kunnen managen.

De twee moeilijkste aspecten staan bovenaan in de lijst. We kunnen heel ondubbelzinnig user requirements specificeren naar aanleiding van workshops met een aantal sleutelgebruikers, maar als deze sleutelgebruikers verkeerd zijn geselecteerd kan het resultaat verre van optimaal zijn. In dit geval zal vaak nog steeds wel redelijk met het systeem gewerkt kunnen worden en levert het zijn voordelen toch nog op. Het is vaak de deskundigheid en ervaring van de interviewer cq. functioneel consultant die bepaalt hoe goed en consequent de uiteindelijke lijst van user requirements is, ongeacht eventuele slechte input.

Meetbaar maken

Het allermoeilijkste aspect is natuurlijk de eerste: 'Weet de klant wel wat hij eigenlijk wil?'. De business requirements bepalen hoe het systeem de bedrijfsdoelstellingen mede helpt te gaan behalen maar zullen ook gebruikt worden om de business case voor het nieuwe systeem meetbaar te maken. Maar wie bepaalt die business requirements eigenlijk? Degene die het systeem gaat gebruiken? Degene die de besluiten neemt? Of degene die ervoor betaalt?

In de vele requirements sessies die ik in mijn carrière heb mogen leiden of meemaken was het de uitdaging om vanuit je eigen kunde en ervaring de mensen die verantwoordelijk zijn voor het aanleveren van de business requirements op één lijn te krijgen en binnen de kaders te houden. Er zijn al genoeg systemen die ik altijd maar 'kinderen met een waterhoofd' noem (bij voorbaat sorry als ik iemand hiermee beledig). Ik bedoel hiermee systemen die té veel kunnen, die functionaliteiten bieden die eigenlijk niet tot de kern of zelfde categorie behoren.

80/20

Dé grootste uitdaging is om de klant tegen zichzelf te beschermen en om een goed samenhangend systeem op te leveren. Jarenlange product management ervaring is hier cruciaal. De taal van de klant spreken en sturend kunnen communiceren is ook erg belangrijk. Uiteindelijk is het de kunst om de klant het gevoel te geven dat ze zelf de requirements hebben bedacht, terwijl jij natuurlijk al lang wist hoe het systeem er uit moest zien. Dit is makkelijker als je systemen ontwikkelt die gebaseerd zijn op een basis uitgangspunt, zoals maatwerk op een standaard ERP-systeem of een standaard platform voor document management. Maar laten we eens heel eerlijk zijn, weten we bij voorbaat eigenlijk al niet wat de klant grotendeels nodig heeft en hebben we dat al niet heel vaak eerder ontwikkeld? 80% van de requirements kan waarschijnlijk één op één gekopieerd worden vanuit een ander project. Alleen die verrekte laatste 20 procent kost meestal 80 procent van de tijd, en dat is vaak het grootste probleem; hoe krijgen we die 80 procent omlaag en veranderen we voor eens en voor altijd die regel?

Bring Your Own Business Process

De eindgebruiker is koning! Vroeger was het zo dat de IT-afdeling voorschreef hoe er met de IT-systemen om moest worden gegaan. Soms kwam dit voort uit de starheid van systemen, maar meestal lag het aan de starheid van IT'ers; zij hebben graag volledige controle. Onder zware druk van de eindgebruiker is dit nu volledig aan het omslaan.

Met de komst van steeds gebruikersvriendelijker apparaten als smart phones en tablets zijn mensen ook steeds 'verwender' geworden op het gebied van automatisering. Voor elke toepassing is wel een app te verkrijgen en hoe meer apps in de app store hoe beter de leverancier gewaardeerd wordt. Het hele app-model is natuurlijk mogelijk gemaakt door de komst van de cloud en Software-as-a-Service modellen. Facebook, Google en Twitter in de consumentenmarkt, gevolgd door Salesforce en Microsoft in de zakelijke markt, hebben een enorme impact gehad op de manier waarop gebruikers kijken naar en omgaan met IT. Deze SaaS oplossingen maken het zelfs voor de kleinste gebruiker mogelijk om op volwassen manier mee te doen. Apps doen daar nog een schepje bovenop.

Keurslijf

De tijd is voorbij dat je met de corporate laptop op kantoor komt waarna het ding eerst een half uur de verplichte veiligheidsprocedures doorloopt en de vereiste software updates installeert. Met Bring Your Own Device werd het al rap mogelijk om deel te nemen in bedrijfsprocessen vanaf je eigen tablet, zonder dat daarbij de veiligheid in het gedrang kwam. Heel fijn, omdat je dan niet meer te maken hebt met het keurslijf waarin je werkgever of opdrachtgever je wil stoppen en ook voor tijdelijk, ingehuurd personeel veel beter omdat je dan niet voor elke klant een bijbehorende laptop nodig hebt.

Maar wat nu als we nog wat verder willen? Wat als er dusdanig slimme platforms ontstaan in de cloud, waarbij je zelfs je eigen bedrijfsproces kan meenemen? Bring Your Own Business Process? BYOBP?

Altijd al gefascineerd door code generatie en model driven applications denk ik dat het mogelijk is om je eigen bedrijfsproces mee te nemen en te laten landen op het IT-platform van de organisatie waar je op dat moment voor werkt. Droom ik? Mensen zijn niet in een keurslijf te persen, ze willen graag hun eigen vrijheden bij het uitvoeren van hun dagelijkse werkzaamheden. Het gaat om het resultaat en niet om de manier waarop. De manier waarop wordt grotendeels bepaald door de gebruiker en niet door de organisatie waar zij voor werkt.

Sociaal

Vandaag de dag is met een platform als Office365 al behoorlijk veel mogelijk op dit gebied. Vanuit een door de eindgebruiker getekend Visio-diagram op basis van de standaard BPMN-taal kan redelijk eenvoudig een 'applicatie' (workflow) worden gegenereerd die in Office 365 kan draaien en waarmee je snel aan de slag kan. In feite is Office 365 dan geen SaaS-oplossing meer maar een PaaS-oplossing; een platform waarop je maatwerk oplossingen laat landen. De grote uitdaging zit hem in het bieden van flexibele samenwerkingsmogelijkheden en volledige auditability. Door middel van zo veel mogelijk geautomatiseerde, in het platform ingebouwde governance moet auditability (wie heeft wat, wanneer en waarom gedaan?) goed realiseerbaar zijn. En door een dynamisch, sociaal samenwerkingsplatform te bieden waarbij mensen op een ad hoc manier bij elkaar komen waar en wanneer nodig, kunnen processen

prima op elkaar afgestemd worden. Zo gebeurt het toch eigenlijk al lang, bij de koffiecorner of in de gang? Alleen nu dan ondersteund door een platform wat ook de gebruikersacties registreert en coördineert.

Door BYOBP op deze manier te implementeren kunnen flexibiliteit en controle prima samenleven. Zeker in een tijd waarin het niet meer 'done' is om medewerkers te monitoren op aanwezigheid maar op resultaat (het kan niet anders met het nieuwe werken) moeten ook de onderliggende platforms dit gaan faciliteren.

Hoe borg je nu eigenlijk architectuur?

Architectuur borgen kan net zo betrouwbaar zijn als een derdehands gehoord verhaal; iedereen die wel eens 'de Lama's' heeft gezien (soms is de afstandsbediening niet van mij) weet hoe betrouwbaar dat is!

Op software gebaseerde oplossingen worden bij voorkeur ontwikkeld onder architectuur. Architectuur is belangrijk, voornamelijk om aan non-functional requirements te kunnen voldoen. Om vanuit een enterprise architectuur (EA) een project onder architectuur te laten uitvoeren, is een PSA (project start architectuur) nodig. Een PSA wordt geschreven door een projectarchitect. Tot zover gaat alles op rolletjes.

Maar, hoe borg je de bedachte architectuur nu eigenlijk? Een projectarchitect is prima in staat om bijvoorbeeld een functioneel ontwerp (FO) te beoordelen op architectuuraspecten. Zo'n FO wordt geschreven door een functioneel consultant die begrijpt hoe de business requirements en use cases omgezet kunnen worden in functionele eisen aan een systeem. Deze consultant moet dus in ieder geval de PSA gelezen hebben om zich te houden aan de architectuurrichtlijnen. Om de architectuur te borgen in het FO, moet dit FO dus ook altijd minimaal door een projectarchitect (formeel!) goedgekeurd worden, alvorens het vervolgtraject ingezet wordt. Dit zou dus ook allemaal prima moeten werken in de praktijk.

Creativiteit van de developer

Het wordt moeilijker naarmate we verder in het project terechtkomen. Na het FO volgt een technisch ontwerp (TO). Dit TO wordt veelal door een lead developer geschreven. De lead developer leest het FO, en hopelijk ook de PSA. Een lead developer is al veel meer een 'techneut' dan de functioneel consultant, en nog veel meer dan de architect. Het kan best zijn dat de PSA door hem zodanig wordt geïnterpreteerd dat er zaken in het TO terecht komen die al niet meer helemaal voldoen aan de principes opgesteld in de PSA. Althans, volgens de interpretatie van de lead developer kan het prima kloppen. Hier begint er al iets van een probleem te ontstaan.

Eigenlijk zou de projectarchitect dus ook het TO volledig moeten kunnen beoordelen. Alleen de architect is hiervoor negen van de tien keer niet 'techneut genoeg'. Hier wordt dus soms het principe van 'we beginnen elk vanaf een andere kant te graven en hopen dat we elkaar in het midden tegenkomen' toegepast. Verre van ideaal.

De volgende fase in het project maakt het nog lastiger. De code wordt door een developer geschreven op basis van het TO. De code wordt vaak door een heel team geschreven, elk teamlid verantwoordelijk voor bepaalde (deel)functionaliteiten bij voorkeur in de vorm van componenten. De gemiddelde developer is prima in staat om een TO te lezen en te interpreteren, is al wat minder goed in het begrijpen van FO's en vind architectuur maar iets voor mensen met stropdassen in ivoren torens. Vaak bevat een TO ook niet volledig in detail alle informatie die nodig is om de oplossing te creëren. Daar komt het dus aan op de creativiteit van de developer. Dat is een gevaar. Verder ontwikkelt niemand meer oplossingen volledig zelf tot op de laatste regel code, maar wordt meestal gebruik gemaakt van één of meerdere frameworks zoals bijvoorbeeld het .Net Framework. En in een service oriented architecture (SOA) maakt men vaak gebruik van al bestaande services die eventueel zelfs niet eens in het eigen datacenter draaien.

Het is dus zaak dat de lead developer de code die geschreven is door de developers controleert. Altijd! Maar wie garandeert de lead developer dat bepaalde componenten uit een framework of de services geconsumeerd in de cloud wel voldoen aan de architectuurprincipes beschreven in de PSA? Soms is die informatie niet eens publiekelijk beschikbaar. Maar toch is dat wel relevant, want wat als dat éne component nu wel die rechtstreekse verbinding met een database maakt die volgens de architectuur 'verboden' is, of erger nog, niet voldoet aan de veiligheidseisen?

Mensenwerk

Architectuur kan waarschijnlijk redelijk geborgd worden in de meeste projecten, als het hele projectteam zijn best doet! We kunnen in een project zowel vanaf boven (vanuit de architect, functioneel consultant en lead developer) als van onderaf (vanuit de functioneel consultant, lead developer en developer) architectuur borgen. Vanaf boven kunnen controles op het geproduceerde FO, TO en de code worden gedaan. Van onderaf kan er tijdens het schrijven van FO, TO en code uitgegaan worden van de interpretatie op het desbetreffende niveau van de PSA. Maar, totdat architectuur in echte 'rules' kunnen worden geschreven die geautomatiseerd gecontroleerd kunnen worden op alle niveaus blijft het mensenwerk en vertrouwen op een goed samenwerkend team!

Verwijderingsbijdrage kan ook voor software

Het beheer van de lifecycle van sofware is bij elke organisatie een grote uitdaging. Elk bedrijf dat enigszins serieus met software ontwikkeling bezig is probeert ook rekening te houden met de laatste fase in het leven van software: de decommisioning ofwel het uit productie nemen van de software. In de praktijk blijkt 'de dood' van software echter behoorlijk overdreven (vrij naar Paul McCartney); het is niet uit te roeien.

Vaak blijkt dat het leven van software meerdere malen verlengd wordt door het op te lappen en draaiende te houden, soms zelfs aan het infuus of de monitor. De reden hiervoor is dat het goedkoper is om de bestaande software nog wat langer in de lucht te houden, zelfs als daarvoor nog aanpassingen moeten worden gedaan om het bijvoorbeeld mogelijk te maken aan te sluiten op nieuwere technologie zoals een update van het database management of operating system of de manier waarop met andere componenten wordt gecommuniceerd. Veelal is dit ook iets wat er langzaam insluipt en op een gegeven moment kan men bijna niet meer terug; er is al zó veel in geïnvesteerd. Net als bij een auto, wanneer wordt onderhoud te duur en moet hij naar de sloop?

Verlatingsangst

Ondertussen blijkt men zich in een steeds diepere kuil in te graven. De software die eigenlijk uitgefaseerd had moeten worden volgens de oorspronkelijke plannen (en afschrijving) is eigenlijk bijna niet meer te vervangen omdat het zo'n belangrijke (spil)functie vervult in het gehele applicatielandschap en de integratie is erg 'hecht'. Ook is menig beheerder aan de software gehecht. En het heeft ook zo veel geld gekost! Feit is wel dat het stuk verouderde software ondertussen de verdere evolutie van het applicatielandschap en de daarop landende bedrijfsprocessen tegenhoudt. Maar wanneer geef je de betreffende software een spuitje?

Er is in software lifecycle management vaak wel aandacht en budget voor onderhoud van software. Dit budget wordt gebruikt om bugs op te lossen en kleine changes door te voeren. Ook is er budget voor de operatie van software, zoals beheer en support. Bijna nooit is er echte aandacht voor de laatste fase, het uit productie nemen van software. Wel wat betreft procedures, maar niet wat betreft budget. Software kan natuurlijk niet zomaar uit productie worden genomen, er dient vervanging voor geregeld te worden in de meeste gevallen. En er dient een migratie uitgevoerd te worden. Maar waarom moet het project om deze nieuwe, vernieuwende software in te zetten eigenlijk boeten, of misschien zelfs wel tegengehouden worden vanwege de hoge kosten om de oude software uit te faseren? De oude software is afbetaald en afgeschreven en voor de nieuwe software is budget, maar waar is het geld voor migratie en 'uitzetten' en verwijderen van de oude software?

Verzekeringspremie

Een oplossing is denk ik wel mogelijk. Waarom introduceren we bijvoorbeeld niet de verwijderingsbijdrage voor software? Een vast bedrag of percentage van de originele ontwikkel- of aanschafkosten dat opzij wordt gezet ten tijde van de aanschaf of ontwikkeling, om latere uitfasering en uitproductiename te financieren? Of misschien in de vorm van een verzekeringspremie voor 'het pensioen' en later 'de uitvaart' van software? Eigenlijk zou één van deze twee varianten bij elke nieuwe implementatie van software in

de projectkosten moeten worden meegenomen, zodat de toekomstige generatie geen last heeft van het ouder, onderhoudsintensiever en minder of zelfs non-functioneel worden van software. Zelfs software heeft recht op een mooi pensioen en een waardige uitvaart en de nieuwe generatie heeft recht op een probleemloze, veelbelovende start om verdere evolutie mogelijk te maken.

Meer “domme” klanten vinden of blijven innoveren?

Geïnspireerd door het boek Karaoke Capitalism en de ontwikkelingen op het gebied van social media heb ik een tijdje terug een paper gepubliceerd over social enterprise platforms. Verder voortbordurend op het thema social enterprise kwamen de volgende vragen naar boven: Moeten we, zoals in Dilbert werd beweerd, meer “domme” klanten vinden? Of is het toch beter om te blijven innoveren? Maar hoe kunnen we ons innovatieproces da n innoveren?

Natuurlijk zijn er altijd méér “domme” klanten te vinden waar we onze karaoke oplossingen (slechte imitaties van anderen producten) aan kunnen verkopen. Maar een best business model is dat niet gebleken. Blijven innoveren is he advies! Innovate, don’t imitate. Traditioneel komt innovatie tot stand binnen de afdeling R&D. Ideeën worden in besloten, van te voren bedachte groepen tot uiting gebracht en besproken, verder vormgegeven en fijngeslepen; innovatie als gestructureerd bedrijfsproces. Hier geldt, zoals terecht opgemerkt in bovenvermeld boek: verbeelding is het grootste goed. Het is de verbeelding die aanleiding geeft tot het ontwikkelen van nieuwe ideeën en uiteindelijke producten die weer leiden tot verdere innovatie en dus de verdere evolutie van de economie en de mensheid. Verbeelding is het grootste kapitaal. Maar waarom is alleen de verbeelding van de mensen in de R&D club van belang? En was dit wel zo’n creatief proces wat met deze verbeelding verder aan de haal ging? Zou het niet veel beter zijn als iedereen, ongeacht rang of stand en plaats in de hiërarchie kon deelnemen aan het innoverende vermogen van een organisatie? Moet het innovatieproces zelf niet eens innoveren?

Any place, any time, any device

Vaak is het zo dat meerdere aardige of zelfs al goede ideeën, indien samengevoegd een briljant idee vormen. Door iedereen input te laten leveren, ongeacht invalshoek of uitgangspunt of op het eerste gezicht beperkte (of helemaal géén) relevantie, kunnen betere ideeën ontstaan en dus effectievere oplossingen voor problemen worden gevonden. In een globale economie en met organisaties waar het nieuwe werken, en dus any place, any time, any device gemeengoed is geworden is het gebruik van digitale social media achtige tools binnen de (virutele) bedrijfsmuren de belangrijkste voedingsbodem om te kunnen blijven innoveren. Door de kennis die op deze manier gedeeld wordt te borgen en hergebruiken ontstaat een echte, breed gedragen kennismanagement omgeving. Door iedereen er aan deel te laten nemen ontstaat een ware social enterprise, waar elke medewerker integraal, volwaardig onderdeel is van de onderneming en dus veel meer bijdraagt aan het uiteindelijke succes.

Bij het implementeren van social media tools binnen de bedrijfsmuren, gaat het vaak niet om de technologie (natúúrlijk kunnen we niet zonder) maar veel meer over de adoptie; het integrale gebruik van dit soort tooling moet onderdeel worden van de genen van de medewerkers en het management. De grootste fout die gemaakt kan worden is te veel focus op technologie en te weinig op adoptie. Door niet te grote stappen in één keer te nemen en het juiste implementatie- en adoptiemodel te kiezen (top down of bottom up, afhankelijk van de volwassenheid van de organisatie) kan een traditionele enterprise een social enterprise worden. Houd er echter rekening mee dat zowel Rome als de social enterprise niet in één dag zijn gebouwd.

Experimenteren

Een social enterprise stelt mensen in staat te participeren in een netwerk om kennis op te bouwen zonder dat men ergens op afgerekend word. Mensen kunnen bijdragen leveren die daadwerkelijk meegenomen worden in de overwegingen om tot een oplossing van een probleem te komen. Of niet, maar daar wordt niemand op afgerekend. Deze manier van bijdragen leveren leidt tot een opener gemeenschap van werknemers, waarin men accepteert dat er fouten kunnen worden gemaakt en men elkaar respecteert. Mensen kunnen in een open sfeer vragen. Experimenteren krijgt de ruimte.

Het creatieve proces wordt veel efficiënter en effectiever; er wordt sneller een oplossing voor problemen gevonden. Zo'n open sfeer draagt bij aan een hogere medewerkerstevredenheid. Hoe gemakkelijker werknemers de juiste sociale tools kunnen gebruiken, hoe gemakkelijker zij kunnen communiceren. Zij kunnen ideeën verbeteren of aanscherpen, snel vragen beantwoorden en vrijelijk informatie delen. Dit leidt tot behoud van kennis in de organisatie, snellere innovatie en dus betere producten en uiteindelijk meer tevreden klanten en medewerkers. Is dit niet véél beter (en leuker!) dan het vinden van meer "domme" klanten?

You can have any color as long as it is black

De uitspraak (van Henry Ford) in de titel in relatie tot SaaS zal veel herkenning oproepen. Maar, is het wel zo slecht? Is het bieden van een standaard oplossing die hetzelfde doet voor iedereen, een one-size-fits-all wel zo erg als men doet lijken?

Veel organisaties zijn eigenwijs als het gaat om IT oplossingen. De business heeft bepaalde wensen en eisen maar wordt niet geremd door enige “logische” IT kennis en is ook vaak niet in staat om processen te reduceren tot gestandaardiseerde, herhaalbare reeksen van taken. Metadenken is best lastig. De mensen op de werkvloer zijn gewend om op een bepaalde manier (binnen processen) te werken en zien niet in waarom dit anders zou moeten, tenzij ze daardoor minder vervelend werk hoeven te doen. Het is echter niet zo zeer het feit dat mensen niet van verandering houden; men wil niet veranderd wórden. Het resultaat is dat het aan te schaffen ERP, CRM of willekeurig ander administratief systeem vaak moet worden aangepast aan de specifieke wensen en eisen rond processen binnen de organisatie.

Standaardisatie

Het mooie echter van standaard producten zoals SAP, Microsoft Dynamics, Salesforce, etc. is dat er vaak een enorme hoeveelheid denk- en standaardisatiewerk aan vooraf is gegaan. De beste manieren om bepaalde processen uit te voeren zijn in de vorm van templates in dit soort producten ondergebracht. Waarom zou je daar van afwijken? Alleen maar om het de eigenwijze mensen in de business en op de werkvloer naar de zin te maken? Weten zij hoeveel het kost om dit soort producten aan te passen aan de (kromme) werkwijze die het resultaat is van deze aanpak?

Het wiel is sinds de oorspronkelijke inceptie al miljarden keren opnieuw uitgevonden en het is er vooralsnog nog steeds niet ronder op geworden. Is het niet veel goedkoper en efficiënter om de organisatie aan te passen aan de software dan de software aan te passen aan de organisatie? Zoals in vele eerdere publicaties al aangehaald communiceert de IT niet goed met de business en vice versa. Dat zal ook niet zo snel gaan veranderen, zolang IT systemen door abstracte code moeten worden “geconfigureerd” (wat is eigenlijk het verschil met “coderen”?) en mensen elkaar nog niet eens begrijpen als ze dezelfde taal met elkaar spreken, laat staan als er een tussentaal nodig is!

Integreren

Door middel van BPM en case management-achtige oplossingen en een op (redelijk door “gewone” mensen te begrijpen) Business Rules gestoeld systeem kan men nog steeds aan finetuning doen en op die manier de geautomatiseerde processen nauw laten aansluiten op de voor de organisatie specifieke aspecten. De blauwdrukken van de processen komen van de plank, de finetuning zit hem in de Business Rules en de inpassing van de processen in de organisatiestructuur.

Als we op die manier naar automatisering gaan kijken, is het helemaal niet meer zo raar om kern bedrijfsprocessen ook in SaaS oplossingen te gaan laten landen. En waar je dan als organisatie wat extra functionaliteit op bepaalde plekken wil, integreeer je gewoon met andere SaaS applicaties of bepaalde Data Services of *loosely coupled* apps. Op die manier kan je toch nog applicatie-overstijgende bedrijfsprocessen inrichten, zonder een bak maatwerk. Zó verschillend zijn we allemaal echt niet! Ik zeg “no code”... Meneer Ford was zo gek nog niet.

Oplossing op zoek naar een probleem?

In “onze” IT wereld ontwikkelen we met enige regelmaat tunnelvisie. De timmerman die alles als een spijker behandelt is een herkenbare analogie. De klant daadwerkelijk van een oplossing voorzien die het beste aansluit bij de business blijft een uitdaging.

Elk nieuw IT paradigma wordt aangegrepen om te verkondigen dat dit alle problemen gaat oplossen. Het resultaat is dat bijvoorbeeld ERP en CRM als applicaties worden ingezet waarvan de geïmplementeerde functionaliteit vaak (ver) buiten het domein van ERP en CRM vallen. Hierdoor ontstaan mega silo oplossingen die niemand meer begrijpt. Ook zijn vendor lock-in en vrijheid van hosting grote problemen bij dit soort uitdijende omgevingen.

Zwitsers zakmes

Hetzelfde geldt voor platforms voor document management en (sociale) samenwerking. De leveranciers (en ook meestal de implementatiepartners) van dit soort oplossingen hebben nogal de neiging om het als Zwitsers zakmes te verkopen en implementeren. Dé oplossing die alle problemen van uw organisatie in één klap oplost. Maar met een Zwitsers zakmes kunt u geen boom omzagen, tenminste, niet binnen afzienbare tijd.

Zo werkt het natuurlijk meestal niet. Analisten als Gartner en Forrester trappen best vaak deuren open die al een tijdje open waren, maar in een aantal dingen blijken ze toch wel erg goed. De hypecycle van Gartner is zo’n voorbeeld van een bruikbare tool. De adoptie van IT themas wordt hier genadeloos mee gemeten. En het mooie is dat elk thema ècht wel door die *trough of disillusionment* heen gaat; hier wordt dan bekend waar het wel en waar het niet goed voor is. Overigens is de hypecycle ook op niet IT themas goed toe te passen: Ik denk persoonlijk dat Adam Smith’s kapitalisme op dit moment door deze *trough* heengaat, maar dat terzijde.

Ook de cloud is zo’n thema. We weten allemaal (in onze onderbuik in ieder geval) dat dit een belangrijk fenomeen is. Analogieën met o.a. water uit de kraan en stroom uit het stopcontact te over. Natuurlijk is het allemaal niet zo simpel. In IT is niets simpel en zal het waarschijnlijk ook nooit echt worden. Maar dat het gaat helpen om bepaalde uitdagingen rond schaalbaarheid, mobiele ontsluiting en niet te vergeten software als service en (externe) integratie gaat oplossen moge duidelijk zijn. Waar het ook erg goed voor is, is de verdere evolutie van de IT’er. Als “banale” zaken als infrastructuur en technisch beheer worden opgelost in de cloud, kunnen we weer een stapje hoger op de IT ladder en meer over oplossingen voor de business gaan praten. Daadwerkelijke oplossingen die de business verder helpen.

Dé oplossing

De cloud is één van de “tools” die we in onze gereedschapskist hebben zitten om oplossingen voor de klant te implementeren. Luisterend naar de klant, helpen we de business beter te opereren door oplossingen te creëren die goed aansluiten bij de vraag. Niet door de cloud, of een samenwerkingsplatform of CRM als “dé oplossing” te positioneren voordat het probleem geheel bekend is. Laten we niet wéér in die valkuil trappen. De klant is het beste gebaat bij een IT thema onafhankelijk advies. Liefst natuurlijk wel gebaseerd op bestaande (branchespecifieke) templates, maar niet in het keurslijf van die templates geforceerd. En

altijd met inachtneming van bewezen, leverancieronafhankelijke best practices op het gebied van business- en informatiearchitectuur. En waar nodig gewoon applicaties en diensten integreren.

Probleem op zoek naar een oplossing, en anders niet!

Rondrennende mannetjes en sla

In een requirements workshop vandaag bij een organisatie in de financiële dienstverlening was één van de onderwerpen dat management het fijn vindt om bij incidenten rondrennende mannetjes te zien die druk bezig zijn om een IT probleem op te lossen. Het kantoor had ook opmerkelijk veel glazen wanden.

Het zien van deze mannetjes geeft het geruststellende idee dat er aan het probleem gewerkt wordt en dat er waarschijnlijk snel een oplossing zal komen. Net als dat er bij de betere koffiemachines altijd bonen te zien zijn (dat moet wel goede koffie zijn) en BI dashboards vaak kleurrijke en bewegende meters bevatten, vertrouwen mensen veelal op visuele informatie. Op het zelfde kantoor hing op de afdeling IT operations ook een aantal grote flatscreens met daarop allerlei statusinformatie in groene, oranje en rode balken en taartpunten. Ik vermoed dat het niet door de IT'ers op de afdeling zelf gebruikt wordt (die krijgen hopelijk pro-actieve alerts), maar eerder daar is voor de managers 'by walking around' en voor de incidentele rondleiding van zakenpartners of klanten, om indruk te maken en om de indruk te wekken dat alles onder controle is. Maar het fijne voor management is wel dat ze te allen tijde inzicht kunnen hebben in de status van de IT operatie. Incidenten kunnen ook door hier en daar wat (mondelinge) instructies te geven van prioriteit veranderd worden indien nodig. Men heeft het gevoel volledig "in control" te zijn. Want men is er bij.

Korte lijntjes

Net als bovengenoemde organisatie, worstelen vele organisaties mede hierdoor met het volledig uitbesteden van bepaalde delen van de IT infrastructuur en applicaties. Bij het uitbesteden naar een hosting provider waar je een directe relatie mee hebt, zijn de lijntjes nog steeds kort en direct. Vaak lopen de mannetjes van de derde partij zelfs gewoon op de eigen werkvloer rond en kun je ze dus nog steeds zien. En beïnvloeden. Voor private cloud geldt het zelfde. Echter, bij het uitbesteden van infrastructuur en applicaties naar een publieke cloud neemt het abstractheidsniveau enorm toe. De mannetjes rennen ergens rond waar je ze niet kan zien. Rennen ze wel rond? Of lopen ze op hun sloffen? Is jouw probleem wel belangrijk genoeg of krijgt een andere tenant meer aandacht? Wordt het probleem wel zo snel mogelijk opgelost?

Loslaten

Door middel van een service level agreement (sla) kan een hoop onzekerheid en frustratie weggenomen worden. Het besef dat incidenten binnen een bepaalde tijd worden opgelost, afhankelijk van de categorie, geeft wat geruststelling. Rapportage van incidentstatus en voortgang door middel van een dashboard helpt ook. De dashboards zullen ongetwijfeld ook steeds informatiever en meer real-time worden. We zullen echter met z'n allen moeten wennen aan het idee van "loslaten". Het niet meer in directe "control" zijn zal niet voor iedereen even gemakkelijk los te laten zijn. Het devies is: Vertrouwen op sla en de cloudleverancier er mee om de oren slaan als het niet nagekomen wordt. En misschien helpt het als er in de "control rooms" van de cloud datacenters videocameras worden opgehangen en dat je via augmented reality kan zien of het betreffende rondrennende mannetje met jouw ticket bezig is... </sarcasm off>.

SaaS en Scrum, vrienden voor het leven

De wereld van de IT verandert de laatste tijd snel. Heel erg snel. Zo snel dat we het niet allemaal meer bij kunnen houden soms. Dit geldt voor zowel IT'ers als eindgebruikers. Eén ding staat als een paal boven water: Cloud Computing gaat niet alleen over uitbesteding van computers, software en beheer. Het is een paradigma verandering die *alles* verandert in onze wereld.

Cloud computing is natuurlijk een gemakkelijke, maar tevens verwarrende paraplu benaming voor een aantal zaken die te pas en te onpas door elkaar worden genoemd.

Nog even in het kort:

- IaaS (infrastructure-as-a-service) is leuk, maar weinig vernieuwend en spannend; virtual machines kunnen we ook gewoon bij onze aloude hosting provider draaien.
- PaaS (platform-as-a-service) is al een stuk interessanter, met name om je maatwerkoplossingen te laten landen en gebruik te maken van out-of-the-box platform mogelijkheden op het gebied van databases en middleware om zodoende sneller oplossingen te kunnen ontwikkelen.
- SaaS (software-as-a-service), natuurlijk veruit het belangrijkste deelgebied in cloud computing; multi-tenant software, voor je gehost en beheerd in de cloud en waarmee je zeer snel oplossingen kunt configureren en in productie nemen.

Combinaties van SaaS

Oervormen van SaaS, zoals Outlook.com of Dropbox.com waren en zijn ook nog steeds bruikbaar, maar eigenlijk ieder een one-trick-pony. SaaS is pas echt interessant als het gaat om applicaties die uitgebreid geconfigureerd kunnen worden om zich aan te passen aan je eigen business wensen en eisen, zodat echte bedrijfsprocessen ondersteund of zelfs uitgevoerd kunnen worden. Denk hierbij aan CRM, ERP, BPM en ECM systemen. Combinaties van deze SaaS oplossingen zijn door middel van integratie ook mogelijk en bieden daarmee nóg meer mogelijkheden.

Rond het inzetten van deze vormen van SaaS, waarbij echte (delen van) primaire of secundaire bedrijfsprocessen kunnen worden geautomatiseerd en geoptimaliseerd nemen we in de praktijk op dit moment drie belangrijke dingen waar:

1. Er is een duidelijk trend richting "standaard-tenzij" oplossingen aan het ontstaan.
2. De business is véél directer betrokken bij het inzetten van deze oplossingen.
3. Goed genoeg is beter dan perfect (denk aan de 80/20 regel).

Minder IT-feestje

Automatisering wordt steeds minder een IT-feestje. SaaS is voor de business. Tegelijkertijd zien we in een trend om weg te bewegen van de traditionele waterval aanpak voor het uitvoeren van ontwikkel- en implementatieprojecten. Meer en meer worden deze projecten agile uitgevoerd en met name Scrum is hierbij een populaire vorm. Niet alleen bij software ontwikkeltrajecten, maar zelfs op IT support- en beheerafdelingen zien we dit de laatste tijd ook ontstaan. Het aanpakken van allerlei kleine of grote projecten door middel van een goed op elkaar ingespeeld team dat in een aantal korte of langere sprints

werkt naar een “prima” oplossing voor een business probleem, in nauwe samenwerking met die zelfde business.

Het creëren van geautomatiseerde oplossingen voor bedrijfsprocessen op basis van SaaS platforms en in de vorm van Scrum trajecten blijkt in de praktijk erg goed te werken. SaaS leent zich bij uitstek voor het snel leveren van “prima” resultaten. Scrum is hierbij de ideale aanpak. Voor het grootste gedeelte ontstaan oplossingen door middel van configuratie. Er wordt zo veel mogelijk gebruik gemaakt van de out-of-the-box mogelijkheden. Men integreert verschillende SaaS producten waar opportuun. No code. Tenzij het écht niet anders kan. Maar telkens moet de business de afweging maken of het het echt waard is om precies dat stukje functionaliteit te creëren waardoor het traject en ook de oplossing zelf mogelijk een stuk complexer wordt en upgrade naar een nieuwere release moeilijker wordt. De consequenties moeten telkens helder zijn. Bijsturing kan snel. Hiervoor bieden SaaS en Scrum de perfecte combinatie: SaaS en Scrum, vrienden voor het leven!

Ook het bouwen van een luchtkasteel vergt een goede architect

Deze uitspraak door de Vlaamse schrijver en dichter Karel Jonckheere (1906-1993) is, zonder dat hij dat waarschijnlijk bedoelde ook van toepassing op IT architectuur. Het is de taak van de architect om te zorgen dat de soms (veel) te hoog gegrepen wensen en eisen van de business toch zo goed als mogelijk op termijn te verwezenlijken zijn.

De geoefend IT architect heeft hiervoor een aantal handige gereedschappen in zijn gereedschapskist. Zelfs met de komst van “enge”, onvoorspelbare dingen als Cloud, Agile, Bring Your Own en Apps zijn er architectuurprincipes toe te passen die beheersbare en veilige groei van de functionaliteiten en dus mogelijkheden faciliteren. Niet alles is natuurlijk van te voren te voorzien, maar de risico’s kunnen wel beperkt worden en een zo hoog mogelijke flexibiliteit is een belangrijk nastreefbaar doel.

Standaardisatie en modularisatie

De uitdagingen met de modernste vormen van IT, bijbehorende methodieken en andere gerelateerde verschijnselen zijn als volgt samen te vatten:

- Cloud computing is eng omdat delen van het IT landschap niet meer binnen de eigen bedrijfsmuren en niet meer in eigen beheer en dus niet meer onder eigen controle draaien; meer flexilibiteit leidt tot minder controle.
- Bij het ontwikkelen van nieuwe toepassingen op de Agile manier is de business sterk betrokken en de drijvende kracht achter de oplossing. Het grootste gevaar wat hier op de loer ligt is dat er een oplossing gecreëerd wordt die volledig is gericht op een korte termijn doelstelling van het management, maar die niet wendbaar is; weer een nieuwe silo applicatie.
- Bring Your Own heeft als grootste uitdaging dat bestaande toepassingen ontsloten moeten kunnen worden op allerlei zelf meegebrachte devices. Veiligheid is hier in het geding. Hoe zorgen we dat de geheime bedrijfsinformatie via deze devices niet op straat komt te liggen bij verlies of diefstal van het apparaat?
- Het fenomeen Apps is de grootste nachtmerrie van elke IT manager. Het zorgvuldig opgebouwde applicatielandschap wordt verstoord door deze vorm van “schaduw IT”. Hier zijn veiligheid en auditability de grootste problemen. Veiligheid is moeilijk af te dwingen, omdat de Apps buiten de eigen vertrouwde infrastructuur draaien. Auditability is misschien een nog wel groter probleem; hoe kunnen we bewijzen dat een document de organisatie heeft verlaten of dat er een stap in het proces is genomen en door wie?

Architectuur gaat in de IT met name over standaardisatie en modularisatie. Beide onontbeerlijk bij het aanpakken van bovenstaande uitdagingen. Standaardisatie helpt bij het beter kunnen beveiligen van de informatiestromen en het goed kunnen uitwisselen van data tussen systemen en diensten. Modularisatie helpt bij het voorkomen van silo’s en het snel kunnen inspringen op veranderende (markt)omstandigheden.

Innovatie versnellen

Vaak wordt de IT architect verweten dat hij een remmende werking heeft op innovatie. Dat is misschien wel eens zo, maar dan betreft het een slechte IT architect. Een goede architectuur is juist bij uitstek in staat om innovatie te versnellen, immers: Innovatie ontstaat vaak door het slim combineren van technologiën. Standaardisatie en modularisatie zijn hierbij van het grootste belang. We kunnen zelfs het tegenovergestelde beweren: IT oplossingen die *niet* onder architectuur worden ontwikkeld beperken of verhinderen innoverende slagkracht op de middellange tot lange termijn.

Luchtkastelen bouwen is misschien niet zo heel zinnig, want laten we realistisch blijven; zo mooi als het door de business bedacht werd wordt het in de praktijk nooit. Maar, de lat hoog leggen en de juiste architectuur neerzetten om zo ver mogelijk te kunnen komen is in het geheel niet dom. De inzet van een ervaren IT architect is hierbij een noodzakelijk goed.

Achteruitautomatiseren

Tijdens een workshop rond business achitectuur bij een organisatie die zich bezig houdt met re-integratie van arbeidskrachten probeerde ik het belangrijkste primaire proces "indiensttreding" in kaart te brengen met als doel het verregaand te gaan digitaliseren. Eén van de workshop deelnemers nam daarbij het woord "achteruitautomatiseren" in de mond.

Een grappig woord! De betreffende persoon had het over een aantal ervaringen die hij tot nu toe had meegemaakt en de trend die hij daar in zag. De intentie van de workshop en het daaruit volgende implementatietraject hebben vanzelfsprekend die trend doorbroken; eindelijk weer in z'n vooruit!

Functionaliteit

Toch is het wel interessant om eens dieper op dit woord in te gaan. Achteruitautomatiseren kan namelijk op vele manier worden geïnterpreteerd, afhankelijk van het perspectief. Laten we er eens een aantal varianten van op een rijtje zetten:

- **Nieuwe software, minder functionaliteit**: Hierbij wordt een nieuwe software implementatie uitgevoerd, waarbij de uiteindelijke geboden functionaliteiten minder zijn dan in de situatie er voor. Redenenen kunnen zijn: door noodzakelijke kostenbesparingen is gekozen voor goedkopere software (of SaaS), die minder functionaliteiten biedt, of de uiteindelijk opgeleverde implementatie van maatwerksoftware levert niet wat er van verwacht werd.
- **Nieuwe software, minder beheersbaar**: Hierbij is de geboden functionaliteit minstens zo goed als bij de vorige software, echter is de nieuwe omgeving niet meer zo goed beheersbaar en moet er veel meer tijd gestoken worden in het in de lucht houden van het pakket, wat weer veel extra menselijke inspanning kost.
- **Nieuwe software, minder flexibel**: De nieuwe softwareimplementatie biedt minstens net zo veel op functioneel vlak als de vorige versie, echter men heeft een silo gecreëerd die niet (makkelijk) uitbreidbaar is of die niet goed kan aansluiten op andere producten of diensten.

Dit zijn maar drie varianten, maar er zijn natuurlijk nog legio andere vormen van achteruitautomatisering te identificeren. Leuke exercitie voor een grijze zondagmiddag.

Architectuur

Twee vormen van achteruitautomatiseren wil ik hier toch wel met name onder de aandacht brengen, omdat die er wat mij betreft de laatste tijd héél erg uitspringen met de komst van web, cloud computing en mobile:

1. **User interface anarchie**: Waar moet je tegenwoordig klikken? En waar kan je dingen ingeven? Wat is een toets? En hoe scroll je? In de tijd van client / server met Windows client applicaties was er, mede dankzij de formidabele inzet van Microsoft op het gebied van standaardisatie, eenduidigheid rond user interface componenten. Die is tegenwoordig met web based user interfaces en apps ver te zoeken en dit leidt vaak tot gefrustreerde gebruikers. Wanneer staat hier eens een (onafhankelijke) partij op en komen er goede standaarden?

2. **Thick apps**: Apps zijn cool. En handig. Behalve als je er honderderden op je tablet of phone hebt en ze zijn ook nog eens heel dik. Apps die tegen de 100% van de functionaliteit in de app zelf hebben zitten zijn veel te log en moeten continu geüpdatet worden op al die honderduizenden of miljoenen devices omdat de functionaliteitswensen maar uitgebreid worden en de kans op bugs groot is. De (business)logica moet weer terug de middle tier in, centraal beheersbaar, schaalbaar en gemakkelijk up-to-date te houden! Thin apps moeten weer de norm worden.

Als men ontwikkelaars zonder architectuur(kennis) aan het werkt laat gaan ontstaan bovenstaande situaties al snel. Zelfs met een Agile aanpak kan dit overigens ook prima voorkomen worden. Een aantal simpele architetuuruitgangspunten moeten aan de start van elk traject helder omschreven worden en bij iedereen in postervorm aan de muur hangen. Laten we in de toekomst op het gebied van IT weer eens echt 3 stappen vooruit gaan, in plaats van 3 stappen vooruit en 2 achteruit.

Beschikbaarheid en schijnveiligheid

Bij het ontwerp van een technische architectuur voor een applicatie of platform zijn de non-functional oftewel quality of service requirements van groot belang. Zeker als het om een bedrijfskritische omgeving gaat. Te vaak zien we in de praktijk echter dat ondanks de juiste specificaties, procedures en afspraken er toch van alles misgaat in geval van een rampsituatie.

Waar zit hem dat nou in? Laten we eerst eens kijken naar belangrijke aspecten van beschikbaarheid en wat daar bij komt kijken. Beschikbaarheid wordt meestal uitgedrukt in zogenaamde uptime. De volgende regels geven voor de meest voorkomende uptime percentages aan wat de bijbehorende downtime per jaar is:

- 99% - 3,65 dagen
- 99,5% - 1,83 dagen
- 99,8% - 17,52 uur
- 99,9% - 8,76 uur
- 99,99% - 52,56 minuten
- 99,999% - 5,26 minuten
- 99,9999% - 31,5 seconden
- 99,99999% - 3,15 seconden

U ziet, vanaf “4 nines” wordt het echt serieus! Naast uptime worden de termen RTO (recovery time objective) en RPO (recovery point objective) gebruikt. De eerste geeft de maximale tijd aan, waarbinnen het systeem na een ramp weer beschikbaar moet zijn. De tweede geeft het maximale verlies aan data aan. Veelal uitgedrukt in minuten. Dus bijvoorbeeld bij een RPO van 15 minuten en een RTO van 60 minuten moet na een ramp het systeem weer binnen 60 minuten beschikbaar zijn en er mag maximaal 15 minuten aan data verloren zijn gegaan.

Beschikbaarheidsafspraken

Bij de veruit meeste organisaties waar ik over de vloer kom, is een uptime van 99,8% of 99,9% gebruikelijk. Microsoft’s Azure PaaS platform garandeert bijvoorbeeld voor de meeste daarop geleverde diensten 99,9% beschikbaarheid. In dat geval is Microsoft verantwoordelijk voor het inderdaad leveren van die beschikbaarheid, volgens de service level agreement (SLA). Indien de spullen bij een hosting provider draaien, is de hosting provider veelal verantwoordelijk voor het garant staan voor de afgesproken beschikbaarheid. Boetes of andersoortige compensatie bij het niet halen van de beschikbaarheidsafspraken staan beschreven in de algemene voorwaarden van de leveranciers. Meestal is het hoogste compensatiebedrag gelijk aan de gebruikelijke maand of jaarfactuur. Voor gevolgschade is men niet aansprakelijk.

Indien uw IT organisatie zelf (deels) verantwoordelijk is voor het hosten van de applicaties of het platform heeft u zelf nog wat meer verantwoordelijkheid. Zeker als technisch beheer bij u ligt. Maar vergeet niet dat gemaakte fouten door functioneel (applicatie) beheerders ook effecten kunnen hebben op de uptime van uw applicaties. Hier heeft de partij waar de applicaties gehost worden niets mee van doen. De SLA’s

afgesproken met uw hosting provider of cloudleverancier bieden, als u kijkt naar de gehele keten waarin het fout kan gaan, dus een stukje schijnveiligheid. Het aloude spreekwoord geldt hier ook: uw keten is zo sterk als de zwakste schakel. Waarbij in dit geval de zwakste schakel vaak óf de mens óf slechte (test) procedures zijn, die ook door mensen zijn opgesteld.

Sneeuwbaleffect

Het wel of niet beschikbaarheid zijn van een software oplossing zit hem veelal in kleine dingen. Dingen die bijvoorbeeld niets met hardware of netwerk te maken hebben. Dingen die vaak afhankelijk van elkaar zijn. Die ook vaak een sneeuwbaleffect tot gevolg hebben en kunnen leiden tot een complete meltdown. Denk aan een volle schijf op een database server. Nog steeds leidt dit in de meeste gevallen tot zéér onvoorspelbaar gedrag van applicaties en het weer terug naar normaal krijgen valt in dat soort situaties echt niet mee. Hierdoor is de RTO vaak een echte uitdaging. Het goed, pro-actief monitoren van het platform is hierbij cruciaal. Voordat iets een echt probleem gaat worden moet je al op de hoogte worden gesteld. De inzet (en juiste configuratie) van monitoring tools en het op de juiste manier melden en escaleren van incidenten helpen om de uptime hoog te houden. En niet te vergeten: de procedures (inclusief disaster recovery) moeten regulier getest worden!

Tot slot, vergeet ook niet dat de uptime niet alleen belangrijk is voor productieomgevingen. In sommige gevallen is de uptime van ontwikkel- en testomgevingen misschien nog wel belangrijker, zeker als er een leger aan inhuurkrachten op deze machines werkt. Wat kost het u wel niet aan verloren productietijd per uur als een aantal Scrum teams tegelijk werkeloos toe moeten kijken in uw tijd?

Een oplossing zonder beveiliging is geen oplossing

Wel eens het boek “Writing secure code” (Microsoft Press) gelezen? Nee? Schande! Zelfs als u geen ontwikkelaar bent zou u het moeten lezen. Dit boek was destijds voor mij een echte eye opener op het gebied van beveiliging van IT componenten en werd door Bill Gates op de must-read lijst voor Microsoft medewerkers gezet.

En terecht. In veel gevallen wordt het beveiligen van software oplossingen behandeld zoals we documentatie en testen behandelen: we doen het als er tijd overblijft in het project. En we hebben er eigenlijk niet zo veel verstand van of zin in. Maar zo werkt het niet! Zeker niet meer sinds de komst van het internet en al helemaal niet meer sinds de komst van mobile en apps. De keten met informatie is zo veilig als de zwakste schakel. Elke component in de keten moet dus met veiligheid voorop ontwikkeld worden. Het boek, en de vele andere resources op dit gebied zullen hierbij uitstekend van dienst zijn. Verplicht leesvoer dus voor iedereen die in de IT acteert. Dit is ook een onderdeel van het vakgebied waar continu in rap tempo verder geleerd kan worden.

Open source

Een tijdje terug werd bekend dat er een lek in OpenSSL zit; de zogenaamde heartbleed bug (zie heartbleed.com). Deze bug is door een programmeur gecreëerd (een menselijke fout dus) en vervolgens is dit als een olievlek verspreid omdat de halve wereld aan elkaar hangt met internetprotocollen zoals HTTP en dat betekent in de praktijk dat dat in veel gevallen op transport niveau “beveiligd” is door middel van de OpenSSL implementatie. Grappig was dat Microsoft trots melde dat haar Azure cloud platform hier geen last van had, behalve de Linux VM’s die op de IaaS laag draaien. Omdat Azure zelf niet op open source gebaseerd is en deze bug niet in de Microsoft closed source implementatie van SSL haar weg heeft gevonden. Andere programmeur. Met meer geluk?

Nee, dit heeft niets met geluk te maken. Dit gaat om het ontwerpen en ontwikkelen van software met de best mogelijke veiligheid als doel door middel van threat model analysis en strenge (deels te automatiseren) source code controle. Hiermee kan vroegtijdig (vaak al tijdens ontwerp) in kaart worden gebracht waar de kwetsbaarheden zich (zullen) bevinden en wat er aan gedaan kan worden om die kwetsbaarheden zo veel mogelijk te voorkomen of, in geval van “inbraak” de negatieve effecten zo veel mogelijk te minimaliseren; als iemand dan al kan inbreken op een proces, dat hij of zij niet de rest van het systeem kan benaderen bijvoorbeeld.

Security by design

Het is de plicht van elk software architect en engineer om veiligheidsrisico’s en bijbehorende softwarematige oplossingen vroegtijdig inzichtelijk te maken en als onderdeel van het ontwikkelproces gelijk aan te pakken. Hiervoor moet dus ook ruimte gemaakt worden in de projectbudgetten en –planning. Dit valt onder het kopje “architectuur”. Voorkomen moet worden dat het oplossen van veiligheidsproblemen door middel van refactoring in een agile team wordt gedaan. Iemand heeft ooit gezegd “you cannot test security into a product” en daar heeft de man (of vrouw) helemaal gelijk in! En

voor agile geldt zeker ook: alles learnings op het gebied van security threats en issues worden in volgende sprints gelijk weer meegenomen.

Natuurlijk is het ook een kwestie van gecalculeerd risico in de meeste gevallen. Maar wat gebeurt er als bijvoorbeeld een component herbruikt wordt in een product of proces met een veel hoger risicoprofiel? Liggen dat soort gevaren niet ook op de loer? Zeker in een meer en meer service oriented cloud wereld, waarbij diensten van over de hele wereld onderdeel kunnen uitmaken van uw totaaloplossing is het zaak om ook daar goed rekening mee te houden in uw threat models. We hebben het gezien met OpenSSL en heartbleed. Iedereen ging er vanuit dat dit veilig was, maar dat viel toch even tegen. En lees maar in de nieuwberichten wat het kost om dit op te lossen. Nog afgezien van de schade geleden door alle gelekte informatie. Want door een lek in dit ene component, lag even bijna alle informatie op het internet voor het oprapen.

Overigens zou ik me ondertussen als open source software gebruiker toch zorgen gaan maken. De oproep tot donaties naar aanleiding van heartbleed heeft bij lange na niet het gewenste effect gehad. Er is dus niet genoeg geld om dit soort problemen in de OpenSSL stack in de toekomst te voorkomen. Aldus de programmeur die de fout had gemaakt: 'Het is ongelukkig dat de software gebruikt wordt door miljoenen mensen, maar dat er maar een paar mensen zijn die er een bijdrage aan leveren.'

Mijn conclusie is dat veilige software ontwerpen en ontwikkelen serieus werk is, niet goedkoop is en altijd voor een belangrijk gedeelte mensenwerk blijft. En maak bij het gebruik van diensten of componenten van derden dezelfde veiligheidsafwegingen als bij het zelf ontwikkelen van software. Maar bovenal: Oplossingen zonder afdoende beveiliging creëren staat gelijk aan het creëren van geen oplossingen.

Usability – keep IT simple?

Usability is een term die in de IT aangeeft in hoeverre een app(licatie) prettig en efficiënt bruikbaar is en dus eenvoudig geaccepteerd zal worden door eindgebruikers. Dit omvat grafische aantrekkelijkheid en consistentie, duidelijke navigatiestructuur en het vinden en ingeven van informatie op de voor de gebruiker logische plek. Is personalistatie hier het hoogste goed?

In mijn jarenlange ervaring met het werken met software ontwikkelaars is in ieder geval één ding duidelijk: De gemiddelde ontwikkelaar heeft geen kaas gegeten van usability. Daarbij maakt het niet uit in welke taal er gewerkt wordt of hoe oud of jong men is en van welk geslacht. Een Cobol ontwikkelaar heeft er net zo weinig verstand van of affiniteit mee als een ASP.Net of Javascript ontwikkelaar. Laat een ontwikkelaar een scherm bouwen en je kunt er van op aan dat er niet of nauwelijks mee te werken is omdat er niets op een logische plek staat en de “control flow” ook niet je van het is. Eén van de belangrijkste lessen op het vlak van usability is dus: trek usability en technische implementatie los van elkaar. De competenties die er voor nodig zijn, zijn niet te verenigen in één persoon.

Gewoon een vak

Als usability ontwerp dan als competentie is toegevoegd aan het ontwikkelteam, is nog steeds maar de vraag of de betreffende persoon écht wel begrijpt hoe het werkt. Net als bij software ontwikkelaars is het in eerste instantie moeilijk om het kaf van het koren te scheiden. Wie bij software ontwikkeling standaard patronen toepast (omdat men deze door en door kent), de juiste frameworks gebruikt en de principes van SOLID volgt is waarschijnlijk in staat om goed onderhoudbare en goed presterende software te ontwikkelen. Usability ontwerp is echter ook gewoon een vak.

Usability ontwerp is een uiterst visueel ingesteld vak. Maar ook een vak waarbij men met menselijke emoties te maken heeft. En waarbij men overtuigingskracht nodig heeft. Waar harde requirements rond functionaliteiten en kwaliteitseisen best SMART te maken zijn en daardoor goed te valideren zijn tijdens het testproces, is het bij usability toch anders. Daar moet enerzijds met de groep eindgebruikers afgestemd worden hoe de schermen en navigatiestructuur er uit moeten komen te zien (consensus bereiken!), maar anderzijds moet ook de ervaring op het gebied van echt bruikbare gebruikersinterfaces “doorgedrukt worden”; vertel de klant wat hij wil!

Het bereiken van consensus is al lastig. Vaak verzandt dat dan in oeverloze discussies en besluit men uiteindelijk om dan maar een hoge mate van personalisatie mogelijk te gaan maken. En daar begint vaak de ellende. Want, wees ‘ns eerlijk, hoe irritant is het niet als in “jouw versie” van een app de knop om de declaratie in te dienen groen is en linksboven zit en bij de versie van je collega is hij blauw en staat hij rechts onderin. Dit is niet alleen lastig qua uitwisselbaarheid en samenwerking tussen collega’s, maar ook een nachtmerrie voor de mensen van support en beheer. Ook wordt de onderhoudbaarheid van de software een stuk complexer. En belangrijker nog, vaak wordt de app er niet eens veel bruikbaarder of begrijpelijker door.

Steve Jobs

Nee, het hoogste goed is een simpel ontwerp. Eén ontwerp dat gebaseerd is op een aantal eenvoudig uit te leggen uitgangspunten. Knoppen zien er altijd het zelfde uit en staan altijd op een voorspelbare plek.

Navigatie door schermen is altijd op de zelfde manier van links naar rechts en van boven naar onder. Hyperlinks zijn underlined. Lettertypes zijn eenduidig. Enzovoort. Gewoon niet aan te passen. Maar eenduidigheid, consistentie en eenvoud. Denk maar aan de producten die ontworpen werden onder de bezielende leiding van wijlen usability expert Steve Jobs. Simpel is nog niet simpel genoeg. En iedereen kan er mee werken zonder dat er eerst een handleiding gelezen hoeft te worden. Zelfs een 2 of 90 jarige en zelfs Stevie Wonder kunnen er direct, intuïtief mee overweg. En zo hoort het! Usability is een vak. Een vak wat uitgeoefend moet worden door mensen die van consistentie en eenduidigheid dromen en het KISS principe volgen.

Is de cloud nog wel te bevatten?

Vroeger was het simpel. Je installeerde software en de bijbehorende database lokaal en zorgde dat alleen die data de voordeur verliet waar jij toestemming toe gaf. De software werd ook na een langdurige selectieprocedure aangeschaft en zorgvuldig in productie genomen. Veranderingen werden volgens een releaseplanning doorgevoerd. Voorspelbaar. Betrouwbaar. Hoe anders is het nu!

Weet u nog waar uw data daadwerkelijk wordt opgeslagen en wie het allemaal kan inzien? Wie er het recht op hebben om het in te zien? Of wanneer het gedeeld mag worden met anderen? Weet u of de data op uw smartphone echt alleen op uw smartphone bewaard wordt? Weet u wanneer u toch echt over moet op een nieuwe release van uw SaaS oplossing? Of wat er gebeurt als uw provider failliet gaat? Heeft u een goede exit strategie? Overziet u alle instellingen nog?

Waarschijnlijk niet. Bijna apathisch klikken we eindgebruikersovereenkomsten weg zonder ze te lezen en tekenen daarmee soms misschien wel ons (figuurlijke) doodvonnis. Privacy waakhonden stellen de praktijken van Google, Microsoft, Apple, Facebook en Amazon af en toe aan de kaak, maar zelden leidt dat tot verandering in wat er daadwerkelijk met uw data gebeurt. Grote boosdoener is dat we gewoonweg geen idee hebben waar we ja op zeggen. Maar vervolgens wel ja zeggen. Omdat we verder moeten. De business of de sociale controle drukt door. En daarbij worden we overgeleverd aan een soort van dictatuur. Want vendor lock-in is een serieus issue. Natuurlijk zijn er contracten en SLA's. Maar in de praktijk heeft u eigenlijk geen poot om op te staan, zeker niet als relatief kleine gebruiker. Eigenlijk zou iedereen die gebruik maakt van cloud diensten ook gelijk mede-aandeelhouder moeten zijn, met stemrecht. Maar ja, of dat ooit gaat gebeuren?

Exit onmogelijk?

Natuurlijk is het zo dat mensen van een steeds groter aantal zaken geen idee meer hebben hoe het werkt. Dat komt ook door de evolutie van de mens en de diversificatie. Maar veel van dat soort zaken zijn in de betrouwbare handen van het management van de ondermening waar we voor werken of van 's lands bestuurders. Ik heb geen idee hoe de belastingdienst werkt, maar ik vertrouw er op dat onze regering de boel goed bestuurt en het geld goed verdeelt. En die regering kiezen wij zelf. Min of meer dan. En als we het er echt niet mee eens zijn gaan we demonstreren of staken en als dat niet helpt pakken we ons boeltje op en emigreren we. Dat gaat relatief simpel en snel. Een exit vanuit een cloud naar on-premises of een andere cloud provider zal echter veel meer voeten in aarde hebben. Zeker als er oplossingen zijn gecreëerd die niet op open standaarden zijn gebaseerd. En het wordt al helemaal moeilijk als er allerlei point-to-point connecties zijn gelegd met allerlei apps en devices. Zo'n oplossing is moeilijk te isoleren en verplaatsen. Wat gebeurt er met je sociale leven als je je Facebook account verwijdert?

Regelgeving

Eigenlijk zouden cloud providers als banken moeten worden behandeld. En dat er net als de Nederlandsche Bank een vehikel is om de cloud providers van universele regelgeving en controle te voorzien. Een cloud provider kan bijvoorbeeld ook "too big to fail" zijn. Daar wel eens aan gedacht? En als een organisatie voor de uitvoering van de primaire bedrijfsprocessen volledig afhankelijk is van de diensten geboden door de cloud provider is het snel gedaan als dit te lang plat ligt. Als gebruiker moet u

tegen uzelf in bescherming worden genomen. We moeten er vanuit kunnen gaan dat als we onze applicaties en data aan de cloud toevertrouwen, dit onder goede governance gebeurt. Want het is echt niet zo dat het ons, of zoals weleens gezegd wordt “de jeugd”, niets interesseert wat er met onze privacy gebeurt. Het is wel degelijk belangrijk, en we willen graag zeker weten dat wat er met onze data gebeurt, ook door ons begrepen wordt of anders gebeurt volgens de regels die voor alle cloud providers van toepassing zijn. Op dit moment staat die regelgeving en controle nog steeds in de kinderschoenen. En dit komt ook omdat de regelgevers zelf het soms niet helemaal meer begrijpen.

Tot slot: Facebook staat met 400 miljoen “inwoners” op plaats nummer 3 in de top 10 van grootste landen ter wereld. Maar mogen deze mensen ook stemmen? Is er democratie? Is er een “NAVO” die er voor kan zorgen dat als er een brandhaard ergens aan de grens met Google+ uitbreekt even de orde komt herstellen? Zijn Google en Facebook lid van deze “cloud NAVO”? Moet er geen “cloud G5” of “cloud EU” komen? En een “cloud Amnesty International”? En een "cloud Schengen verdrag" tussen samenwerkingsverbanden van providers? Food for thought.

Supportability first!

Terugkijkend op de jarenlange evolutie van IT systemen, van de ponskaart tot Internet of Things blijf ik mij verbazen over één ding: Elke CIO en IT manager weet dat de aanschafprijs van software of cloud diensten maar een klein gedeelte van de totale kosten is. Het in de lucht houden kost vele malen meer. Waarom kan dit niet beter en dus goedkoper?

Omdat IT oplossingen ook steeds complexer worden, wordt het in de lucht houden van deze oplossingen ook steeds complexer. De introductie van service oriëntatie, cloud computing, apps, micro services en IoT maakt het allemaal niet makkelijker. Waar je voorheen nog gewoon een ERP systeem implementeerde, wat alle functionaliteiten in één mooie monolitische applicatie biedt, bestaat “de applicatie” (is dat nog een valide concept?) tegenwoordig uit vele bewegende onderdelen en aan elkaar “geknoopte” componenten. Sommige delen draaien in je eigen datacentrum, sommige in de cloud en sommige op je mobiele device. Hoe houdt je zulke complexe omgevingen beheerbaar, veilig en voorspelbaar?

Supportability

Tijdens een traject bij een groot Nederlands oliebedrijf leerde ik voor het eerst de term “supportability” kennen. Later kwam die term zelfs terug in de tijdelijke functie die ik daar vervulde. Sr. Supportability Consultant. Wow, mooie titel! Weliswaar vervulde ik die rol alleen in de wereldwijde “integration services” afdeling, maar dat maakte het juist ook cruciaal. Want, hoe worden alle bewegende onderdelen in een complexe “applicatie” aan elkaar geknoopt? Juist, door middel van integratie technologie. En wat gebeurt er als dat niet feilloos werkt? Juist, dan dondert alles om.

Die integratie technologie is tegenwoordig ook niet meer één product dat je even installeert (of aanzet, in het geval van PaaS of SaaS integratie). De integratie technologie is ook hybride geworden, en bestaat uit zaken als API management tot en met een Service Bus met eventueel bijbehorende Master Data Services, enzovoort.

Het valt mij op dat alle grote leveranciers van software en cloud diensten nog veel te weinig aandacht besteden aan de end-to-end governance en application lifecycle management van dit soort complexe, hybride “applicaties”. Elk onderdeel op zich kan (meestal) prima gemonitord en beheerd worden, maar helemaal end-to-end? Nee, niet echt.

Supportability is een term die gebruikt wordt om aan te geven welke aspecten rond het ontwikkelen van oplossingen te maken hebben met het supportable maken van deze oplossingen. Dat wil zeggen, bij het bedenken van de architectuur moet direct al aandacht zijn voor supportability. Bij het maken van een functioneel ontwerp moet daar ook gelijk al aandacht voor zijn. Dat betekent dat er direct vanaf het begin mensen van beheer en support betrokken moeten worden bij het traject. Los daarvan zou het fijn zijn als de leveranciers van software en cloud diensten verder kijken dan hun eigen “containers”. Een end-to-end visie op governance en application lifecycle management is zeer gewenst. Open standaarden op dit gebied zouden erg welkom zijn!

Volwassenheid in de IT

Mijn oproep aan de leveranciers is: in plaats van de zoveelste nieuwe functionaliteit toe te voegen aan de software of cloud dienst (ook cool hoor, daar niet van) wordt het tijd om met oplossingen te komen voor het creëren van supportable hybride “applicaties”, liefst op basis van een mooie open standaard. Een oplossing die aansluit op bestaande standaarden en beheerprocessen als ITIL, maar die ook mooi ingeplugd kan worden in zowel Eclipse, Visual Studio, VMware, Hyper-V, Docker als Xamarin (om maar een paar kandidaten te noemen). Pas als we op deze manier complexe, hybride “applicaties” kunnen gaan ontwikkelen met een uitvoerbare supportability strategie én de daarbij behorende bruikbare tools bereiken we een volgende niveau van volwassenheid in de IT. En dat is hard nodig.

Durven Nederlanders nog wel te innoveren?

Is Nederland als innovatief land een stille dood aan het sterven? Zijn we zo aan het polderen, reguleren en vangnetten geslagen dat niemand meer durft te innoveren of er de noodzaak toe ziet? Dat niemand meer risico neemt en ondernemerschap toont?

We zien om ons heen het ene na het andere Nederlandse bedrijf omvallen. Bedrijven die zich nog aan dinosaurus business modellen vastklampen. Warenhuizen die het raar vinden dat ze links en rechts worden ingehaald en irrelevant worden. Taxicentrales die niet alleen door hun eigen ondermaatse prestaties klanten verliezen maar meer nog door nieuwe initiatieven, en daar boos om worden en dit via wetgeving willen laten verbieden. Hotels die te slecht zijn voor woorden, maar wel drie of vier sterren claimen en dit voor de hoofdprijs durven te leveren en ondertussen business verliezen aan meer innovatieve manieren van verhuren van tijdelijke woonruimte. Muzikanten en zangers die nog steeds over gouden CD's praten en in albums denken. IT'ers die nog steeds denken dat het eigen datacentrum gewoon blijft bestaan en patches zelf geïnstalleerd moeten worden en silo applicaties je-van-het zijn. Dat alles bij het oude zal blijven. En dat hun eigen baan niet zal veranderen.

Wakker worden mensen!

Risicomijdend gedrag

Onlangs was ik voorzitter bij een meeting met zo'n 75 integratiespecialisten. Eén van de presentaties werd door een bekende integratie architect uit Engeland gegeven. Zijn verhaal ging over real-life hybride, mobiele en IoT integratie scenario's. Bij de vraag aan het publiek wie er al wat met "cloud" had gedaan, stak een ongelofelijk aantal van maar 5 of 6 mensen hun hand op. Tijdens wat offline discussies rond met name de achterliggende oorzaken met onder andere de Engelsman en een aantal andere mensen uit het publiek begon er bij mij een redelijk beeld te ontstaan.

Daarnaast ben ik altijd betrokken bij aardig wat IT projecten bij diverse organisaties en spreek daarbij regelmatig IT managers en CIO's maar natuurlijk ook veel mensen aan de vraag kant van IT; de business. Daarbij valt het me telkens weer op dat de meeste mensen in toenemende mate risicomijdend gedrag vertonen. Maar dat is, zoals ook al bleek uit de bovenstaande discussies, niet het enige probleem. Er zijn maar weinig ondeugende managers die het adagium "beter achteraf vergiffenis vragen dan vooraf toestemming" actief toepassen.

Waar Amerikanen en hun Europese evenknieën, de Engelsen veel meer in mogelijkheden en kansen denken en zaken groots aanpakken, vallen wij Nederlanders telkens terug in het mitigeren van risico's en het eerst en voor alles afkaderen, regelen en reguleren van zaken voordat we ook maar ergens aan beginnen. Nederland is mijns inziens ook het land waar ITIL en Prince2 het braafst worden toegepast. Scrummen is eng. Want hoe weten we dan wat we krijgen? Wat is de business case? Maar wat als het niet levert wat we willen? In plaats daarvan moeten we meer leren denken "Maar wat als-ie het wel gewoon doet?".

Fail fast

Fail fast is iets wat door de de pay-per-use cloud en door de bijbehorende agile aanpak van het ontwikkelen van mooie oplossingen juist mogelijk wordt gemaakt. Hoe eerder je weet dat iets niet gaat werken, hoe beter. En weer verder met het volgende experiment. Totdat je (voorlopig dan) weer de juiste oplossing gevonden hebt en de monden weer kunt voeden. Er moet ook gewoon snel waarde leverende functionaliteit worden opgeleverd. Niet gelijk pixel-perfect. Dat komt later wel, indien nodig. We leven tegenwoordig in een beta-software maatschappij. Google doet het zo. Microsoft doet het zo. Gewoon snel nieuwe functionaliteit leveren en laten beta-gebruiken (testen in productie eigenlijk) door klanten, en nog niet zo'n zorgen maken of het wel perfect is en goed te managen is.

Groot denken! Klein beginnen. Denken als bijvoorbeeld Elon Musk. Zoals hij pas zei: "De menselijke bestuurder van een auto is over een paar jaar een veel groter gevaar dan de automatisch rijdende auto". Omdraaien die perceptie! Hoppa!

Al deze nieuwe initiatieven, die in een zeer korte tijd erg groot en belangrijk in ons leven zijn geworden hebben één ding gemeen: cloud computing. Mobile computing is mede hierdoor mogelijk gemaakt. Agile is de aanpak en cloud gebaseerde beta-software door klanten laten gebruiken en als het aanslaat verder uitwerken is het middel. *Fail fast, learn rapidly*. Omarmen die cloud en experimenteren! Niet in risico's denken maar in kansen.

Cloud. Just do IT.

Staat de software ontwikkelaar op uitsterven?

Met de komst van steeds volwassener SaaS en PaaS diensten, die de mogelijkheden bieden om goed werkende oplossingen met "een aantal muis klikken" te creëren, lijkt er een tendens te ontstaan dat de software ontwikkelaar meewarig wordt aangekeken omdat zijn expertise niet meer nodig zou zijn. Staat de software ontwikkelaar op uitsterven?

Laat ik er gelijk maar helder over zijn: Nee! We hebben in de redelijk korte historie sinds de opkomst van IT systemen gezien dat er telkens weer nieuwe paradigma's zijn ontstaan die de productiviteit op het gebied van de ontwikkeling van IT oplossingen hebben verbeterd. Van ponskaarten via assembly language naar derde generatie talen en 4GL's, we hebben het allemaal voorbij zien komen. Telkens zagen we dat de productiviteit op de korte termijn omhoog ging, maar tegelijkertijd zagen we dat de wensen en eisen van de eindgebruikers ook steeds complexer werden en dat de 4GL vaak tekort schoot als het om die 20% van de functionaliteit ging die net die 80% van de toegevoegde waarde creëert. Want laten we eerlijk zijn: als iedereen dezelfde SaaS oplossingen gebruikt, waar zit dan de toegevoegde waarde voor jouw bedrijf? Zit de innovatie niet juist in het toepassen en combineren van nieuwe technologieën waar de 4GL of SaaS oplossing nog *niet* mee overweg kan? Logica die nog *niet* gegenereerd kan worden vanuit een model?

Zwaargewicht software engineers

Daarnaast zagen we dat het ontwikkelen van de standaard ontwikkelplatforms als 4GL's, RDBMS'en en later de modelleeromgevingen voor business procesen (BPM) en mobiele apps ook steeds complexer werd. De zwaargewicht software engineers worden met name ingezet op het ontwerpen en ontwikkelen van dat soort platforms. Wat is er cooler dan het schrijven van code die code genereert?

In de praktijk merk ik ook regelmatig dat code eigenlijk veel overdraagbaarder is dan configuratie. Configuratie werkt alleen maar in een bepaalde context en kan vaak moeilijk zó worden weergegeven dat je het gelijk begrijpt. Code is dan eigenlijk veel gemakkelijker. En ook gemakkelijker te beheren. Maar dat terzijde.

Microsoft heeft rond de lancering van Azure App Service verkondigd dat *iedereen* in staat zal zijn om applicaties te ontwikkelen met behulp van dat platform. Door API's in een web portal aan elkaar te knopen door middel van Logic Apps wordt dit mogelijk gemaakt. Op basis van een Microservices architectuur ontstaan oplossingen die gemakkelijk "aan elkaar geklikt" kunnen worden. Door zoveel mogelijk gebruik te maken van allerlei API's die via de *market place* beschikbaar komen is het bouwen van een oplossing zo gepiept! Het is in feite één grote Lego doos, met allerlei kleurrijke "steentjes" in zeer veel verschijningsvormen die je in staat stellen snel iets werkends te maken. Sommige van die steentjes worden door Microsoft ter beschikking gesteld als onderdeel van het Azure platform, sommige door derden (denk aan bijvoorbeeld de API's die door Google, Salesforce, LinkedIn, etc. worden ontsloten) en sommigen ontwikkel je zelf.

Turing test

De vraag is, hoeveel van die Legosteentjes gaan we zelf ontwikkelen? Of zit hem daar juist de onderscheidende factor weer? Ik denk van wel. Organisaties zullen zich gaan onderscheiden en net die extra toegevoegde waarde bieden door hun eigen Legosteentjes te gaan ontwikkelen. En daar is de

software ontwikkelaar in z’n element. Pas als de ultieme Turing test succesvol voltooid is, is de software engineer niet meer nodig: Als de software engineer aan de computer vraagt, lever mij een algoritme om mijn meest complexe probleem op te lossen en de computer antwoordt met de meeste efficiënte C#, Java of Python code. Pas dan is hij overbodig geworden.

Tot die tijd geldt Mark Twain’s (licht aangepaste) uitspraak: *The news of the death of the software engineer is highly exaggerated.*

Het einde van de silo applicatie en de roadmap

De afgelopen jaren is er ontiegelijk veel gebeurd op het gebied van IT. En dat is een understatement. De impact van cloud computing is (en wordt) door ons IT'ers nog erger onderschat dan door de eindgebruikers. Ontwikkelaars en implementatie specialisten moeten in rap tempo hun kennis blijven uitbreiden en hun aanpak blijven aanpassen om waarde te kunnen blijven leveren.

Met name op het gebied van PaaS en SaaS gaan de ontwikkelingen snel. Waar voorheen door software leveranciers eens in de 18 of 24 maanden een nieuwe release van één van hun "silo" applicaties werd gedaan, zien we dat er nu bijna continu functionaliteiten bij komen en veranderen; het voor de IT'er en eindgebruiker vertrouwde fenomeen "roadmap" is achterhaald. Er wordt alleen nog gesproken over "big bets" en rode draden.

Daarnaast zien we dat de leveranciers langzamerhand afstappen van dat andere vertrouwde fenomeen, de "silo applicatie". Functionaliteiten worden als dienst(en) ontsloten en zijn steeds meer verweven met de andere functionaliteiten. Waar voorheen veel integratie tussen deze functionaliteiten door middel van ambachtelijk werk door de "system integrator" moest worden uitgevoerd, is veel van deze integratie al automatisch aanwezig en staat deze ook standaard al "aan".

Democatization of IT

Door de snelle uitbreiding van de cloud platforms is de manier van communiceren door de leveranciers ook sterk veranderd. Waar voorheen publiekelijk geheimzinnig werd gedaan over nieuwe innovaties en het partner netwerk van de leverancier redelijk vroegtijdig betrokken werd bij deze ontwikkelingen zodat men er op kon voorbereiden en de klant goed en tijdig kon informeren, zien we nu soms dat de klant nog eerder op de hoogte is van nieuwe features dan de system integrator. Niet dat de klant dan ook gelijk weet wat er precies mee gedaan kan worden en hoe het geïmplementeerd moet worden. Maar toch. Daarnaast heeft de klant zelf een steeds grotere invloed op wat er ontwikkeld gaat worden. Ik zie het nog gaan gebeuren dat de leveranciers een publieke backlog gaan bijhouden, waar de eindgebruikers op kunnen stemmen en waaraan elke dag of week een vast aantal story points wordt gecommitteerd. *Democratization of IT.*

Door deze nieuwe ontwikkelingen zien we met name op het gebied van kennisopbouw, certificering en het beantwoorden van de klantvraag behoorlijke uitdagingen ontstaan. Zelfs voor de keien in onze wereld is het bijna onmogelijk om de snelheid van de ontwikkelingen bij te houden en focus te behouden. Het CBS rapporteerde in 2016 zelfs dat bijna een kwart van de werknemers (met name in de IT) zegt belangrijke nieuwe kennis of vaardigheden te missen. Maar daarnaast zien we ook dat het voor de leveranciers een zeer grote uitdaging is om het trainingsmateriaal en de certificeringen up-to-date te houden. Of überhaupt te produceren. Boeken publiceren heeft geen zin meer, want die zijn al achterhaald als ze bij de drukker worden afgeleverd. Dus hoe moet men dan die kennis delen en certificeren? Er zal een groter beroep op de community van extraverte specialisten worden gedaan die daar graag een bijdrage aan leveren.

Darwin op z'n best

Menig RFI en RFP kan niet meer met goed fatsoen beantwoord worden. Want op welk "product" gaan we de aanbieding baseren en hoe ziet de toekomst van dat product er uit? Wat is de roadmap? Ook aan de

vraagkant zal er dus het één en ander moeten gaan veranderen om het proces aan te passen aan de nieuwe IT wereld. In ieder geval zullen de vragen rond toekomstige ontwikkelingen wat minder direct en specifiek gesteld kunnen worden omdat simpelweg niemand er antwoord op kan geven. En wees als klant op je hoede als er onverhoopt toch een specifiek antwoord wordt gegeven. Want wat is het commitment?

Natuurlijk zijn er ook in SaaS en PaaS nog steeds vakgebieden waar men in kan specialiseren. De BI consultant, de ERP specialist, de CRM kenner, de App ontwikkelaar, de data scientist en de integratie goeroe zullen zich moeten blijven focussen op waar men goed in is. Het feit dat de software gemeengoed wordt wil nog niet zeggen dat het zichzelf automatisch implementeert. Een ERP implementatie betekent alleen niet meer dat je een silo applicatie implementeert, maar de set van ERP patronen implementeert door middel van allerlei verweven en gekoppelde cloud diensten.

Informatie architectuur en governance blijven mede daardoor de belangrijkste onderwerpen in de IT. Branche-specifieke kennis is hier vaak ook van grote toegevoegde waarde. Maar tevens zal elk van deze specialisten zijn grenzen moeten kennen en respecteren en inzien wanneer een andere specialist aangehaakt moet worden. Project teams zullen complexer worden en de bijbehorende planning dus ook. De integratie tussen de specialisten zal soepeler moeten gaan verlopen en daarbij helpt het als er op basis van standaard architecturen en patronen wordt ontworpen en ontwikkeld. En liefst met behulp van een goed op elkaar ingespeeld multi-disciplinair agile team met branchekennis.

Tot slot zullen we ons er van bewust moeten worden dat het blijven bijleren nu nog belangrijker is dan het vroeger was. We zullen dat moeten inbedden in onze dagelijkse werkzaamheden en in onze spaarzame vrije tijd. Niet mee kunnen komen betekent afvallen. Darwin op z’n best. Het delen van die kennis binnen en buiten onze organisaties is ook van het grootste belang en daarbij helpen de nieuwste cloud platforms natuurlijk ook. Maar denken dat we het trucje wel kennen en daar op blijven voortborduren, die tijd is wel voorbij. Blijven investeren in je eigen kennis en kunde en focus houden op je vakgebied is nu belangrijker dan ooit!

Pair architecting

In agile software ontwikkeling is pair programming een bekend fenomeen. De één schrijft de code, terwijl de ander observeert. De voordelen zijn onder andere hogere kwaliteit van de code, betere samenwerking in teams en, niet onbelangrijk, een groter plezier in het werk.

Op het gebied van software architectuur missen we dit eigenlijk. In de praktijk zie ik vaak dat architecten in agile teams behoorlijke eilanden zijn. Zeker in omgevingen waar meerdere Scrum teams werken aan producten die onder een bepaald programma vallen kunnen daardoor op het gebied van architectuur al snel misverstanden ontstaan. Hierdoor kunnen de teams te ver uit elkaar gaan lopen en zijn de onderdelen van de uiteindelijke totaaloplossing niet goed op elkaar afgestemd. Dat kan ook leiden tot moeilijke beheerbare oplossingen en veel refactoring.

Zou *pair architecting* (volgens mij bestaat de term nog niet) hier niet een oplossing kunnen bieden?

Beetje assertief

Er bestaan verschillende vormen van pair programming en dus ook van pair architecting. De twee meest voorkomende zijn:

- Expert-Expert
- Expert-Nieuwkomer

Beide varianten hebben hun voor- en nadelen. In het Expert-Expert geval zal minder vaak tot nieuwe inzichten worden gekomen, omdat de al jaren toegepaste principes niet snel zullen leiden tot vragen of kritiek. Het Expert-Nieuwkomer model levert vaker verrassende inzichten op, omdat de Expert zaken opnieuw helder moet proberen uit te leggen aan de Nieuwkomer en daardoor tot nieuwe inzichten kan komen én de Nieuwkomer zal vaak al jarenlang toegepaste principes ter discussie stellen, als hij of zij een beetje assertief is. Beide varianten leveren in de meeste gevallen een betere en beter onderbouwde architectuur op.

Dure refactoring

Het belangrijkste is echter dat er meerdere mensen bezig zijn met het bedenken, tekenen en documenteren van architectuur en dat men elkaar scherp houdt. Waarbij de één ook meer tactisch bezig is en de ander tegelijkertijd meer strategische inzichten kan bieden. Tegelijkertijd is er borging van kennis bij meerdere personen en kan de rol van het borgen van architectuur in de agile software-ontwikkelteams altijd ingevuld worden, al is het maar parttime. Door regelmatig de rollen om te draaien in de paren (dan tekent en documenteert de één en observeert de ander en dan weer andersom) houdt men het architectenpaar ook fris.

Architecten die in paren samenwerken zullen in de praktijk ook, net als bij programmering, merken dat:

- De kwaliteit van de architectuur beter is en men met meer zekerheid achter de gekozen architectuur staat
- De ontworpen architectuur breder gedragen wordt

Een laatste belangrijke conclusie is dat hoe eerder in een software-ontwikkeltraject fouten worden voorkomen, hoe goedkoper het is om het op te lossen. Architectuurfouten te laat ontdekken kan leiden tot zeer dure refactoring. Wat dat betreft levert pair architecting potentieel nog veel meer voordeel op dan pair programming.

Business Architect is ook een vak

Onder druk van de business ontstaan vaak gedrochten van IT oplossingen. Nu de CMO of Marketing Manager meer IT budget heeft dan de CIO lijkt het einde helemaal zoek. Hoe kunnen we het tij keren?

Er wordt vaak gezegd dat hoe eerder in een softwareontwikkelingstraject een fout wordt vermeden, hoe goedkoper het is om op te lossen. Vaak denkt men hier impliciet aan fouten in informatie- of technische architectuur. In de praktijk blijkt echter dat fouten in business architectuur een nog veel grotere impact kunnen hebben.

Herontwikkelen

Een concreet voorbeeld liep ik laatst tegenaan. Ik moet het wat abstract houden, anders maak ik meer kapot dan me lief is: Vanuit de business werd gekozen om een nieuwe partner aan te haken om een logistiek probleem goedkoper op te lossen. De drijfveer was kostenbesparing. Echter, de bijbehorende IT oplossing paste niet in de standaard architectuur en bleek uiteindelijk zeer moeilijk te realiseren en onderhouden. Het is nu, na een aantal jaren, bij deze organisatie zelfs zo ver dat deze maatwerkcomponent een upgrade naar een nieuwere versie van het platform zwaar bemoeilijkt en daardoor ook de beschikbaarheid en de schaalbaarheid van de oplossing in de weg staat. Ondertussen is er heel veel geld gestoken in het onderzoek naar de problematiek en de uitkomst zal nog veel meer geld gaan kosten: herontwikkelen!

Een informatie- of technisch architect heeft een vakgerichte opleiding gevolgd en houdt zijn kennis op peil door nieuwe trainingen te volgen en bijbehorende certificaten te behalen. Een business architect heeft ooit een economie of bedrijfsadministratie studie gedaan en daar is het waarschijnlijk bij gebleven. En na deze studie is men in rap tempo vergeten om de kosten van het implementeren van een nieuwe oplossing *compleet* te calculeren en deze tegenover de potentiële besparingen te zetten. Debet en credit. Niet alleen in initiële ontwikkelkosten uitgedrukt, maar ook in de *run and maintain* kosten. Daarnaast zien we in de praktijk ook dat de meeste IT'ers wel iets van business en dus commercie begrijpen, maar andersom is dit vaak een stuk dunner gezaaid.

Gewogen beslissing

Kortom, de mogelijke IT consequenties van in de business gemaakte keuzes worden vaak niet of nauwelijks onderzocht of bespreekbaar gemaakt. Het probleem is dat IT architecten vaak als "moeilijke" mensen worden beschouwd die overal beren op de weg zien. Terwijl de IT architect juist nadenkt over het creëren en behouden van een flexibele architectuur die zo goed mogelijk op de toekomst voorbereid blijft. Waar het vaak wel aan schort, is dat de IT architect problemen uitdrukt in termen als "niet onder architectuur" of "geen mooie oplossing". Als vanuit de IT architectuur nu ook meer kwantificeerbare argumenten kunnen worden aangedragen om bepaalde oplossingen niet te kiezen of misschien op een iets andere manier te implementeren zodat het wel onderhoudbaar, beschikbaar en schaalbaar is, kunnen de twee partijen elkaar misschien beter leren begrijpen en dus gezamenlijk tot de beste oplossingen komen.

De verantwoordelijkheid om vanuit business architectuur tot implementeerbare IT architectuur te komen ligt echter wel primair bij de business architect. Als er afwijkingen in de IT architectuur nodig zijn om toch de gewenste oplossing te implementeren moeten de consequenties duidelijk zijn. Als deze consequenties

kwantificeerbaar zijn (en dat zijn ze mijns inziens altijd) kan er door de business een gewogen beslissing worden genomen. Tòch niet kiezen voor van de nieuwe business opportunity, òf in gesprek gaan met de betreffende logistieke partner uit het voorbeeld om tot een compromis te komen waarbij wel aanzienlijke kostenbesparingen kunnen worden behaald en tegelijkertijd een acceptabele IT architectuur gerealiseerd kan worden is natuurlijk het hoogst haalbare.

De Apps tegen de Applicaties

Hoeveel Apps heb jij op je smartphone? En hoeveel op je tablet? En je portal? Zie je de samenhang nog? Is er samenhang? Wie zorgt er voor die samenhang? Wat was er mis met Applicaties?

Applicaties zijn vaak monolitisch. Ze bieden een bepaalde set aan functionaliteiten in een behoorlijk rigide vorm. Meestal draaien ze op een relationele database. Vaak zijn er maar een paar applicaties in gebruik, geleverd door een aantal leveranciers. De Applicatie is gemakkelijk te upgraden. De leverancier zorgt ervoor dat dat goed gaat.

Voor- en nadelen van apps

Apps zijn vaak Microservices. Ze bieden één functie. En zijn daar heel goed in. Elke App heeft zijn eigen opslagstructuur. Vaak zijn er vele Apps in gebruik die van verschillende leveranciers komen. De Apps zijn afzonderlijk gemakkelijk te upgraden. De leverancier zorgt ervoor dat dat goed gaat.

Apps bieden natuurlijk een aantal grote voordelen, zoals:

- Best of breed keuze voor functies in plaats van applicaties is mogelijk
- Specifieke, taakgerichte functies voor jouw rol
- Gemakkelijk en snel in gebruik te nemen en uit te rollen

Iedereen die met Apps werkt zal echter ook de volgende problemen (h)erkennen:

- De relatie tussen Apps is vaak moeilijk te leggen
- Afzonderlijke App upgrades kunnen de totaalbeleving verstoren
- Het is moeilijk om data integriteit te borgen

Niet iedereen kan met een legodoos iets moois en betrouwbaars maken!

Time-to-market

Er zal eigenlijk een fundamentele keuze moeten worden gemaakt: óf je gaat voor (of terug naar) de monolitische Applicatie, óf je gaat mee in de wereld van Apps. In het eerste geval wordt de business- en informatiearchitectuur door de leverancier bepaald en geborgd, in het tweede geval ben je daar zelf verantwoordelijk voor. Als je IT organisatie daar vertrouwd mee is of je hebt er goede hulp bij dan is de weg van Apps op zich prima te behappen. Als je organisatie redelijk onvolwassen is, is het veiliger om voor de monoliet te gaan. En natuurlijk zijn deze monolieten ook in de vorm van SaaS af te nemen.

Tot slot, hybride scenario's zijn vanzelfsprekend een veel voorkomende vorm tegenwoordig; hierbij wordt een mix van (SaaS) Applicaties en Apps ingezet. De Apps worden dan gebruikt voor de kortere time-to-market van nieuwe functionaliteiten en het maken van onderscheidende bedrijfsprocessen. Neemt niet weg dat business-, informatie- en óók servicearchitectuur dan nog steeds zeer belangrijke onderwerpen zijn, want voor je het weet verzandt je in een niet te beheren en upgraden IT landschap en zit je nog steeds muurvast.

API's kijken

Het jaar 2015 was het jaar van de opkomst van de API's. Het huwelijk tussen SOA en Cloud Computing leverde een mooie nieuwe spruit op! Maar net als met echte kinderen, is een beetje opvoeding wel nodig. Apekoppen zijn het vaak. Op zoek naar hun grenzen.

Vindbaar en veilig

Voor de sceptici onder ons is er natuurlijk niet veel veranderd. Een API is gewoon een application programming interface. Nieuwe wijn in oude zakken. Hoe ongelijk hebben ze. De essentie van API's is dat ze vindbaar zijn, goed gedocumenteerd en goed beveiligbaar. En dat je er geld mee kunt verdienen. Maar ja, vertel dat een scepticus.

Maar helaas zijn veruit de meeste API's niet uit zichzelf goed vindbaar, goed gedocumenteerd en goed beveiligd. Een nieuwe fenomeen *API Management* is afgelopen jaar een hot topic geworden. De meeste integratie middleware en aPaaS (application platform as a service) leveranciers zijn in dit gat gesprongen en hebben in rap tempo mooie oplossingen en diensten in de markt gezet.

Goed beheren en monitoren

API Management zorgt er voor dat vaak al bestaande API's ontsloten kunnen worden naar de buitenwereld, waarbij authenticatie, autorisatie, virtualisatie, monitoring en performance goed geregeld zijn. In feite creëert API Management een lichtgewicht proxy, die goed beheerd en gemonitord kan worden. Daarnaast bied het meestal een mooi platform voor ontwikkelaars, die de API's kunnen gebruiken om apps of portalen aan te koppelen.

Deze ontwikkelaars kunnen ook op een slimme manier de API's van verschillende aanbieders combineren in apps en daarmee meerwaarde creëren op basis van bestaande data. Als (bron) API leverancier kun je bepalen of je de API gratis ontsluit, of dat je ook betaalde varianten beschikbaar stelt. Brons, zilver en goud bijvoorbeeld. Dit kan voor de eigenaar van de data natuurlijk ook erg interessant en lucratief zijn.

Kijken, kijken en niet kopen?

Toch vraag ik me wel af waar het dan naar toe gaat op het gebied van software licenties en het huidige dominante model van reclame inkomsten die het internet mogelijk maken. Als iedereen haar functionaliteit gaat ontsluiten via betaalde, vaak anoniem benaderbare API's zal de leverancier van de achterliggende (SaaS) applicaties niet heel blij worden met de weinige *seats* die verkocht worden. En als de data van bestaande SaaS oplossingen, die nu (bijna) gratis kunnen zijn omdat er reclame wordt getoond op de portal of in de app, ineens via API's benaderd kunnen worden en in een reclameloze context worden getoond zullen de reclamemakers ook achter hun oren gaan krabben. De technologie is er klaar voor, maar de zogenaamde *API economy* is op dit moment nog volop in ontwikkeling!

Anarchitectuur

Cloud. Een eufemisme voor het Internet. Internet. Een virtuele samenleving zonder overkoepelend bestuur. Anarchie! Hoe kunnen we er toch voor zorgen dat dat blijft werken?

Voor velen is het misschien een verrassing, voor de ingewijden niet: De fysieke wereld is opgedeeld in landen, met elk hun eigen bestuur en het Internet is één globale virtuele wereld. De Internet goeroes vinden ook dat dat vooral zo moet blijven. Het Internet is zelfsturend.

Bestuurloos?

Toch is er natuurlijk wel een aantal kaders waarbinnen het Internet zich ontwikkelt. Globale afspraken over onder andere domeinarchitectuur waren toch wel nodig. De goede keuzes die daar in het begin zijn gemaakt, bleken erg belangrijk. Wel lopen we af en toe eens vast. Denk hierbij aan de beperkingen van IP nummering. Laten we voor eens en voor altijd nu eens onthouden: als je als ontwikkelaar denkt “nah, dat moet groot genoeg zijn”, vermenigvuldig het dan minimaal met 1000 voortaan. Maar dat terzijde.

Architectuur is uitgevonden om met name in de fundering rekening te houden met toekomstige zaken. Anders storten dingen in. Maar zeker in de IT betekent architectuur “het kunnen omgaan met verandering”. Een goede architectuur kan jarenlang mee. Tot de eventuele volgende paradigmaverschuiving.

Loos bestuur?

Is architectuur dus nodig? Ja. Kan het werken in een anarchistische omgeving als het Internet? Zeker. Mits je er rekening mee houdt dat er altijd rebellen zijn die zich niets aantrekken van je architectuur en je bouwsel gewoon met graffiti bespuiten. Of een dakterras op je platte dak bouwen. Door tegen betaalbare kosten rekening te houden met het ergste kan een architectuur zich prima staande houden. Maar zoals gezegd kan een volgende grote uitvinding in de IT wel het één en ander teweeg brengen. Een bit wat bijvoorbeeld zowel 1 als 0 kan zijn is iets waar we even over na moeten denken voor we het breed uit gaan rollen.

Ik ben van mening dat met de huidige manier van ontwikkelen van gedistribueerde oplossingen we een heel eind op weg zijn om onder een goede architectuur te werken die prima kan landen in de anarchistische cloud. Microservices architectuur is een uitstekende manier om in deze chaos te kunnen gedijen. Microservices architectuur is eigenlijk anarchitectuur! En wat zijn dan de DevOps eigenlijk?

p.s. Titel met dank aan Skunk Anansie.

Het einde van het document als bron

Met de komst van omnichannel klantcommunicatie is één ding pijnlijk duidelijk geworden: het document als bron van data is niet langer houdbaar. Natuurlijk hadden we al veel langer geleden afscheid moeten nemen van dit fenomeen, maar wat betekent het nu eigenlijk?

Digitaliseren van analoge (lees: meestal ergens anders afgedrukte informatie) is leuk, maar het is slechts een hinderlijke stap in de richting waar we echt naar toe willen: digitaal authentiek werken. Het scannen van door een computer geprint document is eigenlijk waanzin; die scanoplossing had een integratieoplossing moeten zijn! Scanners moeten zo spoedig mogelijk naar het museum, vlak naast de faxmachines en de screenscrapers op de afdeling “Integratiedwalingen”.

Digitaal authentiek

Wat betekent nu eigenlijk digitaal authentiek? En waarom helpt het organisaties om efficiënter, sneller en met minder fouten met de klant te communiceren, ja zelfs de klant centraal te plaatsen? Digitaal authentiek betekent dat informatie een digitale ontstaansgeschiedenis heeft en vervolgens gedurende het hele verwerkingsproces digitaal blijft. Dit betekent dat we niet langer documenten ontvangen (op papier geprint door de ander en daarna lokaal ingescand, of “digitale documenten” zoals pdf’s), maar digitale “records” ontvangen; berichten dus. Berichten die niet als documenten, onveranderd worden opgeslagen als extensie op de meer traditionele records, maar juist genormaliseerd en als records opgeslagen worden. Het document zoals we dat kenden, moet als “verschijningsvorm” worden behandeld en niet meer als bron. De berichten zijn het transportmedium. Ook eigenlijk een verschijningsvorm dus. Waarbij we stilletjes hopen dat de papieren verschijningsvorm van berichten langzaam zal uitsterven. Alleen nog even alle 75-plussers een DigiD en Berichtenbox aansmeren ☺

Immutable architecture

De informatie uit deze berichten wordt dus als genormaliseerde data in relationele, niet-relationele en event databases opgeslagen. Deze informatie wordt telkens verrijkt door de processen die plaatsvinden; de klant die een tweet stuurt, een webformulier invult, een app start of een notificatie krijgt om actie te ondernemen. En door middel van Internet of Things zelfs soms ongemerkt informatie deelt die onderdeel vormt van een ketenproces. Al deze informatie bij elkaar vormt het dossier van bijvoorbeeld een klantcase. Door dit op basis van immutable architecture (wijzig geen informatie, maar voeg er telkens aan toe) te doen, ontstaat een uitstekende records database waarbij telkens perfect is terug te herleiden wanneer en hoe de informatie ontstaan is. Auditability en compliance zijn dan veel gemakkelijker te realiseren en borgen. Deze informatie is niet meer in een traditionele silo database op te slaan en met een bijbehorende silo applicatie te onderhouden. Hier zijn andere architecturen en oplossingen voor nodig. Oplossingen die met grote hoeveelheden data overweg kunnen en deze eenvoudig kunnen presenteren en event-driven, omnichannel actie kunnen ondernemen. De technologie is er klaar voor. Nu nog de IT afdeling zo ver zien te krijgen dat er ingezien wordt dat dit de toekomst is en dat de organisatie zich hiervoor klaar gaat stomen. Laten we als IT’ers ervoor gaan zorgen dat het woord document snel een archaïsch woord wordt.

Probleren

Wat begon als een typfout heeft misschien wel het (voor mij in ieder geval) meest waardevolle woord van 2016 opgeleverd. Een mooie kandidaat om toe te voegen aan de Dikke van Dale. Zeker op het gebied van innovatie en dan met name in de IT is probleren namelijk de belangrijkste methodiek.

Met de komst van cloud computing is probleren nog veel gemakkelijker geworden. In plaats van lange trajecten voor het aanschaffen van hardware en software kunnen services nu simpelweg aan, maar ook weer snel uitgezet worden. En door middel van microservices architectuur kan snel geschakeld worden om nieuwe oplossingen te ontwikkelen en uit te testen.

Fail fast, learn rapidly

Dit maakt het mogelijk om zonder al te veel risico te proberen en te experimenteren; fail fast, learn rapidly. Een mooi adagium wat met name door agile software ontwikkelaars omarmd wordt. Je komt er zo snel mogelijk en tegen zo min mogelijk operationele kosten achter of iets werkt of niet. En de cloud en microservices maken dit nog zó veel gemakkelijker. Hoe eerder je weet of iets (niet) kan werken, hoe beter. Probleren dus! En het probleergeld is de kosten van de (in zo beperkt mogelijke mate) gemaakte uren. Maar hierop kan in mindering worden gebracht de waarde van het geleerde. Waarmee het volgende experiment weer wat doelgerichter kan plaatsvinden.

Organizational memory

Soms kan het zelfs al voldoende zijn om door middel van alleen user interface mockups te bepalen of iets zinvol is om mee door te gaan of niet. Op die manier kan, zonder maar een regel code te schrijven heel snel bepaald worden of iets een goede oplossing kan zijn voor een bepaald business probleem. Probleren betekent niet perse dat er code moet worden geschreven of dingen geconfigureerd moeten worden. De belangrijkste doelstelling van agile ontwikkeling van oplossingen is om zeer efficiënt een zo'n hoog mogelijke door de product owner bepaalde business waarde te creëren. Een goed probleerder vergeet daarbij niet om het geleerde te borgen voor de rest van het team en de organisatie. Want het is eeuwig zonde als de volgende persoon iets zou probleren als een andere persoon al eerder had geleerd dat het een fast fail was. Organizational memory maakt probleren nog doeltreffender.

Zelfrijdende auto als Paard van Troje

We zijn er allemaal zo blij mee, die innovaties van de consumententechnologie reuzen als Google, Facebook en Apple. Zelfrijdende auto's, apps die vrienden voor je maken en dingen die jou vinden in plaats van andersom. Je gaat haast denken dat ze het beste met je voor hebben.

Het leuke is dat het ook echt handige toepassingen zijn! Wie wil er niet achterin zijn eigen auto zitten werken of andere fijne dingen doen, terwijl je lekker relaxed op je bestemming aankomt? En hoe handig is het als het niet meer zoveel tijd kost om dat leuke plekje te reserveren?

Jij bent het onderwerp van je dingen

Zonder dat we het echter beseffen, zijn we een steeds kleiner onderdeel aan het worden van onze "device experience", maar we zijn nog wel steeds het belangrijkste onderwerp. Want laten we wel wezen: die zelfrijdende auto is gewoon een IoT device. Een sensor. Een heleboel sensors. Een soort Google Maps autootje, maar dan eentje die data verzamelt over jou en je gedrag en alles om je heen. Een opt-out is hier niet mogelijk. Of je geniet van je zelfrijdende auto en neemt voor lief dat alles over je verzameld wordt en vervolgens gebruikt wordt om de beste deals aan jou te verkopen, of je rijdt gewoon lekker zelf.

Singularity is al bereikt

Anoniem data verzamelen klinkt leuk, maar bestaat dat eigenlijk wel? Met de correlatiemogelijkheden van de huidige big data oplossingen is het een peulenschil om geanonimiseerde data weer te de-anonimiseren. Daar is nauwelijks ontkomen aan. Dat betekent dat als de data op één of andere manier ontsloten wordt door deze technologie reuzen, het heel makkelijk is om er nóg meer informatie uit te halen. En het te herleiden naar personen. Informatie die weer door anderen te gelde kan worden gemaakt. De API Economy vaart er wel bij.

De singularity is dichter bij dan we denken (lees eens Ray Kurzweil's The Singularity is Near). We hebben het eigenlijk al gewoon bereikt: We kunnen niet meer bestaan zonder onze technologie, maar de grote jongens weten ondertussen steeds meer over ons. Meer nog dan we zelf weten. Zo lang ze het te goeder trouw gebruiken is het prima, want we vinden dat eigenlijk wel fijn. Maar als het gehackt kan worden en vervolgens misbruikt wordt door niet zulke fijne organisaties moeten we op gaan passen. Of als het gebruikt wordt door overheidsinstanties die zichzelf dat recht geven. De opt-out knop voor de smart city heeft grote gevolgen… Nowhere to hide.

Disruptie door de gevestigde orde

De digitale ontwikkelingen denderen maar door. Met de komst van de cloud lijkt het wel alsof alles ook ineens sneller gaat. De ondergang van traditionele business modellen is onvermijdelijk. De directies van vele gevestigde bedrijven kijken als konijntjes in de koplampen van de aanstormende concurrenten.

Meer dan de helft van de zakelijke gebruikers van bijvoorbeeld Microsoft's cloud platform Azure is start-up. Deze start-ups zijn in staat om op een slimme manier oplossingen te combineren en dit als dienst aan te bieden en daarmee op grote schaal klanten te winnen èn geld te verdienen. Schaalbaarheid is geen probleem, daar zorgt de cloud voor. Maar waarom lukt het de gevestigde orde niet om hier tegenwicht aan te bieden?

Arrogantie

Vaak is arrogantie het probleem. En die arrogantie ontstaat vanuit (semi) monopolistische praktijken. En als klant pikken we die arrogantie de hele dag door. Ik erger me persoonlijk kapot aan personeel in horecazaken dat het belangrijker vindt om met elkaar te praten dan klaar te staan voor de klant. Of een taxichauffeur die je botweg vertelt dat je beter dat stukje kan gaan lopen. Of een hotel waar je de hoofdprijs voor betaalt maar dat je vervolgens trakteert op luidruchtige airco's, onbehulpzaam personeel en onbetaalbaar ontbijt. Of TV zenders die stompzinnig amusement en de ene herhaling na de andere uitzenden. Vervang deze diensten door een app en mensen die wel hun best doen en het je echt naar je zin willen maken. Diensten die echt lijken te weten wat je wilt. Welkom Thuisbezorgd, Uber, Airbnb en Netflix.

Deze diensten maken allemaal gebruik van grootschalige clouddiensten en putten hun (klant) kennis uit enorme hoeveelheden data. Maar ze passen ook continu hun diensten aan de wensen van de klant aan. De klant staat hier echt centraal en men doet er alles aan om het hen naar de zin te maken. De grote organisaties, ketens en monopolies lijken te blijven teren op oude successen en gaan er vanuit dat er niets zal veranderen. V&D bestaat niet voor niets niet meer. Het gebeurt echt, raad van bestuur! Wake up!

Wendbaarheid

In tegenstelling tot wat je geneigd bent te denken, zouden de gevestigde bedrijven wel degelijk in staat moeten zijn om tegenwicht te bieden aan dit soort start-ups. Maar dan moet er wel het één en ander gebeuren, zowel op het vlak van technologie als cultuur. Budget is er (nog) genoeg. Kennis is er ook in overvloed. Deze organisaties zitten op een enorme berg aan zeer waardevolle digitale informatie. Informatie die de start-ups, zeker in het begin, ontberen. Soms is hierbij privacy in het gedrang. Maar waarom vinden we het wel normaal als Google of Amazon ons vertelt dat we misschien wel geïnteresseerd zijn in een product, maar vinden we het bijvoorbeeld niet normaal dat de caissière bij Albert Heijn ons vertelt dat je de eieren bent vergeten? Disruptie! Ik voorzie een grote toekomst voor digitale transformatie in de traditionele analoge dienstverlening. De echt persoonlijke benadering weer terug, maar dan mogelijk gemaakt door digitale hulpmiddelen. Er zal dan echter wel iets fundamenteels moeten veranderen in de bedrijfsvoering van die traditionele organisaties.

Een eerste belangrijk veranderpunt is de omslag naar een agile organisatie, die in staat is om doelgericht en snel oplossingen te creëren waar klanten op wachten (zonder dat ze zich daar altijd van bewust zijn).

Het is niet nodig om zaken al helemaal doorgedacht te hebben alvorens je ergens aan begint. Probeer het in een lab, probeert het uit op één of twee locaties. Zorg dat je DevOps implementeert en zit bovenop de experimenten en de waardevolle data die ze opleveren. Stuur continu bij. De caissière is in het geval AH onderdeel van je DevOps team.

Systems of Innovation

Een tweede belangrijk veranderpunt is om los te komen van de strakke en stroeve, procesgerichte back-end systemen die elke vorm van wendbaarheid ontberen. Laten we deze systemen vooral (voorlopig nog) doen waar ze goed in zijn: robuuste, niet snel veranderende processen afhandelen. Saaie dingen. Wat er nodig is om die nieuwe agility van de organisatie te kunnen ondersteunen, en ook de benodigde data te kunnen analyseren en ontsluiten is een flexibele laag bovenop deze “systems of record”. Een laag die differentiatie en innovatie mogelijk maakt. Dit is nodig, zolang de systems of record nog niet op microservices architectuur zijn gebaseerd en dus niet wendbaar zijn.

Het implementeren van dit soort wendbare oplossingen kan natuurlijk alleen met behulp van de juiste, cloud gebaseerde platformtechnologie en een goede data analyse en integratie visie. Maar zonder een agile team, waarin zowel medewerkers van uw organisatie als technologie specialisten van uw IT leverancier opereren is het zelfs met de beste technologie van de wereld een grote uitdaging. Een goed op elkaar ingespeeld team is cruciaal. Een visionaire en sterke product owner ook. Een senior management dat dit faciliteert is hierbij onontbeerlijk. Guided self-organization werkt hierbij het beste: Hybride teams die gestuurd worden aan de hand van duidelijke korte en langere termijn doelstellingen en die na elke succesvolle ingebruikname van nieuwe oplossingen weer bezig zijn met de volgende klantgedreven innovaties. Er is veel vrijheid bij *hoe* zaken worden georganiseerd, ontwikkeld en geïmplementeerd. Continuous innovation kan op deze manier zelfs bij de dinosauriërs onder de organisaties succesvol worden toegepast. Zo kan een traditionele bank een populaire fintech worden en een energieleverancier een klimaatadviseur.

Two and a half speed IT

We kennen ondertussen allemaal de publicaties van Gartner, McKinsey en de overige invloedrijke analisten en adviesbureaus over two-speed en bi-modal IT. Zo'n architectuur waarbij je de robuuste, niet snel veranderende processen in je back-end systemen op orde hebt en daar bovenop een wendbare laag creëert om aan de snel veranderende klantvraag te kunnen blijven voldoen.

Deze wendbare laag wordt vaak met een platform ingevuld, en meer en meer is dit een cloud platform. Deze oplossingen zijn meestal grafisch georiënteerd, waarmee modelleren het nieuwe programmeren wordt. Het doel is dan dat door middel van dit platform sneller nieuwe of veranderende klantprocessen kunnen worden geïmplementeerd. Vaak heeft men hierbij als tweede doelstelling dat deze processen bij voorkeur door de business "power users" in elkaar kunnen worden geklikt. Zodat men niet op die langzame IT afdeling hoeft te wachten.

De rol van middleware

In dit soort omgevingen is het cruciaal om de data architectuur en integratiemogelijkheden op orde te hebben. Daar zit vaak het grote probleem. De data architectuur bij elk bedrijf waar ik tot nu toe rond heb gelopen is vaak op z'n zachtst gezegd een rommeltje. Door er master data management (MDM) en operational data stores (ODS) en data quality services (DQS) tegen aan te plakken worden de onderliggende echte problemen vaak gemaskeerd. Een goede oplossing voor data (integriteit) in combinatie met een service oriented architecture is de heilige graal. Wellicht dat blockchain hier wel iets kan gaan betekenen in de niet al te verre toekomst. Maar dat is een ander, ook interessant onderwerp; voor later. Vooralsnog doen we het hier vaak met een enterprise service bus. Maar steeds vaker horen we hier over dat dat een bottleneck vormt, een centraal beheerde molog is en specialisten niet te vinden zijn.

API's to the rescue?

Sinds een paar jaar bevinden we ons echter in het API tijdperk. Alles heeft een API. Dat is super, met name als het gaat om de vele SaaS applicaties die we gebruiken. Die kunnen we dan lekker makkelijk klik-klak-klaar integreren in onze overige processen. De volwassenheid van dit soort API's wil echter nog wel eens te wensen over laten. Puberteit is een beter woord. Eigen, on-premises applicaties worden ook steeds vaker voorzien van API's. En als we nieuwe, op microservices architectuur gebaseerde oplossingen ontwikkelen communiceren deze services met elkaar via API calls. Maar waar zit dan de proces logica? De service compositie logica die je normaal gesproken in de ESB onderbrengt? En gaan deze API's daadwerkelijk rechtstreeks met elkaar communiceren? Synchroon? Waarmee we in feite weer silo applicaties creëren? Of toch maar liever asynchroon via pub/sub oplossingen? Lichtgewicht service bussen? API Management?

De vertragende factor

Feit is dat we middleware nodig zullen blijven hebben. Liefst natuurlijk zo lichtgewicht mogelijk. Het is echt nooit de bedoeling geweest dat ESB's silo's werden. Het feit dat we met zo'n divers landschap aan applicaties en bijbehorende uitwisselingsformaten en –protocollen te maken hebben, heeft gezorgd voor veel te veel logica in onze service bussen. En men heeft vaak BPM met proces orkestratie verward. Liefst routeren we natuurlijk gewoon alleen canonieke berichten via de bus. Maar dat blijkt vaak nog een stapje

te ver, een utopie eigenlijk. En dus zitten we met een centraal stuk middleware dat onderhouden wordt door specialisten. Die schaars zijn. En die onze time-to-market van nieuwe of veranderende klantprocessen ernstig vertragen.

Gaan API's daar verandering in brengen? Ik denk deels. Dit zit hem met name in het feit dat meer en meer applicaties en nieuw te ontwikkelen oplossingen op basis van API's kunnen communiceren. Dat is tenminste een (defacto) standaard. En als we die API's verstandig opzetten (zoals ooit de bedoeling was met SOA, maar dat hebben we met z'n allen een beetje verpest; de 8 principes van service ontwerp zijn een beetje vergeten), zijn ze goed te gebruiken om oplossingen snel mee te kunnen creëren. Zelfs door de business "power users". Neemt niet weg dat aandacht voor goede data architectuur, microservices met ingebouwde business rules, betrouwbare, idempotente API's, goede design- en runtime governance, etc. zeer belangrijk zijn. En daar zijn helaas toch nog steeds specialisten voor nodig.

Door nu in elk agile ontwikkelteam dat zich bezig houdt met het creëren van oplossingen een ontwikkelaar met integratie ervaring op te nemen is dit probleem grotendeels op te lossen. Hij of zij zal ook de linking pin zijn met het centrale integratie team (ICC), waar onder architectuur de juiste API's zullen worden ontwikkeld en ontsloten. Omdat in de integratie middleware grotendeels met standaard patronen wordt gewerkt en er maar één smaak van ontsluiten van services is (namelijk: API's), zijn ook hier de architectuur bouwblokken sneller op te leveren. In het Scaled Agile Framework (SAFe) noemen we dit de Architectural Runway. En die is nodig om oplossingen voor de business te kunnen blijven realiseren. Two and a half speed IT, here we come!

Terug in je hokje?

Met het achter ons laten van silo applicaties en het omarmen van publieke cloud technologie lijkt er ook een trend te ontstaan dat IT'ers steeds meer van meer dingen moeten weten. Vroeger was het simpel: je bent back-end ontwikkelaar, front-end ontwikkelaar of integratie ontwikkelaar. En binnen deze hokjes specialiseerde je vaak weer. Daar red je het niet meer mee!

Als we kijken naar de lijst van bouwblokken die beschikbaar zijn in Microsoft Azure's Platform as a Service is dat best schrikken. Wat een lijst! Het aantal services neemt ook nog steeds toe. De ontwikkelingen zijn met de komst van cloud computing in een enorme stroomversnelling terecht gekomen. Agile en snelle innovatie gaan kennelijk echt hand in hand. Big Data, Internet of Things, Machine Learning, Bot Framework; het zijn allemaal vakgebieden op zich, maar we hebben het wel stuk voor stuk nodig om er moderne oplossingen mee te creëren. Ga er maar aan staan!

Agile oplossingen realiseren

Ook het creëren van oplossingen op de voor ons op een agile manier beschikbaar gestelde innovatieve cloud platforms gebeurt tegenwoordig meestal op een agile manier: Eén of meerdere Scrum teams die aan producten ontwikkelen die telkens verder evolueren aan de hand van real-time business input en een vaak experimentele aanpak. Deze teams zijn multi-disciplinair. Dat betekent dat elk team in staat is om end-to-end een complete oplossing of deelproduct te realiseren. Een product dat als geheel in productie kan. Inclusief database, back-end logica, front-end en alle integraties. En dat men in staat moet zijn om taken van elkaar over te nemen, elkaar te helpen. Het team levert de oplossing. Men is zo min mogelijk afhankelijk van individuele "towers of knowledge". Die schalen namelijk niet. En zijn ook single points of failure.

Domein expertise verschuift

Voor hard-core specialisten is dit een grote uitdaging. In plaats van puur bezig te zijn met een bepaalde technologie, bijvoorbeeld SQL Server, Azure Stream Analytics of .Net ontwikkeling, heeft de ontwikkelaar ook verstand nodig van de oplossing die er gebouwd moet worden én is hands-on kennis van de aanpalende technologieën nodig. Er wordt aan hele stories gewerkt met verschillende mensen en het team moet immers leveren. De gereedschapskist blijkt al snel te klein. Wat een enorme hoeveelheid componenten en technologische doorbraken. Hoe houdt men de ontwikkelingen bij? Pair programming is hierbij een goed middel om de kennis te delen. En natuurlijk de gewone Scrum rituelen. Tegelijkertijd ontwikkelt het team zich tot een kenniscentrum voor een specifiek business domein. Dat is erg waardevol en blijkt in de praktijk een grote driver voor het sneller kunnen leveren van waardevolle oplossingen. Verticaal specialiseren, als team. Ongeacht de te gebruiken technologie.

Terug in je hokje?

Nee. We eisen veel van de ontwikkelaar en ze worden daar ook prima voor betaald. In plaats van specialistische individuen, is teamontwikkeling veel effectiever. Leren van elkaar, elkaars taken kunnen overnemen, bredere kennis opdoen maar tevens verdieping zoeken. Continue blijven leren hoewel het soms lijkt alsof "we're drinking from the firehose" (zoals de Amerikanen dat zo mooi zeggen). We hebben meer generalisten nodig, die voor 80% uit de voeten kunnen met alle componenten. En voor de

overgebleven 20% hebben we dan nog een aantal lone wolves. De towers of knowledge. De echte nerds. Af en toe laten we ze uit hun hokje, en mogen ze hun kunstje doen. ☺

Europa als digitale kolonie van Silicon Valley

Er gebeurt op het gebied van geopolitiek iets onvoorstelbaar groots op dit moment. En het heeft niets met Rusland, het Midden Oosten, China of Noord-Korea te maken. Ook zijn er geen overheden of regeringsleiders bij betrokken. Het voltrekt zich onder onze neus zonder dat we er echt bij stil staan.

Met de komst van het Internet is er in de wereld veel veranderd. De grootste revolutie in de geschiedenis van de mensheid tot nu toe heeft de afgelopen 15 jaar plaatsgevonden. Nieuws van over de hele wereld bereikt ons sneller dan ooit, en we reageren daar ook sneller dan ooit op. Nuance is ver te zoeken. Populisten regeren door middel van Twitter. Wat is de volgende stap: partijprogramma's via Snapchat?

Are we human, or are we dancer?

Maar dat is binnenkort allemaal irrelevant. Landelijke politiek is niet lang meer houdbaar, zeker niet op manier zoals het nu gebeurt. Het Europese gedachtengoed is waarschijnlijk nog iets langer houdbaar. Maar uiteindelijk zal er echt over de grenzen van de landen en continenten heen geregeerd moeten worden. Op dit moment heeft de governance board van Apple een véél grotere macht dan welke landelijke overheid dan ook. Zijn we nog burgers van een land, of van Google? Facebook heeft meer "inwoners" dan wel land ter wereld ook! Is de governance board van deze organisaties democratisch gekozen?

We're IoT devices!

Uiteindelijk draait het deze partijen allemaal om de data die er verzameld wordt en de inkomsten die daarmee gegenereerd kunnen worden. Google is echt niet geïnteresseerd in zelfrijdende auto's. Het zijn niet meer dan IoT devices die persoonlijk data genereren. Wij zijn het verlengstuk van onze devices. Omgedraaide singularity! De machines hebben ons nodig om geld uit te geven. Robots moeten belasting gaan betalen. Facebook is geen park, maar een winkelcentrum! Apple verkoopt een lifestyle die hen héél veel geld oplevert. En we storten ons allemaal als lemmingen in de "afgrond".

Het digitale koloniale tijdperk

Vroeger werden een land onder de voet gelopen en uitgebuit. De "gouden eeuw" is iets waar Rutte trots op is, maar is in feite een periode van brute annexatie van allerlei landen die betere resources en naïevere mensen hadden dan wijzelf. Op dit moment is het Silicon Valley wat de dienst uitmaakt in de wereld. Een piepklein stukje aarde, waar al onze data verzameld wordt en in geld wordt omgezet. Waar ze overigens in "onze landen" niet of nauwelijks belasting over afdragen. We kunnen niet meer zonder, we worden uitgebuit en we vinden het heerlijk. Het verschil tussen de digitale kolonie en de aloude kolonie uit de gouden eeuw is dat onze resources niet uitgeput raken en dat we niet in opstand komen. We vinden het allemaal wel prima zo. Is er nog een politiek leider die hier macht over heeft? Of zijn de belangen zo verstrengeld dat Silicon Valley too big to fail is? Grappig overigens dat Silicon Valley ooit onderdeel van Mexico was, maar dat terzijde.

Het draait allemaal om de business app

Veel van wat wij IT'ers doen vinden we zelf erg belangrijk. Of het nu infrastructuur, back-end development, front-end development, integratie, security, beheer of iets anders is, we zullen ons expertisegebied altijd in het midden van de PowerPoint plaatsen, met de rest er wat kleiner omheen.

Dat is ook logisch. Het is ons middelpunt waarom ons hele leventje draait, en we bekijken alles om ons heen vanuit dat oogpunt. Vaak ontwikkelen we deze dingen ook geïsoleerd van elkaar. Op zich is daar, technisch gezien, niets mis mee; met de "contract first" aanpak zou het goed moeten komen. Maar vaak missen we hierdoor wel de essentie. En de bijbehorende passie. Het wordt een soort fabriekje met een paar slecht op elkaar aangesloten lopende banden waarvan niet duidelijk is waarvoor het dient.

Wat en hoe weten we wel

Hierdoor missen we ook meestal het mooiste resultaat. Het beheren en dus zo goed mogelijk in de lucht houden van de eindoplossing is daarbij in negen van de tien gevallen ook niet goed belegd. Hierdoor kan de gebruikerservaring ernstige deuken oplopen. De verantwoordelijkheid voor de oplossing als geheel en de verschillende onderliggende componenten blijkt dan een complex verhaal. Om het nog maar niet te hebben over ontwikkel-, implementatie- en exploitatiebudgetten en bijbehorende tijdsbesteding. Dit moet beter!

IT'ers zijn techneuten. Wij concentreren ons op wat we moeten implementeren en hoe we dit technisch gezien het beste kunnen doen. Meer vragen we ons ook meestal niet af. Want al druk genoeg. Een waterval aanpak voor het ontwikkelen en implementeren van oplossingen werkt dit ook in de hand. De meeste oplossingen worden op dit moment toch nog op die manier gerealiseerd. Hoe vaak we ook horen dat bedrijven scrummen. Maar scrummen is meer dan sprints vullen.

Maar waarom?

We kennen ondertussen natuurlijk allemaal Simon Sinek en zijn beroemde gouden cirkel. De essentie hiervan is dat je altijd moet beginnen met "The Why". Waarom ontwikkelen we deze oplossing? Waarom geloven we er in? Wat is, behalve dat er geld mee verdiend wordt, het hogere doel?

Dit is de kern van de rol van de product owner. De product owner moet als hij wakker wordt gemaakt direct kunnen vertellen wat "Het Waarom" van zijn oplossing is. Het team moet dit beleven. En elke beslissing die wordt genomen over de prioriteit, functionaliteit, architectuur en gebruikerservaring moet dit als belangrijkste uitgangspunt hebben. Belangrijk hierbij is dat het hele team zich verantwoordelijk voelt voor de oplossing en dat er geen zaken over de schutting gegooid worden. "Lost in translation" is het grootste gevaar met de waterval aanpak. Met een inspiratieloos product als gevolg.

De onvermijdelijke ijsberg

Uiteindelijk draait het allemaal dus om de business app en met name zijn bestaansrecht. En met business app bedoel ik functionaliteit in de breedste zin van het woord, in de vorm van een app, portal, applicatie, API, het maakt niet uit. Maar het draait dus om die app. De app is ook waar budget voor is. De app heeft een product owner. Maar de app kan alleen bestaan bij de gratie van een goede onderliggende architectuur. Data. Integratie. API's. Platform. Het deel onder water, wat niet sexy is en wat ook je

budgetten opvreet en waarop je schip kapot kan lopen; de onvermijdelijke ijsberg. Door de app niet alleen als front-end te zien, maar als een geheel van componenten dat de verschijningsvorm en gebruikersfunctionaliteit mogelijk maakt, en waarbij al deze componenten door hetzelfde scrum team ontwikkeld worden, kan het geheel als de app worden gezien. De app met de sterke product owner, de visionair die ook het bestaansrecht en dus “Het Waarom” kan verdedigen, en dus ook de bijbehorende budgetten en tijd. Inclusief de benodigde onderliggende motor, die zorgt dat de app echt kan vliegen! De app die door een team ontwikkeld wordt. De app die als team gedragen wordt. De app die de meeste waarde toevoegt. De app die continue evolueert. Of zelfs een revolutie ontketent. De app die het verschil maakt!

Digitaal bladgoud

In deze tijd van digitale transformatie en disruptie, gaan veel mensen er van uit dat het iets is wat hippe nieuwe digitale bedrijven doen om gevestigde organisaties snel overbodig te laten worden. Of, dat het iets is wat een gevestigd bedrijf als een "bult" aan het bestaande proces toevoegt, in de vorm van een nieuw digitaal verkoopkanaal.

Veel gevestigde bedrijven kijken nog steeds als de spreekwoordelijke konijntjes in de koplampen wat dat betreft. Sommige huren IT'ers in om een project uit te voeren om zo'n nieuw digitaal kanaal mogelijk te maken. En dat is dan digitaliseren. Alleen zij noemen het dan digitale transformatie. Het is eigenlijk niets meer dan digitaal bladgoud.

Er is wel wat meer nodig

Echt digitale transformatie behelst wel wat meer. Wat is nu eigenlijk de echte toegevoegde waarde van je bedrijf? Of wat zou de echte toegevoegde waarde kunnen zijn? Zou je huidige kernpropositie niet een bijproduct van iets veel waardevollers kunnen zijn? Kun je door het vermarkten van verzamelde data, jouw product zo goedkoop maken dat het zichzelf op een andere manier financiert en tegelijkertijd nog meer waarde creëert? Het draait allemaal om het slim gebruiken van data. En er wordt veel data verzameld in elke organisatie. Door IoT in te zetten wordt er nog véél meer data verzameld. Data die uniek is binnen het bedrijfsproces van je organisatie. Data die gegenereerd wordt door auto's, smartphones, machines, heftrucks, wearables, schepen, deuren en allerlei specialistische sensoren. Data die je kunt minen om tot de juiste inzichten te komen. Hierbij staan tegenwoordig allerlei mooie standaard bouwblokken in de cloud tot je beschikking, in de vorm van PaaS diensten. Want systems of innovation creëer je met PaaS diensten!

Innovatie is geen project

Het innoveren van een business proces kan natuurlijk in de vorm van een project. Maar wat als dat maar een tijdelijke fix geeft? Innoveren is een continu proces. Iets wat je niet als project kan oppakken eigenlijk. Het is ook niet een team. Of een paar teams. Het is een mindset. Een mindset die je wel met een team of meerdere teams kunt invullen. Maar waarbij eigenlijk het hele bedrijf betrokken moet zijn. Want het is niet alleen een technisch feestje. Het moet multi-disciplinair en liefst ook multi-gender. Mensen uit je organisatie die vaak met klanten in gesprek zijn, marketing, business development, maar ook de afdeling klantenservice barsten waarschijnlijk allemaal van de goede ideeën. Als je deze mensen in een aantal scrum teams laat experimenteren ontstaan er snel goed bruikbare oplossingen. Dat is geen fulltime job (want hé, je hebt ook nog je "echte werk"; je moet ook verkopen, klanten tot hun dienst staan en de bedrijfswaarden uitdragen in de vorm van websites en social media uitingen), maar een rol in één of meerdere teams. Maar als je het goed aanpakt, is er dan wel een marketing team dat buiten scrum teams opereert? Wat doen die daar dan nog??

Elk bedrijf is een software bedrijf

Omdat innovatie tegenwoordig veelal te maken heeft met het digitale deel van het bedrijfsproces, is elk bedrijf eigenlijk een software bedrijf. En net als een software bedrijf ben je continu met nieuwe releases bezig. In dit cloud- en mobiele tijdperk verwacht men dat je eigenlijk dagelijks met nieuwe, onderscheidende functionaliteit komt. Dat betekent dat je als organisatie ook innovatie in je DNA moet

krijgen en houden. Hierbij kan een externe partij in eerste instantie helpen, om je organisatie deze change door te laten maken en vervolgens de agile werkwijze te omarmen. En om ideation te leren en dit om te zetten in echte oplossingen. En in eerste instantie zal je organisatie zeker nog goed begeleid moeten worden. Niet alleen op het technische vlak. Maar op een gegeven moment moet je het wel zelf kunnen. Wie weet trek je als organisatie dan wel weer echt, jong IT talent aan wat in dienst komt, omdat je weer een sexy bedrijf hebt! Als je dat hebt bereikt, weet je dat je het goed doet. Dan ben je echt digitaal getransformeerd, en heb je tegelijkertijd een disruptie in de IT dienstverlening tot stand gebracht. Ook wij IT dienstverleners zullen continu verder moeten evolueren!

Een IT’er is ook maar gewoon een uniek mens

In de IT houdt men van in hokjes plaatsen. Je bent ontwikkelaar. Of architect. Of designer. Er wordt ook graag gegeneraliseerd als het om de verschillende soorten werkzaamheden en de beoogde doelgroep gaat; het ene type mens is beter in bepaalde taken dan andere. Tegelijkertijd lopen de gemoederen rond diversiteit mede dankzij een (gewezen) engineer bij Google hoog op.

Door al deze ontwikkelingen zijn vacatures daarom steeds moeilijker te schrijven. En worden ook steeds nietszeggender lijkt wel. Alleen de functietitel is al erg moeilijk om te bedenken. Maar we willen wel graag de beste talenten op gesprek krijgen en uiteindelijk aannemen. Kunnen we als vacature niet gewoon het profiel van een bestaande medewerker gebruiken en erbij zetten: zo ééntje willen we er nog graag drie van?

Van een 7 een 9 maken

Laten we onszelf niet voor de gek houden. Sommige soft-skills zijn wel enigszins aan te leren, maar echt een kei in communicatie worden terwijl je eigenlijk het liefst in een hoekje stil je werk doet is lastig. Hardcore ontwikkelaar zijn, terwijl je eigenlijk een mensen-mens bent is ook niet waarschijnlijk zonder het te forceren. Het is bij mij biologisch bepaald dat ik geen kaart kan lezen (al denkt mijn vrouw nog steeds dat ik het kan leren als ik maar m’n best doe), maar ik ben best goed in het snel scannen van tekst en het daarna begrijpen van de essentie van het verhaal. Laten we dáár dan maar ons voordeel uit halen!

Het is veel makkelijker én effectiever om de goede eigenschappen die je al hebt nóg beter te maken. Van een 7 een 9 maken kost minder inspanning en frustratie, is vaak een natuurlijker proces, en levert betere en meer duurzame resultaten op dan van een 4 een 6 proberen te maken. Het tegelijkertijd iets proberen te verbeteren van skills die echt onvoldoende zijn is natuurlijk altijd mooi meegenomen.

Laten we het persoonlijk houden

Persoonlijke ontwikkelings plannen (POP) zouden ingezet moeten worden voor precies dat: de ontwikkeling van jou als persoon. De focus moet liggen op het individu. En als organisatie in zijn geheel zijn we verantwoordelijk voor het aanbrengen van de juiste balans in de groep van mensen die we aannemen en inzetten. Als we vinden dat er te weinig medewerkers van een bepaald type aanwezig zijn omdat de teambalans niet goed voelt, moeten we dat veranderen. Dat is een continu proces. Tegelijkertijd moet de persoonlijke ontwikkeling gericht zijn op het versterken van de al sterke skills en waar echt nodig verbeteren van de eventuele zachtere eigenschappen. Iemand die van nature niet goed is in het presenteren voor een groep, zal daar waarschijnlijk nooit in gaan uitblinken. Maar het kan zijn dat hij of zij het delen van kennis op die manier wel erg belangrijk vindt en graag wil doen. Door dan de focus op de inhoud te leggen en met bepaalde “trucjes” de presentatie skills te verbeteren kan zo iemand zich toch tot een uitstekend verteller ontpoppen en zal men zijn of haar iets minder gelikte manier van overbrengen voor lief nemen.

Het gaat om de groepsdynamiek

Om nog even op het onderwerp diversiteit terug te komen. Het gaat er niet om of elke functie in principe door elk persoon ongeacht huidskleur, geslacht of wat dan ook uitgevoerd kan worden. Het belangrijkste

is dat er bij het vormen van teams een goed uitgebalanceerd geheel ontstaat. Van verschillende type persoonlijkheden. Of dat nu vrouwen of mannen zijn. Of alfa's of beta's. Of blauwe of rode persoonlijkheden. De groepsdynamiek is het belangrijkst. Alleen dan ontstaan de mooiste oplossingen!

Vertrouwen oogsten na agile zaaien

Het proces van agile software ontwikkeling is geen technische aangelegenheid. Het biedt de, op papier, beste uitgangspunten voor een grote kans van slagen. Maar waar draait het nu eigenlijk echt om?

We kennen allemaal de “droge” voordelen van iteratief ontwikkelen wel ondertussen. In plaats van big bang opleveren en naar alle waarschijnlijk een grote teleurstelling moeten incasseren, kun je nu kleine teleurstellinkjes sneller bijsturen. Dat is natuurlijk negatief verwoord, maar het is wel de technische essentie.

Betrokken stakeholders

Het begint bij een product owner die inhoudelijk betrokken is, maar ook de business goed weet te vertegenwoordigen. Dat zijn eigenschappen die best moeilijk te verenigen zijn. Politieke gevoeligheden, vriendjespolitiek en andere krachtvelden moeten hier overwonnen worden. En tegelijkertijd dondersgoed weten te verwoorden en onderbouwen wat je van het team wilt. Het is essentieel gebleken dat de product owner probeert de stakeholders zo betrokken mogelijk te laten zijn. Dat betekent in de praktijk dat de stakeholders altijd aanwezig zijn bij sprint reviews en planning sessies. En daar direct hun input op een constructieve manier leveren. Het leuke is dat als ze daadwerkelijk bij elkaar in één ruimte zitten elke twee of drie weken, de interactie tussen de stakeholders ook daadwerkelijk op gang komt. Vaak weten deze mensen van elkaar niet eens echt goed waar ze mee bezig zijn, en blijkt dit dus een uitstekend middel om ze op één lijn te krijgen en de krachten te bundelen. Want telkens blijkt weer dat de verschillende meningen eigenlijk helemaal niet zo ver uit elkaar liggen en dat men eigenlijk allemaal precies het zelfde doel voor ogen heeft: een efficiëntere en effectievere bedrijfsvoering.

IJzersterk team

Dit is eigenlijk de essentie van agile werken. Je werkt als team iteratief naar gezamenlijke doelstellingen toe. En dat team is niet alleen het scrum team, maar juìst de combinatie van het scrum team met de stakeholders. Want het zijn de stakeholders die onder regie van de product owner bepalen hoe het meest waardevolle product kan ontstaan. Het leuke is, dat als het team daadwerkelijk zo veel mogelijk fysiek bij elkaar komt bij elke sprint review en planning sessie er een steeds hechter geheel ontstaat. En dat brengt me bij het volgende: een scrum team met daarbij de stakeholders is niet iets wat een kort leven moet leiden. Zo’n team is er juist om steeds beter op elkaar ingespeeld te raken en dus geleidelijk steeds efficiënter (en voorspelbaarder) kan werken. Dit betekent dat er niet voor elk project een nieuw team samengesteld wordt, maar dat de projecten binnen een programma “langs het team gehaald worden”. Een soort van fabriekje dus. Met een lopende band die backlog heet. Alléén op die manier ontstaat er een ijzersterk team dat een heel programma kan faciliteren.

Vertrouwen als voedingsbodem

Doordat er daadwerkelijk elke iteratie meerwaarde geleverd wordt, ontstaat er steeds meer vertrouwen bij de stakeholders. De sprint reviews, die in het begin wat onwennig zijn, worden steeds enthousiaster. Men durft elkaar echt aan te spreken. Zelfs de introverten worden betrokken team leden die de extroverten kunnen overtuigen van bepaalde oplossingsrichtingen. Het scrum team krijgt ook steeds meer zelfvertrouwen, bij zo veel enthousiaste reacties. Wat ooit begon als team om een bepaald probleem op

te lossen, is nu een team dat steeds meer waarde creëert. Door toepassing van de nieuwste cloud snufjes wordt er zelfs hier en daar echt geëxperimenteerd en geïnnoveerd. Waar de stakeholders eerst “bang” waren voor al te vernieuwende functies, zien we dat door het vertrouwen dat ontstaan is ook meer lef ontstaat en men buiten de gebaande paadjes durft te gaan. Natuurlijk zijn er af en toe dips, vallen er hardere woorden en is de moraal soms iets lager. Juist dan is de betrokkenheid van de scrum master erg belangrijk om te zorgen dat de discipline gehandhaafd blijft en dat het team zo goed mogelijk het werk kan blijven doen. Die dips overleven ze dan vrijwel moeiteloos. In de praktijk zien we dat zo’n team door vertrouwen te kweken jaren in dezelfde samenstelling kan blijven bestaan en ook echt meetbaar positief bij blijft dragen aan de groeiende successen van een bedrijfsvoering. Dat is waar agile om draait: het kweken van vertrouwen en daarmee de beste, innovatieve oplossingen blijven realiseren. Om zo je bedrijf op een hoger plan te tillen, en dan weer hoger en weer hoger!

Over kannibalen en blockchains

Blockchain staat op dit moment helemaal bovenin de hype cycle van Gartner. Dus net voordat het het ravijn in dondert en we even "de rust" gaan nemen om tot echte waardevolle oplossingen te komen. Maar die hype is altijd wél nodig om enthousiasme te genereren en te experimenteren!

Want je zou maar de eerste zijn. Hoewel, verdient degene die bijvoorbeeld de Bitcoin blockchain is gestart er eigenlijk wel aan? Heeft hij ergens een minertje draaien die af en toe wat afroomt? Of krijgen we dalijk reclame op blockchains?

Complexe smart contracts

Hoe een blockchain werkt, daar is al genoeg over geschreven. Die fase zijn we ondertussen wel voorbij. De uitdaging is om op zoek te gaan naar echt bruikbare toepassingen. Wat kun je nu eigenlijk qua processen optimaliseren of zelfs geheel vernieuwen als je een gedeeld grootboek inzet? Zou dat dan een publieke-, consortium- of private blockchain moeten zijn? Het lijkt met name van toepassing in processen waar op dit moment een afhankelijkheid van een derde partij aanwezig is, die als intermediair opereert. Fintech heeft natuurlijk een aantal mooie praktijkcases opgeleverd, maar ook bijvoorbeeld bij de overheid en met name in de logistiek is men driftig op zoek naar goede cases.

Op zich is blockchain een gemakkelijk te begrijpen fenomeen. Het wordt al wat abstracter als we het over smart contracts gaan hebben. Maar dat is wel waar blockchain eigenlijk om draait. Het smart contract waar Bitcoin om draait is de eenvoudigste vorm: schrijf ergens wat af en schrijf dat bij bij een ander. Daar komt geen moeilijk gebruik van Oracles (een vertrouwde agent die externe informatie in de blockchain opslaat, bijvoorbeeld valutakoersen per dag) bij kijken. Maar wat als het smart contract een stuk complexer wordt? Door bijvoorbeeld niet alleen tussen twee partijen te gelden, maar meer? En waarbij ook transacties buiten de blockchain uitgevoerd moeten worden? In de logistiek bijvoorbeeld zou een smart contract voor en vrachtbrief erg waardevol kunnen zijn. Transparant voor alle partijen in de keten. Want dat is waar blockchain over gaat.

Kannibaliseren?

Maar dat is niet even 1, 2, 3 op te zetten en gebruiken in de logistieke keten. Zeker niet als je weet dat in de logistiek de marge verdiend wordt door "intransparantie". Men rijdt liever een vrachtwagen leeg terug dan dat de concurrent weet welk tarief er op de heenweg gehanteerd werd. Dus als we écht met blockchain aan de gang willen in bijvoorbeeld de logistiek, zal er eerst een beter verdienmodel bedacht moeten worden. Een verdienmodel dat transparantie als onwrikbaar basis gegeven hanteert. Leuk, maar tijdens dat we dit gaan doen moet wel de schoorsteen blijven roken! Kannibaliseren van je eigen business model is wel het laatste wat je wilt. Hoewel? Als jij het niet doet, is een ander je misschien voor!

Zelfs Uber wordt op dit moment bedreigd door een blockchain alternatief. Want wat is Uber eigenlijk meer dan een intermediair die slim gebruik maakt van een globaal digitaal netwerk? Hebben we daar wel een Uber voor nodig? Kunnen we ons als "taxi chauffeurs 3.0" niet gewoon verenigen in een consortium- of zelfs publieke blockchain?

Lef nodig!

Uiteindelijk draait het allemaal om lef. Digitale transformatie is niet iets wat je graag je concurrent als eerste ziet doen, zodat jij straks zonder business zit. Nog erger is het als de concurrent uit een compleet onverwachte hoek komt. Het is makkelijker om digitaal te transformeren door middel van blockchain technologie als je in het bezit ben van waardevolle “real world” assets. Dus bijvoorbeeld vrachtwagens, schepen, vliegtuigen, warehouses en terminals. Maar je ziet liever niet dat er een Uber voor scheepsvervoer van containers ontstaat, die de assets van jouw logistieke bedrijf alleen pay-per-minute huurt tegen uitgeknepen tarieven. Je wilt juist als bedrijf toegevoegde waarde leveren in liefst de hele keten, end-to-end. En daar ook de regie over voeren. Maar dat is juist de essentie van blockchain: Er is geen centrale regie! Tenminste, niet bij een publieke blockchain.

Met een consortium blockchain kan wel elke partij in de keten als vertrouwde partij deelnemen. Eigenlijk creëer je dan een verregaande samenwerkingsvorm die als geheel eigenaar is van de blockchain. Maar dan moet dát wel eerst geregeld worden voordat je zo’n blockchain kan optuigen. Er is lef nodig om dit te doen, maar het resultaat zal wel zijn dat er een onderling transparante samenwerkingsvorm ontstaat, die als geheel veel krachtiger kan opereren in de markt: 1+1=3. De toegevoegde waarde van elke partij in de keten kan dan prima vertaald worden in een eerlijk tarief en een goed verdienmodel voor de hele keten. En de klant wordt optimaal bediend.

Help, ik ben gedigitaliseerd! Wat nu?

Elke organisatie is op dit moment wel op één of andere manier bezig met het digitaliseren van processen. Dit begint vaak bij de klant, die graag op meerdere manieren eenvoudig bediend wil kunnen worden. Maar wat is nu de volgende stap?

In de pers ging het het afgelopen jaar of over cryptocurrencies of over digitale transformatie. Beide zijn vrij ongrijpbare zaken en er is veel hype. Laten we eens kijken of we door middel van digitaliseren wat stappen kunnen ondernemen om digitaal te kunnen gaan transformeren.

Weg met die documenten

Digitaliseren is op zich al best een uitdaging voor menig bedrijf. We zijn vaak gewend om documenten als centrale vorm van waarheid te zien en dat zijn eigenlijk onhandelbare dingen om via digitale kanalen te delen en mee te interacteren. We moeten zo snel mogelijk van die pdf'jes af! Echt digitaliseren houdt dus ook in dat we in plaats van een document als "record" te gebruiken, echte records gaan gebruiken. In een relationele, maar waarschijnlijk nog beter in een niet-relationele database. Het document is dan eigenlijk een verschijningsvorm, die waarschijnlijk langzaam uitsterft. En als het een contract betreft, zal het hopelijk een smart contract zijn, die via blockchain uitgevoerd wordt. En zo komen we toch weer bij die andere hype, de cryptocurrency uit. Maar daar gaat dit verhaal niet over.

Digitaliseren is pas echt mogelijk dus, als de opslag slim is. Tot die tijd wordt er vaak met facades gewerkt. Lipstick on a pig ook wel genoemd. Je kent ze wel, van die mooie klantportalen waar je vervolgens een pdf'je van kunt downloaden. Redelijk zinloos. Self-service is dan echt een eufemisme voor do-it-your-self geworden, waarbij je als eindgebruiker eigenlijk alleen maar hopeloos tussen allerlei portalen aan het navigeren bent, de service ver te zoeken is en je steeds gefrustreerder raakt. Je komt er gewoon achter dat het steeds minder efficiënt wordt op die manier. En dat is niet de bedoeling van automatisering.

Kernsystemen de cloud in

De technologie is er gewoon. Mits we gebruik maken van de cloud. In sommige branches gaat de adoptie van cloud technologie erg voortvarend, maar ik zie toch nog redelijk veel "late majority" en "laggard" gedrag. Zeker als het gaat om de kernsystemen is men nog niet erg geneigd om deze in de cloud te laten draaien. En dat is juist waar het onderscheidend vermogen zou moeten zitten en je dus juist van al die mooie cloud spulletjes gebruik zou moeten maken. In de meeste cloud transitie modellen wordt als eerste gekeken naar IaaS en SaaS. Bij die eerste verplaats je je bestaande (gevirtualiseerde) systemen naar de cloud en bespaar je wat op operationele kosten. Bij de tweede migreer je je commodity oplossing naar een nog meer commodity SaaS dienst. Met beide varianten ben je alleen met kostenbesparingen bezig. Maar dus niet echt aan het innoveren. Tenminste, niet onderscheidend innoveren. En dus nog zeker niet heel erg aan het digitaal transformeren.

Die kernsystemen moeten dus de cloud in!

Pas dan kun je echt gebruik gaan maken van "cloud power". Met name op het gebied van PaaS gebeurt er veel in de cloud. Tot een jaar geleden was het ondenkbaar dat je als klein of middelgroot bedrijf dingen als machine learning, cognitive services of slimme op AI gebaseerde bots kon gebruiken. Alleen de grote

spelers konden genoeg investeren in hard- en software om daar echt gebruik van te kunnen gaan maken. Nu is het letterlijk voor iedereen bereikbaar. Zonder investering vooraf in hard- of software. Gewoon “aan zetten” in de cloud en betalen naar gebruik.

Ecosystemen omhelzen

Maar al die slimmigheden gebruiken is nog niet genoeg. Natuurlijk kun je het digitaliseren nog veel verder verbeteren nadat je afscheid hebt genomen van het document als de bron. Door toevoeging van echte omni-channel interactiemogelijkheden lijkt volledige digitalisering een feit. Maar, dat is een illusie. Nog steeds lipstick on a pig. Want, kunnen de acties die door klanten via je mooie omni-channel facade geïnitieerd worden aan de achterkant daadwerkelijk straight-through worden uitgevoerd? Waarschijnlijk niet, of in het beste geval maar deels. Misschien alleen het gedeelte waar je als organisatie zelf controle over hebt. Of juist niet, en creëer je aan de achterkant weer een facade richting je ketenpartners.

Daarom is het belangrijk om niet alleen naar je eigen interne systemen te kijken, maar ook de koppeling te leggen met de toeleveranciers, logistieke dienstverleners, overheden, etc. En dat doen we liefst niet op basis van import- en export mogelijkheden en uitwisseling van Excel sheets via FTP servers of email bijlagen. Nee, dat moet op basis van API’s. Volledige geïntegreerd in de systemen in het ecosysteem. Dat heeft wat voeten in aarde, maar daarmee kun je dus wel échte meerwaarde gaan creëren en echt digitaal gaan transformeren. En in de tussentijd kan het digitale proces steeds gebruikersvriendelijker (en dus aangenamer) gemaakt worden met onder andere op AI gebaseerde chatbots en zorgen dat door middel van slimme data de eindgebruiker steeds minder zelf hoeft te beslissen en invullen.

Maar, slim acteren in en gebruik maken van het ecosysteem is het belangrijkste. Want digitale transformatie draait om ecosystemen. Geen Uber, Airbnb of Amazon zonder cloud, social, maps, apps en data in combinatie met fysieke assets. En dat vergt naadloze integratie. En dat maakt data slim.

Serverless maar niet mindless

Het moderniseren van applicaties en daarmee je klanten beter kunnen bedienen is de belangrijkste reden om aan cloud transitie te doen. De goedkoopste vorm is applicaties uitzetten en de cruciale modules migreren naar SaaS. Daarnaast is het opnieuw naar de architectuur kijken en van API's voorzien een belangrijke stap om digitale ecosystemen te kunnen realiseren. Maar waar in het applicatielandschap gehakt wordt vallen ook spaanders. Hoe gaan we hier mee om?

In traditionele IT organisaties is er een sterke onderverdeling in ontwikkeling en beheer. Ook zien we bij elke organisatie wel een toename in het aantal applicaties, maar wordt er zelden afscheid van een applicatie genomen. Het bestaansrecht van sommige medewerkers is direct afhankelijk van een applicatie. De boodschap dat de applicatie vervangen gaat worden door iets moderners of gewoon uitgezet gaat worden zal niet altijd even goed vallen bij de technisch- en functioneel applicatie beheerders. De rol van de medewerkers in de IT organisatie zal mee moeten veranderen met het moderniseren van het applicatielandschap.

Cloud transitie serieus aanpakken

Bij elke organisatie is tegenwoordig wel een beleid dat voorrang geeft aan SaaS en PaaS boven eigen gehoste applicaties of private cloud. Vaak wordt er begonnen met SaaS diensten. Eerst een kleintje (Dropbox) en later steeds grotere en meer strategische (Office 365). Wat eerst begint als shadow-IT, gedreven vanuit de business, krijgt steeds serieuzere vormen. Vervolgens ontstaat er bij de IT afdeling de notie dat een hybride cloud architectuur noodzakelijk wordt. Omdat de huidige omgeving wildgroei kent en moeilijk te beheren is. Cloud transitie moet serieus aangepakt worden. Het ontwikkelen van de architectuur begint als iets wat alleen met techniek te maken heeft. Hierbij komen onderwerpen als infrastructuur, security, identity en te gebruiken cloud services aan bod. En aan welke principes en requirements er voldaan moet worden. Allemaal belangrijke onderwerpen.

Naadloze dienstverlening

Het huidige applicatielandschap is vaak ondergebracht bij een hosting provider. Deze provider beheert de infra en de servers en is verantwoordelijk voor het "technisch" in de lucht houden van het landschap. De hosting provider staat vaak niet te springen om organisaties de cloud in te helpen. Hun verdienmodel is marge op hardware. Daar is ook het eufemisme "private cloud" waarschijnlijk door ontstaan. Maar wanneer de procedure om een nieuwe VM aan te vragen en op te spinnen twee weken duurt, kunnen we niet echt van cloud computing spreken. Deze providers zullen dus mee moeten gaan bewegen met de keiharde eisen van hun klanten. Tegelijkertijd zal de eigen organisatie met de nieuwe realiteit van hybride cloud om moeten leren gaan. We hebben met twee omgevingen te maken die technisch én organisatorisch tot één geheel moeten worden gesmeed om naadloze dienstverlening te kunnen bieden. Daar zul je zelf voor aan de slag moeten.

Niet alleen een technisch feestje

Cloud transitie is dus zeker niet alleen een technisch feestje. Niet alleen de rol van medewerkers verandert, ook de manier waarop de IT organisatie is ingericht. En hoe met het verlenen van (interne) cloud services moet worden omgegaan. Waar sommige rollen zullen vervallen, ontstaan weer nieuwe rollen simpelweg

omdat het ontwikkelen voor en het beheren van een hybride cloud omgeving complex is. En omdat het nieuwe skills en samenwerkingsvormen noodzakelijk maakt. De cloud maakt het met name mogelijk om een sneller time-to-market te realiseren voor moderne oplossingen. Maar zonder bijbehorende, op DevOps gebaseerde en rond functionele domeinen georganiseerde teams zal de cloud op zich niets veranderen aan de snelheid van realiseren van oplossingen.

Het is juist de combinatie van techniek en organisatie die dit mogelijk maakt. De IT organisatie zal getraind moeten worden in deze nieuwe manier van (samen)werken. De functioneel applicatie beheerder zal onderdeel moeten worden van deze teams. Hij of zij is niet verantwoordelijk voor het in de lucht houden van applicaties, maar voor (delen) van een bedrijfsproces. En met de juiste motivatie en begeleiding zal deze rol en verantwoordelijkheid al snel goed opgepakt en omarmd worden! Nieuw rollen zoals cloud solution architect en cloud engineer zullen ontstaan. Er zal een cloud regie team moeten zijn, dat de regie voert over de hybride cloud architectuur en de daarop landende oplossingen. Coaching van de IT organisatie vanuit dit team is ook een belangrijk aspect. En de meestal nog op oude principes werkende hosting provider zal mee moeten bewegen, want alleen een goede regie is niet voldoende. Cloud transitie betekent dan misschien afscheid nemen van servers, maar zeker niet van goed over zowel de technische- als de business architectuur nadenken en hier de regie over voeren!

Zelfsturende teams die wèl werken

Zelfsturing is hèt modewoord van 2018. Maar als er iets moeilijk is om succesvol te implementeren dan is het dat wel. Waar gaat het vaak mis, en wat kun je doen om dat te voorkomen?

We gaan te traag. En we doen de verkeerde dingen. Ook hebben we als individu te weinig impact en voldoening. Allemaal zaken waarvan gedacht wordt dat we het beter kunnen als we organisaties niet meer op Stalinistische manier aansturen. Zelfsturing lijkt het hedendaagse antwoord op deze uitdagingen. Maar dat gaat echt niet vanzelf.

De belangrijkste valkuilen

Een organisatie die de beslissing neemt om met zelfsturende teams te gaan werken doet dit vaak omdat er bottom up over dit fenomeen wordt begonnen. Zelfsturing en agile worden vaak in één adem genoemd. En dat idee komt zelden van boven af. Dat blijkt één van de grootste bronnen van het niet goed of helemaal niet werken van zelfsturende teams.

We zien in de praktijk vaak de volgende issues bij zelfsturende teams:

- **Per team andere werkwijze en tools**: De zelfsturende teams zijn in principe zelf verantwoordelijk voor de keuze van werkwijze en tools. Architectuur keuzes worden vaak niet meer centraal geborgd. Hierdoor ontstaan oplossingen die niet makkelijk overdraagbaar zijn en kennisdeling en herbruikbaarheid zijn daardoor lastig te implementeren. Er is veel verspilling. En dat is niet erg duurzaam;
- **Groepsdynamiek**: Er zijn vele onderzoeken geweest naar het gedrag van mensen in groepen. Keuzes die in een groep door individuen worden gemaakt worden beïnvloed door de meerderheid. Er wordt daarom steeds vaker gezegd *the majority is always wrong*. En dat er geen informatie wordt gedeeld maar *bias*. Vooroordelen dus. Dat zijn niet te onderschatten gevaren;
- **De product owner is niet sterk of heeft geen mandaat**: De product owner en zijn of haar relatie met de stakeholders is cruciaal. De requirements komen hier vandaan. De *what* wordt hier bepaald. De *how* door het team. De *why* door de directie van de organisatie. En dat laatste brengt me bij het volgende punt;
- **Er is geen missie, visie en strategie**: De *why* is vaak niet goed gedefinieerd. Als het team geen doel heeft, zijn de kaders ook niet helder en zal het snel ontsporen. Niet omdat het team niet goed werkt, maar omdat het geen goede stip op de horizon heeft;
- **Het team is schijn-zelfsturend**: Het is voor menig directie of management lastig om de touwtjes te vieren, om niet aan micro- of zelfs nanomanagement te doen. Sturen op output in plaats van inzet is in de praktijk moeilijker dan gedacht.

Wat moet er gebeuren om zelfsturende teams wel succesvol te laten zijn?

Ook zelfsturende teams hebben sturing nodig

Om van bovenaf te beginnen: Het is kansloos om teams aan het werk te zetten die geen goede kaders hebben. Een team kan eigenlijk niet beginnen, zonder dat de missie, visie en strategie van de organisatie bekend is. Het is belangrijk om een afdeling waar binnen teams werkzaam zijn een vertaling van die

corporate missie, visie en strategie te laten maken. Met z'n allen. Dit helpt om de *buy-in* en *purpose* van de teams te vergroten. Als de corporate missie is "Iedere klant moet tevreden en enthousiast zijn over ons" kan de vertaling naar de strategie voor de afdeling Aftercare bijvoorbeeld zijn "Klantvragen worden via elk mogelijk kanaal snel en tot tevredenheid behandeld". Epics, features en user stories die op de backlog komen zouden altijd eerst aan een paar vragen moeten worden onderworpen om te bepalen of ze ondersteunend zijn aan de missie en visie.

Tegelijkertijd moet binnen de directie ook duidelijk worden dat er op KPI's gestuurd moet gaan worden die met output te maken hebben. Heeft het geleverde bijgedragen aan de missie en in welke mate? Niet: Hoeveel uur heeft het team aan elke feature gewerkt en was dat binnen budget. Geef het team een bepaald mandaat, waarbinnen ze zelf beslissingen kunnen nemen. Zolang de output maar goed of boven verwachting is. En beloon hiervoor het team in plaats van de individuen.

Teams werken op basis van backlogs. Backlogs die, zoals boven al genoemd, gevoed worden met epics, features en user stories. De *what*. Maar wat als de *what* niet goed bepaald is? Wat als de product owner de vraag van de stakeholders niet goed begrijpt? Of de stakeholders eigenlijk niet goed weten wat ze eigenlijk nodig hebben? Het zelfsturende team gaat dan aan de slag met verkeerde uitgangspunten. Daar kan geen zelfsturing iets aan verbeteren, tenminste niet op de punten die ècht impact hebben.

The crowd is always right?

Dan is er nog de manier waarop het team samenwerkt maar zeker ook de samenwerking tussen meerdere teams in een afdeling of programma. Om te voorkomen dat zelfsturende teams van de weg raken, is het ook noodzakelijk dat er architectuur kaders zijn. Zowel op business-, informatie- als technisch vlak. Natuurlijk kunnen die architectuurkaders gezamenlijk ontwikkeld worden. Maar dat moet wel teamoverstijgende output leveren. Wat ook vaak goed werkt is als een team nieuwe inzichten op het gebied van architectuur ontwikkelt, dit met de andere teams gedeeld wordt en vervolgens door hen geadopteerd wordt.

Tenslotte nog het fenomeen groepsdynamiek. In de praktijk blijkt bijvoorbeeld keer op keer dat mensen onder sociale druk vaak meegaan met de meerderheid, vaak onbewust geleidt door iemand met de grootste mond (en niet persé de beste ideeën). Men wordt het dan snel eens met elkaar. Op die manier kunnen zelfs de grootste mogelijke potentiële blunders goed aanvoelen om te doen. Bij het maken van beslissingen in teams is het belangrijk dat elk individu afzonderlijk (zonder groepsoverleg of teamdruk) zijn of haar mening geeft. Daarbij is het ook belangrijk dat het een divers team is. Planning poker is een goed voorbeeld om tot juiste inschattingen te komen. Hoe dingen opgelost moeten worden en wat er nog meer op de backlog zou moeten komen zou eigenlijk zo'n zelfde soort proces moeten volgen. Op zo'n manier kom je wèl tot de situatie *the crowd is always right*.

Zelfsturing gaat dus niet vanzelf, maar mits het goed geïmplementeerd en telkens verbeterd wordt kan het de wendbaarheid van een organisatie zeker verbeteren. En tegelijkertijd het plezier en de *purpose* van de teamleden enorm verhogen. En dat komt de organisatie alleen maar ten goede.

Beleid automatiseren

We kennen ze allemaal wel, de strategische adviezen, architecturen, governance documenten en andere goedbedoelde stukken rond IT oplossingen die niet of maar deels toegepast worden of zelfs in een la verdwijnen. En we blijven er geld aan uitgeven en ons afvragen waarom het niet goed werkt. Hoe kunnen we dit wél effectief maken?

In de IT zijn we erg goed in het automatiseren van allerlei processen om het de beheerder of de ontwikkelaar makkelijker te maken, maar het geautomatiseerd controleren en afdwingen van beleid lijkt ons nog steeds niet zo goed te lukken. Waarin zit hem dat nou precies?

Zero touch beleid

In het cloud tijdperk is het het grootste doel om *zero touch* te werk te gaan. Alles wat met testen, uitrollen, beheren en monitoren te maken heeft gebeurt op basis van scripts. Dit kan niet anders in een *continuous integration and deployment* (CI/CD) omgeving waar time-to-market heilig is en cloud ontwikkelingen steeds sneller op ons af komen. De ontwikkelaar en beheerder wordt het op deze manier gemakkelijk gemaakt. Maar of wat er ontwikkeld en in productie wordt genomen voldoet aan de strategie, principes, architectuur en overig IT beleid wordt tot op de dag van vandaag nog grotendeels handmatig en steekproefsgewijs gecontroleerd, met in de ene hand het beleidsdocument en de andere hand met opgestoken wijsvinger.

Het wordt tijd dat ook het toepassen van beleidsregels verregaand geautomatiseerd wordt. Zodat de regels die ooit bedacht zijn zo volledig mogelijk automatisch worden gecontroleerd. Niet alleen steekproefsgewijs of ingebed in een proces, maar real-time en automatisch.

Smart governance

Beleid automatiseren kan niet door het in documenten te blijven beschrijven. Dit moet in uit te voeren code (scripts) wat onder versiebeheer ontwikkeld is gebeuren. Net als dat er *smart contracts* die op blockchains draaien zijn, moet er *smart governance* komen. Dat betekent dat, net als dat beheerders hebben moeten leren ontwikkelen om beheertaken te kunnen automatiseren, dat architecten en beleidsmakers moeten leren om tools en talen te gebruiken waarin ze hun principes en regels in modellen en code kunnen ontwikkelen die vervolgens automatisch kunnen worden toegepast.

Op zich zijn er al wel tools waarmee (deels) in formele regels beleid kan worden beschreven, maar die blijken nog lang niet toereikend genoeg en worden nog te weinig gebruikt. Ze worden met name gebruikt om, daar is-ie weer: te documenteren. En dit gedocumenteerde staat dan weer los van hoe het in de cloud omgeving soms geautomatiseerd wordt toegepast. Bijvoorbeeld: in het beleid staat dat het opschalen van een cloud resource bij meer dan 25% groei per week eerst goedgekeurd moet worden. En vervolgens wordt dit door een beheerder in een policy geschreven die automatisch wordt toegepast door de cloud service. Wat zou het fijn zijn als dat gewoon vanuit de zelfde bron beschreven als toegepast kan worden. Zodat er geen mismatch meer kan zijn, en het *altijd* gebeurt.

Wat we nodig hebben is dus zero touch beleid door middel van smart governance!

Containers zijn gewoon IaaS lieve mensen

Veel organisaties waar ik kom zitten volop in de cloud transitie. Bijna allemaal hebben ze een *SaaS tenzij* beleid. Maar tegelijkertijd wordt er geworsteld met het fenomeen *vendor lock-in*. Hoe zit dat nu precies?

Het hoogst haalbare in cloud computing is SaaS. Heerlijk, geen onderhoud aan applicaties en geen technisch beheer. Maar gewoon applicatie functionaliteit afnemen als water uit de kraan. Maar wel commodity en weinig onderscheidend. Daarna komt als runner up PaaS. Hiermee kun je je onderscheidend vermogen creëren en lekker innoveren. Dingen als kunstmatige intelligentie en slimme bots komen ineens binnen handbereik van kleine en middelgrote organisaties, zonder enorme investeringen vooraf te hoeven doen. IaaS is het laatste redmiddel; dit wordt met name gebruikt als applicaties niet gemakkelijk om te turnen zijn in PaaS of SaaS maar nog wel erg belangrijk zijn. Maar helaas wordt hier vaak wel mee begonnen, omdat het zo lekker gemakkelijk over te zetten is en misschien toch wel wat operationeel voordeel oplevert.

Waarde creëren

Juist met PaaS kun je de meeste toegevoegde waarde creëren dus. Met name op het vlak van data en integratie is er veel winst te behalen. De ontwikkelingen gaan razendsnel, en je kunt er direct gebruik van maken. Geen lange trajecten om infrastructuur op te tuigen. Gewoon gààn. We hebben het hier natuurlijk gewoon over maatwerk.

En vroeger creëerde je maatwerk in een 3- of 4-gl programmeertaal, misschien in combinatie met wat specifieke bibliotheken met herbruikbare functionaliteit. Dat was goed te porteren. Omdat op de meeste operating systems wel een compiler of interpreter beschikbaar was en òf de source code van die libraries beschikbaar was òf de library ook geporteerd was naar de andere omgevingen.

Tegenwoordig is dat een stuk lastiger. Elke cloud leverancier heeft eigen frameworks en PaaS bouwblokken, die totaal niet compatible zijn met de technologie van andere cloud leveranciers. De frameworks zijn enorm geworden. Groter dan je eigen code. Applicaties en apps worden ontwikkeld deels in code (bijvoorbeeld C#, Java of PHP) en deels met deze steeds groter wordende frameworks. PaaS is eigenlijk gewoon een leverancier specifiek framework. En de verhouding eigen code versus framework verschuift steeds meer richting framework.

Stickiness versus time-to-market

Wat de cloud provider zo mooi *stickiness* noemt is toch wel een eufemisme voor wat je als cloud consument gewoon *vendor lock-in* noemt. Met SaaS zit je behoorlijk vast. Even migreren is niet zo makkelijk. En aangezien deze SaaS diensten eigenlijk nooit op open standaarden zijn gebaseerd is het ook niet makkelijk te automatiseren. Incompatibele opslagmethoden en procesdefinities en -engines zijn de boosdoener. Op PaaS gebied geldt eigenlijk precies het zelfde. Standaarden als XML, JSON en BPEL zijn leuk, maar weinig overdraagbaar als er een zware, toch wel leverancier specifieke implementatie op leunt; het PaaS framework. Opnieuw ontwikkelen is dus de enige mogelijkheid als je naar een andere PaaS provider wilt verhuizen. Maar tegelijkertijd is de time-to-market van oplossingen gebouwd op PaaS in combinatie met SaaS toch wel het allerkortst. En dat is ook erg veel waard!

We zien nu dat er vaak gepraat en geschreven wordt over containers. Lekker schaalbaar, afzonderlijk uitrolbaar en ook porteerbaar. Containers kunnen gemakkelijk verhuisd worden van AWS naar Azure bijvoorbeeld. En andersom. Echter, het is natuurlijk gewoon IaaS. Je bent zelf verantwoordelijk voor de stack. Een klein stackje vergeleken met een VM, maar toch. Onderhoud en beheer doe je dus helemaal zelf. Time-to-market met containers is te vergelijken met traditioneel software ontwikkelen met een 3-gl, maar dan zeer porteerbaar en schaalbaar uitrollen.

Het blijft een moeilijke afweging: portabiliteit en dus exit strategie versus gebruiksgemak en time-to-market. Per case dus een afweging maken op basis van uw requirements op dit gebied!

Digitale self-service als productiviteitskiller

Ik kom de laatste tijd om in de self-service acties die ik moet ondernemen om dingen bij mijn interne en externe leveranciers en afnemers voor elkaar te krijgen. En ik ben waarschijnlijk niet de enige. Watskeburt?

Eerder schreef ik het al: Self-service is een eufemisme voor doe-het-zelf. We zijn daarin compleet doorgeslagen. Want wat zie ik in mijn dagelijkse praktijk?

De “mijnxyz” hel

Zowel in m’n eigen bedrijf, als bij mijn opdrachtgevers als bij mijn leveranciers heb ik elke dag te maken met allerlei administratieve rompslomp. Een verlofaanvraag, een aanvraag voor een project, een registratie van een nieuwe opportunity, een beurt voor m’n leaseauto plannen, het downloaden van m’n jaaropgave, het indienen van declaraties, het invullen van de urenregistratie, het rapporteren over de voortgang van een project, etc. Privé gaat het nog verder: het downloaden van rekeningafschriften, het veranderen van m’n voorschot energie, het aanpassen van m’n polis, het downloaden van een bekeuring, het afmelden van een verbouwing, het lezen van een bericht van een overheidsinstantie, etc. etc.

Stuk voor stuk voorbeelden van (hopelijk) goedbedoelde digitalisering, die zich wat mijn ervaring betreft nog in de luiers bevinden. Want met elk formulier dat ingevuld moet worden loopt je tegen de volgende dingen aan:

- Wéér m’n gegevens invullen want niet goed geïntegreerd
- Wéér door 13 schermen navigeren
- Wéér net niet die éne optie die jij nodig hebt in het lijstje

De ergste ervaring die ik had tot nu toe was die van m’n - nu gelukkig vorige, want net nieuwe auto - leasemaatschappij, die me dwong om een terugroepactie te gaan plannen via hun “mijnxyz” omgeving, omdat ik niet rechtstreeks met een dealer mocht afspreken. Het leek me ook handig om het gelijk te combineren met een beurt. Dat formulier op die site, waarvan ik natuurlijk eerst weer het wachtwoord moest vernieuwen omdat ik er al een jaar niet geweest was (goh, waarom?), dwong me vervolgens een lijst van niet merkdealers door te spitten en na veel mailverkeer en nog een keer proberen 2 uur van m’n leven verspild te hebben heb ik de handdoek in de ring gegooid. Ik heb ze letterlijk gemaild “binnenkort krijgt u m’n auto terug en zoekt u het maar uit met die terugroepactie en beurt”. Niks meer gehoord overigens. Ben wel benieuwd of ze nu zelf dat formulier moeten gaan invullen om het alsnog te gaan regelen. Dat zal wel niet, anders zou dat formulier niet zo slecht zijn waarschijnlijk.

Omgekeerde nearshoring?

Ik begin gewoon te merken dat ik met dit soort dingen zoveel tijd kwijt ben op een dag, dat ik serieus minder productief aan het worden ben. Wat heeft m’n opdrachtgever nu liever: 100% tijd van mij om de opdracht tot een goed einde te brengen? Of 80% en de rest van m’n tijd besteden aan het invullen en afhandelen van onzinnige of slecht bruikbare formulieren? Om aan hun proces te voldoen? Zeker in de zakelijke dienstverlening is dit een echte productiviteitskiller, en kost het extra veel geld, gezien de tarieven voor externe inhuur.

Ik heb het idee dat bedrijven en leveranciers hun administratieve processen omgekeerd aan het nearshoren zijn naar hun eigen medewerkers en klanten. Met als gevolg een algehele productiviteitsdaling. Als het centraal planbureau dit gaat uitrekenen gaan we schrikken denk ik!

Daarom pleit ik voor het volgende: Een slimme chatbot, die mij aanhoort en het vervolgens gewoon asynchroon gaat regelen, zonder me verder lastig te vallen. Met de huidige stand van technologie op het gebied van AI, data en integratie lijkt me dat geen probleem. Maar als blijkt dat dat (nog) niet goed (genoeg) te integreren of automatiseren is, graag gewoon weer een persoon daar achter, die het op gaat lossen. En die persoon gaat zorgen dat het digitaliseren verder verbeterd wordt, zodat het volgende keer wel volledig automatisch kan. Laten we hem of haar gewoon “de red tape fixer” noemen. En laten we hem of haar koesteren. Uiteindelijk een stuk goedkoper, leuker en minder frustrerend voor iedereen.

De IT moet niet te goed worden

IT staat nog steeds best wel in de kinderschoenen. Veel van wat er voortgebracht wordt levert bij lange na niet wat er bij het dromen van verwacht werd. Gelukkig maar. Want wat gebeurt er als het te goed wordt?

We zien daar al een aantal voorbeelden van. Blockchain bijvoorbeeld, en kunstmatige intelligentie.

Blockchain te goed?

Ja, dat las u goed. Blockchain. In de blockchain wordt alles zodanig opgeslagen dat het voor iedereen transparant inzichtelijk is en niet weerlegbaar. Dat is tegelijkertijd te grootste tekortkoming van blockchain technologie. Want, als je dat niet wilt zou je een private blockchain kunnen gebruiken. Alleen, wat is de toegevoegde waarde van blockchain technologie dan nog? De grootste reden waarom blockchain initiatieven mislukken is dat de aangesloten organisaties moeite hebben met de transparantie van de blockchain. Veel bedrijven zitten daar helemaal niet op te wachten. In de logistieke wereld bijvoorbeeld, leeft men van de *in*transparantie. Dat is waar de marge zit.

Het zelfde probleem zien we met te goede KPI's. Je hebt een lekker werkend dataplatform, de aangesloten systemen zijn betrouwbaar en dus zijn de KPI's ook betrouwbaar en dus goed bruikbaar. Je kunt nu gaan bijsturen en daarmee optimalisaties in processen gaan bewerkstelligen. Lean Six Sigma vaart hier wel bij. Elk overbodig stukje werk wordt eruit geoptimaliseerd, de kwaliteit en voorspelbaarheid wordt (bijna) perfect. En tegelijkertijd worden wij allemaal robotjes. Waar kunnen we onze creativiteit dan nog kwijt?

Hoe voorkomen we dat alles doodslaat?

Daarnaast hebben we ook steeds meer te maken met regulering en wetgeving. Onder andere rond privacy. En de bijbehorende auditing. Als we alles op deze manier transparant maken en optimaliseren, hoe kan dan de economie nog goed werken? Valt er nog wel geld te verdienen, als klanten precies weten hoe de leverancier zijn tijd besteedt en inkoop doet? Hoe de processen er precies uit zien? Wat precies de toegevoegde waarde is waarover men BTW betaalt? Wat krijgen we allemaal nu precies voor dat maandbedrag? Kan dat inzichtelijk gemaakt worden, zodat we kunnen vergelijken met andere leveranciers? Kunnen marges wel objectief onderbouwd worden? Bestaan er straks nog wel patenten of is alles open source? Zodat iedereen overal inzicht in kan hebben?

Kan een economie wel zonder "rommelen"? Ik denk het niet!

Laten vooral ruimte laten voor creativiteit en niet te veel alles doodslaan met perfecte KPI's, traceability, volledige transparantie en audit rapporten. Laat de menselijke factor nog een belangrijk onderdeel zijn van de autonomie van teams. Anders zijn wij straks echt de *biological stupidity* en worden we geleefd door computers. Dan is de klimaatverandering misschien wel onze enige "redding". En waar doen de computers het dan nog voor?

Voorbij de relationele database

De kans is groot dat de meeste transactionele data in uw organisatie nog in relationele databases als SQL Server, Oracle en DB2 worden opgeslagen. Bij het moderniseren van uw applicatielandschap zal dit echter in rap tempo veranderen. Waarom? En wat betekent dat dan?

De problemen van de relationele database

Moderne software oplossingen zijn service georiënteerd en zeer gedistribueerd, waarbij de systems of record nauwelijks nog van een user interface zijn voorzien maar alleen goede, *mediated* API's hebben die de business logica ontsluiten. En maatwerk realiseer je door middel van low code en integratie bovenop deze API's. De compositie van deze services ontsloten door API's vindt dus plaats op een hoger niveau. En dat is waar je onderscheidende vermogen zit. Daar gebruik je kleine apps die precies doen wat jij nodig hebt, volledig geïntegreerd met je applicatielandschap. De achterliggende systems of record zullen ondertussen ook gemoderniseerd gaan worden door de leveranciers, waarbij deze ook wegbewegen van de silo en relationele database gedachte. Mark my words.

Er zijn namelijk drie enorme problemen met de relationele database:

- Relaties worden hardcoded gelegd, om de referentiële integriteit te waarborgen. Daar is geen plaats meer voor in een op microservices en apps gebaseerde software architectuur;
- Transacties worden uitgevoerd in de context van deze ene database. Dat kan niet werken in een gedistribueerd applicatielandschap. *Commit* en *rollback* zijn fenomenen van de jaren 80 die niet meer houdbaar zijn;
- En niet te vergeten, de kosten van relationele databases zijn veel te hoog omdat ze vaak op end user licenties gebaseerd zijn.

In de praktijk zie ik vaak al dat relationele databases misbruikt worden om niet relationele data op te slaan. Gewoon omdat zo'n ding er nu eenmaal al is en beheerd wordt. En omdat het dan makkelijk met de backup mee kan. Dit is een slechte strategie. Daar moet verandering in komen. Nu.

Hoe er op voor te bereiden?

Nu is de tijd om al langzaam afscheid te gaan nemen van de relationele database in je maatwerk oplossingen. Ga voor elke behoefte aan data opslag na wat precies het doel is, en hoe de relatie met andere data gelegd kan worden. Er zijn tegenwoordig talloze opslag methoden, zeker in de cloud. Om er een paar te noemen:

- NoSQL of Document database – hierin kun je bijvoorbeeld JSON objecten opslaan en makkelijk en flexibel in zoeken. Deze databases kennen geen schema;
- Graph database – hier sla je allerlei informatie in op en kun je dynamisch n:m relaties leggen. Ideaal om profielen samen te stellen en data naar je toe te laten komen die voor jou interessant is;
- Data lake – hier kun je allerlei variëteiten en volumes van data in opslaan en vervolgens big data analyses op doen. Dit zal ook vaak machine gegenereerde data zijn, zoals door IoT.

In een (micro)service architectuur is het van belang om *eventual consistency* te regelen. De database regelt dat niet meer voor je, simpelweg omdat dat niet kan over meerdere databases heen. Hoe je je services en API's organiseert wordt steeds belangrijker, inclusief de bijbehorende data architectuur.

Hoe zit het dan met business intelligence en analytics? Die oplossingen zullen in rap tempo moeten meebewegen. Meer en meer data die nodig is om dashboards te vullen of rapporten te sturen zal niet uit relationele databases komen. Traditionele ETL is niet meer toe te passen. Kubussen zijn niet zo makkelijk meer op te bouwen. *Artificial Intelligence* als onderdeel van je analytics oplossingen, zelfs als het tradioneel terugkijken betreft ("wat is er gebeurd?"), is onontkoombaar. Laat staan als je *predictive* ("wat gaat er gebeuren?") en zelfs *prescriptive* ("wat moet ik doen om dit te realiseren?") oplossingen nodig gaat hebben. Over dat soort dingen moet je nu al gaan nadenken en je architectuur er op aanpassen. Stil zitten en wachten tot "de hype overwaait" is geen optie! De relationele database is wat mij betreft al klinisch dood. En de bijbehorende SQL skills kunnen in rap tempo de vuilnisbak in.

Applicatieverlatingsangst

In vrijwel elke organisatie is een grote hoeveelheid applicaties in gebruik. Dit is zo gegroeid in de loop der jaren. Soms wordt er gerationaliseerd. Maar meestal groeit het applicatielandschap alleen maar door. Hoe komt dit? Wat betekent het? En wat kunnen we er aan doen?

Migratie is duur een pijnlijk

Ooit heb ik op een Gartner conferentie het volgende gehoord: “Een ERP vervangen is heel duur. Gemiddeld kost het 2 tot 3 CIO’s.”. Het begint natuurlijk met het selectieproces van de nieuwe applicatie of het nieuwe platform, gevolgd door de selectie van een partner om de migratie te realiseren. En dan begint de echte migratie. De business wil meestal alle data en alle processen migreren. Terwijl het vaak eenvoudiger is om het meeste achter te laten en opnieuw te beginnen. En om je processen eenvoudiger te maken, zodat het aansluit bij de standaard features. Gewoon de data die je niet operationeel nodig hebt in een datawarehouse stoppen en zo veel mogelijk van scratch af aan beginnen dus. Maar zo gaat het in de praktijk dus vaak niet. Men wil alles overhalen. En “onze processen zijn uniek” dus we hebben maatwerk nodig! En omdat ERP’s op geen enkele standaard gebaseerd zijn, is dit dus allemaal ambachtelijk werk. Zowel de configuratie als de migratie. En nog maar niet te spreken over al het maatwerk. Erg duur dus. En het gaat ook nogal eens mis. Vandaar dat er ook regelmatig wat mensen op stuklopen. Zo’n 2 tot 3 CIO’s.

Technical debt stapelt zich op

Maar zolang je wacht met upgraden of migreren stapelt de technical debt zich gewoon verder op. De platforms en frameworks waar de applicaties op draaien worden niet meer ondersteund en zijn niet meer te patchen. Waardoor ze een steeds groter veiligheidsrisico vormen. Er zijn geen goede management rapportages meer te ontsluiten. Ondertussen vormen deze applicaties nog wel een onderdeel van het applicatielandschap, en is integratie ook een steeds grotere uitdaging. Een integratielaag vormt hier een belangrijke schakel. Deze is in staat om allerlei legacy systemen te kunnen blijven ontsluiten. Alleen daardoor ontstaat vaak de welbekende spaghetti. En wat als je de integratielaag zelf moet vervangen? Omdat het niet meer ondersteund wordt door de leverancier, of gewoon niet meer doorontwikkeld wordt? En ondertussen wordt het steeds lastiger om de organisatie en het bestuur ervan te overtuigen dat redesign toch echt noodzakelijk is, omdat de processen dan veel efficiënter kunnen verlopen en dat dure maatwerk dan niet meer nodig is. Tussendoor toch nog maar even een nieuw projectje om die ene integratie te realiseren. En laten we dat toch nog maar even met dat oude platform doen, want dat kennen we zo goed en dan zijn de risico’s niet zo hoog. Technical debt++.

Niet sexy meer

Een upgrade of migratie naar nieuwe technologie begint meestal met het maken van een business case. Want ja, we moeten de investering in de migratie of upgrade natuurlijk wel binnen 3 jaar terugverdienen. Het vervelende is dat je vaak een applicatie moet vervangen omdat het out-of-support is geraakt of omdat om andere redenen er geen toekomst meer is voor de applicatie. Terwijl de applicatie soms nog prima voldoet. Ook al is hij geschreven in Delphi of draait hij op een Wang machine. Je moet eigenlijk verder, ook omdat de mensen die er mee werken zoals de ontwikkelaars en de beheerders binnenkort met pensioen gaan, of gewoon niet meer te krijgen zijn omdat het zulke sterk verouderde technologie is. Niet sexy meer

dus. Maar vervangen zonder dat je er echt directe meerwaarde voor terugkrijgt? Als je niet eens met lagere licentiekosten de migratiekosten kan terugverdienen? Dat is wel een erg zure appel. Natuurlijk krijg je er wel meerwaarde voor terug. Zeker als je kiest om naar een SaaS of PaaS dienst te migreren. Meerwaarde in de vorm van kortere time-to-market, hogere productiviteit, makkelijker en dus minder arbeidsintensief onderhoud, moderne technologie en dus makkelijker te integreren en van de nieuwste features op het gebied van data en analytics te voorzien. En natuurlijk minder tijd en dus geld kwijt aan onderhoud. Van de beruchte 80/20 (onderhoud / innovatie) naar meer 50/50. Maar maak met al die verschillende ingrediënten maar eens een sluitende business case. Dat lijkt te hoog gegrepen voor de meeste organisaties. Dan zijn we ineens niet meer zo data gedreven maar spreekt de onderbuik. En wordt het een probleem van de volgende CIO. Dus, of we hebben meer lef nodig (en gaan af en toe op ons bek), of betere tools om business cases te maken die alle factoren meewegen. Ik pleit voor het laatste.

Datagedrevenheid

Het nieuwste buzzword op het gebied van Data & AI is “datagedreven”. Een organisatie moet datagedreven worden. Om concurrerender te worden en een digitale transformatie te kunnen doorlopen en daardoor nog onderscheidender te worden. Wat betekent datagedreven zijn nu eigenlijk? En wat komt er bij kijken?

Ware data

Data verzamelen en centraal beschikbaar stellen via een modern datawarehouse klinkt relatief eenvoudig. Je sluit de bronnen aan, transformeert ze naar een eenduidig model en ontsluit dit naar dashboards. Dit kan tegenwoordig data zijn van allerlei variëteiten en volumes, dus ook niet-relationele. De cloud services om deze transformaties uit te voeren (ELT) zijn sinds de laatste paar jaar behoorlijk geavanceerd. Het probleem zit echter vaak in dingen als eenduidige data definities en datakwaliteit. Eenduidige data definities realiseren is een uitdaging die met name in een complex applicatielandschap ontstaat. Bijna bij elke organisatie dus. Landschappen waar bijvoorbeeld meerdere ERP systemen draaien. Omdat er aan fusies en overnames is gedaan. Wat betekent dan bijvoorbeeld “bruto prijs” in zo’n geval? Is die definitie overal het zelfde? En zo niet (meestal dus), hoe krijgen we dat dan getransformeerd naar wel een eenduidige definitie in het datawarehouse? Zodat daarover gerapporteerd kan worden? Datakwaliteit is weer een ander issue. Dit ontstaat vaak door slechte (maatwerk) applicaties. Meestal door invoerschermen die wat “relaxed” zijn met invoercontroles. Of door het veel voorkomende “misbruik” van velden in een database. Waardoor het ontstaat dat in de ene implementatie van de applicatie veld x betekenis y heeft en in de andere betekenis z. Hoe ga je daar nu eenduidig over kunnen rapporteren? En analyseren? Bij big data is het probleem nog wat groter. Want je kunt nog zo veel data verzamelen; als de kwaliteit niet goed is, kun je er ook niets fatsoenlijks mee voorspellen. Valt niet mee!

Dashboardjes kieken?

Veel datawarehouses, moderne of niet, worden gebruikt om rapportjes en dashboardjes te voeden. Natuurlijk is dat een belangrijke functie; menig manager stuurt zijn afdeling of bedrijf op basis van deze informatie, door in een wekelijkse of maandelijkse meeting op basis van deze informatie akties uit te zetten. Vaak heeft zo’n manager ook nog wel een goed onderbuik gevoel en ziet dat er iets niet goed is in zo’n rapport of dashboard. Omdat het niet goed aanvoelt. Er kan dan altijd handmatig bijgestuurd worden. Maar wat gebeurt er als data niet alleen in een dashboard of rapport eindigt, maar ook gebruikt wordt om automatisch actie te ondernemen? Dus bijvoorbeeld in predictive scenario’s. Of nog geavanceerder: In prescriptive scenario’s, waarbij de manager eigenlijk door de data verteld wordt wat hij moet doen om een bepaalde doelstelling te behalen. De manager zal de onderliggende logica vaak al niet meer doorgronden en zal dus volledig moeten vertrouwen op de data en de algoritmes. Dat zal in het begin even wennen zijn!

Echt sturen op data

In dit soort geavanceerde scenario’s zijn goede data definities en datakwaliteit dus cruciaal en zelfs gevaarlijk als het niet op orde is. Dat betekent dat governance op het dataplatform de nodige aandacht vereist. Het dataplatform, bestaande uit technologie die overweg kan met zowel gestructureerde als niet

gestructureerde data, en zowel relationele als niet-relationele data, is het centrale punt waar je de governance op orde moet hebben. Zaken als architectuurprincipes, data transformatiepatronen, kwaliteits standaarden, onderhoud op de standaard modellen. Dat is waar governance om draait. Dit is waar ook een onderwerp als masterdata haar plekje moet hebben. En waar je de datacatalogus moet hebben draaien en up-to-date houden. Dit is waar betrouwbare data kan worden gevonden. En waar je op kunt sturen. Net als bij security en privacy vraagstukken, geldt dat datagedreven worden iets is wat niet alleen een technisch feestje is, maar juist ook een organisatorische uitdaging. Waarbij strakke handhaving op het gebied van data compleetheid en datakwaliteit nodig is; iets waarop je door je leidinggevende en je collega's aangesproken kan worden indien nodig. Dat is met name voor de cowboys (en girls) in de organisatie een uitdaging. Maar veel wordt juist gewonnen met het op een correcte manier ontstaan van data. Gelukkig ontstaat steeds meer data door IoT devices en niet door menselijk handelen, maar totdat dat 100% is zullen we hier volop aandacht aan moeten besteden, in de hele organisatie.

Het vergt een bepaalde gedrevenheid in de organisatie om dit voor elkaar te krijgen en te houden. Een datagedrevenheid!

Het einde van monolitische platforms

Met de rappe opkomst van functioneel rijke PaaS diensten geleverd door de zogenaamde *cloud mega vendors* zoals Microsoft is er nogal wat veranderd. In eerste instantie heeft het tot verwarring geleid in de markt, maar we zien nu dat organisaties het fenomeen echt gaan begrijpen en omarmen.

Denken in services

Vooral in het begin was het zowel voor afnemers als leveranciers van cloud diensten een moeilijk verhaal. Met name als het gaat om data- en integratie producten. Vroeger was het makkelijk als je een Microsoft-tenzij beleid hebt: als je een oplossing voor data wilde kreeg je SQL Server en als je een oplossing voor integratie wilde kreeg je BizTalk Server. Lekker makkelijk features vergelijken tussen deze producten van Microsoft en andere leveranciers als Oracle of IBM. Checklistje afwerken, scoren en kiezen.

Tegenwoordig worden dit soort data- en integratie functionaliteiten niet meer door producten geleverd maar door een set aan cloud services. Dit heeft in eerste instantie tot enorme verwarring geleid. Wat vergelijken we nu eigenlijk met wat? Zeker als het ging om vergelijkingen tussen aan de ene kant een set van cloud services en aan de andere kant een monolitisch platform.

Afscheid van platform “producten”

Zelfs voor Microsoft was dit moeilijk. Want aan de ene kant is er een krachtig verhaal omdat Microsoft Azure alle bouwblokken biedt om een volwaardig integratie platform mee te realiseren, maar aan de andere kant moest er geconcurreerd worden met bepaalde *pure-play vendors* die monolitische producten boden. Producten die erg goed zijn in één ding. En waarbij de diverse componenten in dat product 1:1 van elkaar afhankelijk zijn. Ook Microsoft had op het integratie gebied zo’n product: BizTalk Server. Een fantatisch integratie product, maar wel monolitisch en weinig service geörienteerd. De processen in zo’n product zijn vooraf bedacht en “ingebakken”. En weinig flexibel dus.

Toch zagen we dat de concurrentie niet makkelijk was. Op enig moment in 2018 is door Microsoft zelfs besloten om alle services die in Azure nodig zijn om een integratielaag mee te realiseren te bundelen en “Azure Integration Services” te noemen. Dat zelfde zagen we gebeuren met “IoT Suite” en op andere gebieden. Natuurlijk was dit met name om het voor potentiële klanten mogelijk te maken producten met producten te kunnen vergelijken. Persoonlijk vond ik dat een zwaktebod van Microsoft. Het slaat eigenlijk helemaal nergens op. Het is *juist* de kracht dat je een *loosely coupled* set van services biedt waarmee je heel flexibel integratielagen kunt realiseren. En dat je daarnaast het hele Azure platform tot je beschikking hebt dat je ook onderdeel kunt maken van je end-to-end processen, of het nu integratie-, data- of app- of samenwerking geörienteerde (deel)oplossingen zijn.

Hetzelfde geldt voor data oplossingen. Tegenwoordig hebben we relationele data stores, table stores, document stores, blob stores, data lakes, enzovoort. Ook dat is geprobeerd allemaal in SQL Server te stoppen ooit. In Microsoft Azure zijn dit allerlei “losse” services die elk in hun eigen schaalbare containers draaien en die met behulp van o.a. Data Factory aan elkaar geknoopt worden tot een data platform dat voor uw organisatie goed past. En waar je door middel van weer allerlei analyse services zoals Databricks of PowerBI gebruik van kunt maken om er informatie van te maken.

Niet-functionele aspecten

Wat interessant is, is dat Microsoft niet alleen al deze bruikbare features als services heeft ontsloten, maar dat er tegelijkertijd veel aandacht is besteed aan architectuur guidance en aan de niet-functionele aspecten ervan. Waar je dus voorheen een silo applicatie had als BizTalk Server en SQL Server (en alle vergelijkbare producten van andere leveranciers), waarin al de beheer en monitoring tools ingebouwd waren, zie je nu dat er over alle cloud services heen tooling is ontstaan die juist end-to-end beheer en monitoring mogelijk maken. Bijvoorbeeld om door middel van infrastructure-as-code bepaalde workloads uit te rollen en om inzicht te krijgen in de relatie tussen alle services. We hebben dus te maken met *loosely coupled* oplossingen die tegelijkertijd heel erg beheerbaar zijn en goed te monitoren. Dat is echt vooruitgang! Ik voorzie dat de monolitische integratie-, data-, samenwerking- en app platforms een langzame dood gaan sterven. Lang leve PaaS!

Slimme data op een presenteerblaadje

Het gebruik van data staat in elke branche en organisatie tegenwoordig wel hoog op de prioriteitenlijst. Iedereen wil datagedreven gaan werken. Maar in de praktijk betekent dat nog te vaak het ontsluiten van deze data in dashboards en rapporten.

Dashboards en rapporten worden in het algemeen door managers gebruikt om inzichten te verkrijgen in wat er gebeurd is en eventueel waarom het gebeurd is. En langzamerhand waagt men zich voorzichtig aan voorspellingen doen. Maar dat staat in veel gevallen nog in de kinderschoenen.

Decline of the dashboard

De laatste inzichten rond de top 10 trends op het gebied van data & analytics door Gartner laten een aantal opvallende dingen zien. Ik zoom in deze opinie wat verder in op een aantal van deze trends, maar met name op “Decline of the dashboard” omdat deze me al jaren nauw aan het hart gaat. Ik heb zelf een lange historie in systeem- en ketenintegratie, en bij alle oplossingen die we ontwikkeld en geïmplementeerd hebben stond het kunnen monitoren van de integratiestromen altijd hoog op de verlanglijst. Er moest altijd een tool komen om te kunnen monitoren. En altijd was dit in de vorm van een dashboard. Met *fancy visuals.* Beheerders en managers moesten altijd iets “tastbaars” hebben om te zien of de zaken nog goed draaien. Liefst ook op een groot scherm aan de muur bij de afdeling support. Telkens heb ik mij er tegen verzet, maar telkens was het een harde eis.

Maar waarom zou je hele dagen naar een dashboard turen, terwijl het 34 jaar geleden (het prille begin van mijn belevenissen in de IT) ook al mogelijk was om *management by exception* te doen? Alleen als er iets aan de hand is op de hoogte worden gesteld van het feit, waarna je direct actie kunt ondernemen!

Het zelfde hebben we zien gebeuren in data & analytics. Mensen willen dashboards en rapporten. Om naar te kunnen staren. En mee te spelen (of je er achter te verbergen). En om vervolgens in een andere applicatie iets met (een waarschijnlijk afgeleide vorm van) deze data te gaan doen. Waarschijnlijk door het over te typen. Of het aan iemand anders door te geven, zodat hij of zij er iets mee kan gaan doen. Ongelofelijk.

Datagedreven acties

Waar het in data & analytics om gaat, is dat je in de juiste context en op het juiste moment de beschikking hebt over de meest relevante data. Zeker als het over het nemen van operationele en tactische beslissingen gaat. Natuurlijk heb je nog steeds wat aan dashboards om trends in het verleden en in de toekomst te kunnen analyseren, en om *what-if* scenario’s te doorgronden. Want dat is waar je je strategie op aanpast, zodat je minstens je voorgenomen doelstellingen kunt behalen.

Maar voor veruit de meeste mensen die bij organisaties werken gaat het dagelijks slim gebruiken van data over het presenteren ervan in de juiste context. Welke prijs kan ik hier het beste afgeven voor deze bulkvracht? Wat is de beste doelgroep om aan te spreken voor deze huurwoning die leeg komt te staan binnenkort? Welke skills moeten we op gaan werven? Dit soort informatie die je helpt om goeie operationele en tactische beslissingen te nemen moet hapklaar worden aangereikt. Binnen de app waar je op dat moment mee aan het werk bent. Het scherm waar je op dat moment een prijs moet bepalen en

ingeven. De tab in de ERP applicatie waarin je de verhuur van deze specifieke woning beheert. De HR applicatie die je social media uitingen rond vacatures publiceert. Dit noemen we datagedreven momenten. Op basis waarvan je direct, in de juiste context actie kunt ondernemen. Datagedreven acties dus!

Operationaliseren van data

Om dit te kunnen bewerkstelligen is het beheer van je dataplatform een must. De kwaliteit van data en de toegepaste algoritmes moet goed zijn en blijven. Zeker als er steeds meer op basis van *machine learning* informatie – *in app* - op basis van automatische voorspellingen wordt aangeleverd. Waarmee het systeem zelfs kan voorschrijven welke actie je het beste kunt ondernemen (*prescriptive*) om een bepaald doel te bereiken. Het bewaken van de kwaliteit van deze AI modellen, het integreren van de benodigde brondata en het ontsluiten van contextgevoelige informatie via API's richting de eindgebruikersapps is niet eenvoudig. De meeste mensen die verantwoordelijk zijn voor het ontwikkelen van deze modellen hebben daar geen kaas van gegeten. Dat is weer een vak apart, samengevat door de term *MLops*.

Datagedreven actie kan niet zonder MLops. Maar voor een groot gedeelte zeker zonder dashboards en rapporten. De data – als het gaat om *predictive* of *prescriptive* dus "slimme data" - moet betrouwbaar op een presenteerblaadje worden aangereikt, zodat je het direct kunt gebruiken als dè informatie die je op dat moment nodig hebt.

Falen met software is een dure hobby

Er wordt elke dag veel uitgegeven aan het realiseren van software oplossingen. Organisaties zien kansen om zich te onderscheiden en digitaal te innoveren met maatwerk. Deze software wordt tegenwoordig ontwikkeld met low code app-, data- en integratieplatforms geboden door cloud leveranciers. De praktijk leert helaas dat we zo op nog grotere schaal onbruikbare oplossingen produceren.

Waar gaat het mis? Een berucht artikel, **"why software fails"** uit 2005 beschrijft dit in detail. Het is treurig om te zien dat er nog niet veel veranderd is in de afgelopen 15 jaar. In deze opinie belicht ik twee van de belangrijkste redenen voor het falen van software oplossingen:

- Unrealistic or unarticulated project goals – de strategie, aanpak en doelstellingen zijn niet duidelijk geformuleerd
- Badly defined system requirements – de wensen en eisen voor de oplossing zijn niet goed in kaart gebracht

Beide redenen zorgen ervoor dat je pas op een laat moment in het traject er achter komt dat de software niet gaat leveren wat er eigenlijk de bedoeling mee was en kun je waarschijnlijk grotendeels opnieuw beginnen. Een dure grap dus.

Platformstrategie en aanpak

Het inzetten van een low code platform om daarop maatwerk te realiseren is een strategische keuze voor elke organisatie. Maar met het aanschaffen van het platform is maar een klein hobbeltje genomen. Met welk doel gaan we dit platform inzetten? Welke leidende principes hanteren we? Welke bedrijfsdoelen gaan we helpen behalen? Hoe past het in het applicatielandschap en hoe gaan we om met informatiearchitectuur? Hoe gaan we deze verandering binnen de organisatie managen? Hoe meten we het succes?

Bij veel organisaties zien we nog steeds dat de IT afdeling hier een leidende rol speelt, terwijl het allemaal om business oplossingen gaat. De business wenst zich te onderscheiden van de concurrenten en wil digitaal innoveren. Om zo een groter marktaandeel te behalen of de marges te verbeteren. Het innoveren van business processen en daarmee verregaande optimalisatie door te voeren is iets wat niet thuis hoort in de *systems of record*. Dat zijn vaak robuuste, administratieve applicaties die ook eigenlijk *commodity* zijn. Steeds vaker zijn dit SaaS applicaties. Iedereen gebruikt de zelfde.

Waar je het onderscheid kunt maken is in het slim aan elkaar koppelen van deze applicaties en cloud services. Door data te verzamelen en verrijken en nieuwe inzichten te vergaren. Door het inzetten van kunstmatige intelligentie. En op de persona's gerichte maatwerk apps, die precies doen waar de betreffende persoon in haar deel van het bedrijfsproces behoefte aan heeft. En liefst zonder al te veel in- of over te typen. Laat de gecombineerde data het meeste werk voor je doen.

Fouten gemaakt in strategie en architectuur (business-, applicatie- en informatiearchitectuur) kunnen dure gevolgen hebben. Hoe eerder je in het software voorbrengingsproces fouten weet te vermijden, hoe goedkoper het is. Zoals in het bovengenoemde artikel zo mooi is verwoord: als je een breifout hebt

gemaakt en daar pas veel te laat achter komt, kun je bijna helemaal opnieuw aan de trui beginnen. Daarmee gaat veel tijd en geld verloren.

Het belangrijkste onderwerp om tot een goede strategie, aanpak en doelstellingen te komen is om workshops met de juiste stakeholders te organiseren. Begin hierbij altijd met een workshop die bepaalt welke stakeholders er precies nodig zijn en zorg ervoor dat ze genoeg tijd beschikbaar hebben. Als dit niet zorgvuldig gebeurd, is de kans op falen verderop in het proces veel groter.

User Centered Design

Als je het aan de eindgebruikers van software oplossingen vraagt hoe dingen beter kunnen, weten ze het meestal wel. Of, denken ze het te weten. De kunst is om de juiste gebruikers bij elkaar te krijgen en er naast te gaan zitten tijdens het uitvoeren van hun werkzaamheden.

Maar daarnaast is het belangrijk om van proceseigenaren te weten te komen hoe processen nu het beste gedigitaliseerd en geoptimaliseerd kunnen worden. Vaak wordt nog gedacht in oude oplossingen; “in het huidige systeem gebeurt het zo”. Een gebruiker is in principe maar een klein radertje in het geheel, en heeft vaak geen zicht op het grotere doel van zo’n proces en hoe dingen echt verbeterd kunnen worden. Om echt innovatieve functionaliteit toe te voegen. Om die klant echt beter te kunnen gaan bedienen.

Er wordt nog veel te vaak *inside out* gedacht. De echte *why* (de vraag achter de vraag) is moeilijk te achterhalen. Dat vergt creativiteit van de UX designer. Het draait tegenwoordig om *customer-centric UX* en niet meer om *organization-centric UX* (zie ook “How UX is transforming business” in Forbes). En het vertalen naar goede requirements is echt wel een vak, waar dit soort mensen goed in getraind is en veel ervaring mee heeft. Het zijn vaak niet eens “techneuten”. Door eerst de gebruikersinteractie en procesflows uit te tekenen en eventueel in een prototype te gieten kan er al heel snel met eindgebruikers en proceseigenaren afgestemd worden of het aan de eisen en wensen voldoet. En of het inderdaad een verbeterd proces oplevert.

Eigenlijk begint daar het *change* proces al; iedereen moet worden meegenomen in dit optimalisatieproces. Met de ontwikkelaars in het DevOps team wordt vervolgens besproken of het technisch haalbaar is. Voordat er ook nog maar een regel code geschreven is. Dit kan allemaal prima gecombineerd worden met een agile aanpak, waarbij er met elke sprint weer waarde wordt toegevoegd. Op deze manier is *fail fast* mogelijk, en kan het in de volgende sprint bijgestuurd worden. Zo wordt het realiseren van innovatieve software oplossingen goedkoper en krijg je echt precies wat bijdraagt aan de in de strategiefase bepaalde doelstellingen. Doelstellingen die uiteraard ook *customer-centric* moeten zijn!

Teveel ballen in de lucht houden

Een transitie naar de cloud is geen “walk in the park”. Het betekent nieuwe competenties aanleren, een nieuwe manier van service delivery richting de interne business en tijdens de migratie, die vaak langere tijd in beslag neemt, ook de huidige omgeving up-and-running houden. Wat moet je doen om dit in goede banen te leiden?

Bijna elke organisatie waar ik strategisch betrokken ben (geweest) bij dit soort trajecten heeft de ambitie om zelf de benodigde cloudkennis op te bouwen en om zoveel mogelijk van het cloud beheer zelf te kunnen doen. Men wil zelf de regie voeren en ook een groot gedeelte van de migratie, en het uiteindelijke beheer zelf uitvoeren. Tegelijkertijd ontbreekt het vaak aan tijd bij de huidige interne IT specialisten. En niet te vergeten bij de diverse product owners van de verschillende domeinen.

Regie voeren

Allereerst moet de organisatie in staat worden om zelf de regie te kunnen gaan voeren over het programma en de bijbehorende projecten. De strategie en doelstellingen van de cloud transitie moeten eerst helder worden. Door middel van strategie en business value workshops uitgevoerd met de belangrijkste (business en IT) stakeholders wordt eerst duidelijk gemaakt waarom we de cloud transitie ingaan, en welke doelstellingen we willen gaan bereiken. Dit kunnen zowel kwantitatieve als kwalitatieve doelstellingen zijn. Het helpt om dit krachtig en beeldend in een infographic te zetten en dit met het hele team dat betrokken is te delen. Hang het ook als posters aan de muur op kantoor! Dan is voor iedereen helder wat de uitgangspunten zijn. De guiding principles voor de product owners zijn dan duidelijk en dat is belangrijk voor de aanpak en prioritering.

Vervolgens moet de organisatie die verantwoordelijk is voor de transitie en de uiteindelijke doelomgeving in de cloud vormgegeven worden. Wie heeft wat in zijn of haar portefeuille en wat zijn de verantwoordelijkheden en taken?

Kennis opbouwen

Natuurlijk is het belangrijk dat externe kennis wordt ingehuurd om zo’n traject te begeleiden. Immers, die hebben het vaker gedaan en kennen de best practices, valkuilen en succesfactoren. Maar de eigen organisatie moet ook de kennis gaan opbouwen op het gebied van cloud services en delivery van die services richting de business. En tegelijkertijd zijn de eigen IT medewerkers cruciaal bij de overdracht van alle organisatiekennis op het gebied van IT richting de externe partij. Zij weten het beste hoe de huidige omgeving in elkaar zit en waarom bepaalde keuzes destijds zijn gemaakt. Veel van deze keuzes zijn ook in de cloud nog steeds van toepassing.

Tijd vrijmaken

Eén belangrijk ding dat echter vaak vergeten wordt, is dat het team dit alles naast de “gewone” werkzaamheden moet oppakken. Dus: de huidige omgeving draaiende houden, zelf kennis opbouwen én interne kennis overdragen naar de externe partij. En daarnaast ook nog eens de nieuw opgeleverde services in de cloud draaiende houden. Dat kost zo 1,5 FTE per FTE. En dat gaat dus niet. Daar loopt het vaak spaak. Zeker ook omdat dat betekent dat de externe leverancier, die voor een groot gedeelte

afhankelijk is van de input van en vloeiende samenwerking met uw IT medewerkers, hierdoor vertraagd wordt en moeilijkheden krijgt met de resourcing en de deadline van het project.

Een aantal dingen kan gedaan worden op dit zo veel mogelijk te voorkomen:

1. Zorg dat er een externe leverancier is aangehaakt die ook (als is het maar tijdelijk of gedeeltelijk) de nieuw opgeleverde services technisch kan beheren in de vorm van een managed services contract. Hier hoeft dan voorlopig even geen eigen aandacht aan te worden besteed;
2. Minimaliseer de werkzaamheden aan de huidige IT infrastructuur. Doe alleen het hoognodige nog daaraan, met name op het gebied van security. Hoe langer het transitie traject duurt, hoe groter de kans dat er grotere dingen in de huidige infra opgepakt moeten worden die weer vertragend werken op het cloud transitie traject;
3. Zorg dat uw IT medewerkers kunnen focussen en niet te veel tijd kwijt zijn aan “randzaken”. Blok vaste dagen en uren voor dit cloud transitie traject. Een goede product owner en scrum master of project manager kunnen hierbij ondersteunen en faciliteren;
4. Haak niet élke IT medewerker aan in élke vergadering of workshop. Dit kost enorm veel tijd en is meestal onnodig. Bepaal zorgvuldig wie aan wat moet deelnemen. Zorg er voor dat er een communicatieplatform aan het begin wordt opgetuigd waarop de belangrijkste beslissingen worden gedeeld met iedereen, zodat iedereen blijft aangehaakt en betrokken.
5. Tot slot: neem signalen van uw eigen IT medewerkers en die van de externen op dit gebeid uiterst serieus. Laat het niet te lang doormodderen en escaleer het tijdig als blijkt dat er niet voldoende tijd kan worden vrijgemaakt voor dit project.

De grootste bottleneck in IT trajecten, en met name cloud transitie trajecten, is de beschikbaarheid van interne IT resources. Als dat risico bij de start van het programma of project al bekend is, is het belangrijk om daar gelijk mitigerende maatregelen voor te nemen. Bovenstaande tips kunnen hierbij helpen.

Kantelen voor procesoptimalisatie

Als we vanuit Motion10 kijken naar de business waarde van IT oplossingen hebben we het vaak over procesoptimalisatie. Efficiënter en effectiever werken in de "bulk" processen, zodat er tijd overblijft om echt aandacht te geven aan de uitzonderingsgevallen. Waarom lukt dit toch in veel gevallen niet? Er wordt te vaak gedacht dat de vervanging van een IT systeem dé oplossing is.

Elke organisatie heeft vele verticale afdelingen die elk een eigen lijnmanager hebben met zijn of haar afdelings KPI's. De medewerkers op de afdelingen acteren in "hun deel" van het primaire of secundaire proces, in de door die afdeling gebruikte systemen. De processen binnen de afdeling zijn meestal wel redelijk op orde en alles lijkt koek en ei.

Afdelingoverstijgend

Maar heel vaak zit het probleem tussen de afdelingen in: In de overdracht, redundantie en het niet echt kennen van de andere rollen in het proces. Of tussen de eigen organisatie en derde partijen. De overdracht van informatie is dan niet optimaal. Er ontstaan wachtrijen: Men wacht op elkaar alvorens verder te kunnen in het eigen deelproces. Terwijl je natuurlijk liefst hebt dat alles "straight through" doorloopt. Daar zijn vaak grote winsten te behalen. We zien regelmatig dat de afdelingen daar zelf geen goed zicht op hebben. Het deelproces werkt immers prima, maar het hele afdelingen-overstijgende proces niet zo.

Dit zit hem vaak in de organisatie. Niet alleen in het feit dat de afdelingen (silo's binnen de organisatie eigenlijk) eigen managers hebben, maar ook omdat er niemand is die verantwoordelijk is voor het hele end-to-end proces. Er zijn wel functioneel applicatiebeheerders, maar die zijn alleen verantwoordelijk voor een stukje van de procesautomatisering. En de manier waarop deze applicaties zijn ingericht en werken is vaak puur ontworpen met het oog op gebruikersvriendelijkheid voor de betreffende applicatiegebruikers. Er wordt dan niet gekeken naar de efficiëntie en effectiviteit door het hele end-to-end proces heen. De schermen die zij gebruiken zijn waarschijnlijk door een *User Centered Design* expert vormgegeven na het meelopen met de gebruikers en het bevragen van deze gebruikers op het gebied van mogelijke optimalisaties. Het werkt dan perfect voor hem of haar, in die specifieke rol, in dat specifieke deel van het proces.

People, process & technology

Vaak ontbreekt het ook aan inzichten in de effectiviteit en efficiëntie van het end-to-end proces. Er is geen of alleen gebrekkige business proces monitoring. Pijnpunten en bottlenecks in het proces komen dan alleen aan het licht als "de klant gaat piepen"; het zogenaamde piep-systeem.

Wat er eigenlijk nodig is, is een proces eigenaar per primair en secundair bedrijfsproces. Die proces eigenaar heeft ook de tools nodig om inzichten te krijgen in de performance (kwalitatief en kwantitatief) van het proces. Sommige bottlenecks kunnen misschien weg-geautomatiseerd worden door een workflow of een app in te zetten. Anderen kunnen weer opgelost worden door betere procesafspraken te hanteren. De meeste kunnen waarschijnlijk opgelost worden door meer datagedreven te gaan werken. Out-of-the-box denken helpt hierbij, vaak geïnspireerd door een externe partij. We hebben het dan eigenlijk over *Process Centered Design*.

Een aantal voorbeelden:

- Door burgers pro-actief op de hoogte te stellen dat hun paspoort bijna verloopt en korting te geven door niet vlak voor de vakanties te komen verlengen kan topdrukte (en het inzetten van extra capaciteit) bij het gemeentehuis voorkomen worden. Dit kan gerealiseerd worden door een workflow te koppelen aan een databron en elke dag een keer te laten lopen om emails te versturen aan de betreffende burgers.
- Door actief gebruik te maken van data op het gebied van het weer kan proactief het call-center van een verzekeraar opgeschaald worden met mensen uit een andere afdeling die minder deadline gedreven is. Deze mensen hebben een basistraining “schade afhandeling” ontvangen en krijgen op de dagen dat het nodig is geautomatiseerd tijdelijk toegang tot de betreffende systemen.
- Door de benodigde acties en controles in het verhuurmutatieproces van een woningcorporatie meer parallel te laten verlopen en de betrokken personen en instanties te voorzien van de juiste data in de juiste context kan de leegstand (en daarmee de periode van huurderving) verkort worden.

Dit soort inzichten en daarop volgende verbeteringen van de processen kunnen alleen tot stand komen als alle rollen in het hele end-to-end proces betrokken zijn. We kantelen de organisatie virtueel, specifiek voor een proces. Onder leiding van de betreffende proces eigenaar. Onder begeleiding van een Motion10 LEAN consultant wordt dan snel helder waar de bottlenecks op welke manier opgelost kunnen worden. En dat kan dan zowel een organisatorische-, proces- of technologische aanpassing betreffen. *People, Process and Technology!*

Frankenstein IT-oplossingen

Bij de vele organisaties waar ik betrokken ben geweest bij IT strategie en architectuur valt me telkens weer op dat men zich regelmatig “ingraaft” in technologie. Wat ooit begon met een mooie architectuur is ontspoord in een Frankenstein oplossing zoals ik het maar gewoon noem.

Vaak komt dit voort uit een combinatie van het gebruik van monolitische (en voor meerdere jaren aangeschafte) software producten, het fenomeen “sunk cost” en architecten die als gevolg daarvan als timmerman alles als een spijker (moeten) beschouwen.

Frankenstein oplossingen

Wat bedoel ik precies met Frankenstein oplossingen? Een aantal voorbeelden:

- Er is maatwerk nodig om het ERP systeem geschikt te maken voor de specifieke organisatieprocessen. Er wordt gekozen om het maatwerk op zodanige manier verweven in het ERP onder te brengen dat elke update van de kern leidt tot problemen. Er is niet gekozen voor loosely coupled maatwerk. Omdat daar in het IT landschap geen mogelijkheid voor was.
- De integratie middleware biedt mogelijkheden om business logica te creëren en data langdurig op te slaan. Dit wordt gebruikt om tekortkomingen op bijvoorbeeld het gebied van datakwaliteit of gemis aan bepaalde benodigde data in de bronsystemen te maskeren. Daarnaast wordt vaak de fout gemaakt dat het middleware team daar ook nog eens verantwoordelijk voor wordt.
- Er is gekozen voor een NoSQL dataplatform en hiermee worden ook oplossingen gecreëerd om relationele databronnen aan te sluiten en data harmonisatie problematiek op te lossen. De oplossingen zijn niet meer te begrijpen en resulteren in bijvoorbeeld autorisatie en privacy problemen zodra een relatie in de bron verandert.

De tools zijn in alle drie de voorbeelden als hamer gebruikt. Maar de oplossing is vaak niet een spijker. Soms is er een schroef nodig, of lijm, of een click-systeem. Maar omdat we nu eenmaal een hamer in handen hebben proberen we het toch maar passend te maken. Terwijl je als kind toch al met de blokkendoos hebt geleerd dat een driehoekig blokje niet in een vierkant gat past. Tenzij je het heel creatief probeert te draaien en toch passend te maken. Of er heel hard met een hamer op slaat.

Hoe komen we hier uit?

Met de komst van PaaS (cloud platform-as-a-service) diensten is het gemakkelijker geworden om de juiste service voor de juiste uitdaging te kiezen. Wat in het begin leek op een legodoos en waar veel afnemers bang waren voor de complexiteit die die grote set van verschillende services met zich meebrengt, blijkt dit toch een zegen te zijn. Zeker met de komst van steeds betere governance en monitoring tooling in de cloud worden de oplossingen die je ontwikkelt op basis van deze vele PaaS diensten steeds beter beheersbaar.

Voor de architect wordt het ook makkelijker. In plaats van keuzes te moeten maken op basis van de al aangeschafte dure licenties van product x of y, kan er eenvoudig gekozen worden voor een nieuwe PaaS

dienst, als de betreffende uitdaging daar om vraagt. Er kan ook eenvoudiger snel gefaald worden, omdat je als het toch niet goed blijkt te werken in een proof-of-concept situatie, je de cloud service gewoon weer uitzet.

Het vereist wel enorm veel kennis bij de architecten. Niet alleen op het gebied van de verschillende beschikbare cloud diensten, maar zeker ook rond de verschillende architectuurpatronen. En de gebruikskosten per dienst. Gelukkig worden de cloud leveranciers hier ook steeds duidelijker in. Elke leverancier heeft wel “open source” best practices en patterns waar je eenvoudig gebruik van kunt maken. En een rekenmachine om de operationele kosten beter in te kunnen schatten.

Tot slot: Het is altijd moeilijk om een eenmaal gekozen richting te veranderen. Zeker als er al veel geld in geïnvesteerd is. Maar denk hierbij vooral ook goed na over de onderhoudskosten. En het werkplezier van degenen die het dagelijks in de lucht moeten houden. Het blijkt in de praktijk maar al te vaak dat 80% van je IT budget naar onderhoud gaat, zodat er maar 20% overblijft voor echte innovatie. En dat IT medewerkers daardoor ontevreden vertrekken. Dus, zorg er als organisatie voor dat je architecten meer gereedschappen in hun gereedschapskist hebben dan alleen een hamer!

Over Gijs

Gijs is een *old professional* in de IT. Tenminste, in dat hokje wordt hij door de maatschappij geplaatst. Sinds een paar jaar 50-plusser en dus komt hij in aanmerking voor een aanleunwoning, gesubsidieerde luiers voor volwassenen en korting op het museum en openbaar vervoer.

Alle gekheid op een stokje, Gijs heeft jarenlange ervaring in de IT. Ooit als junior begonnen als hardcore C systeemprogrammeur in 1986. Veel geleerd bij z'n eerst baantje, door toedoen van een enthousiaste en geniale programmeur en veel op stap gaan met een evenzo geniale business analist. Dat zijn de mentoren die je als jonge hond nodig hebt! In die tijd was hij dus veel met C, Unix, VMS, Linux, Oracle, DB2 bezig. Eigenlijk alles behalve Microsoft. Het mooiste vond hij software schrijven die code genereert of transportprotocollen ontwikkelen. Lekker complex.

In zijn verdere loopbaan is hij lead van een R&D team geweest dat integratie middleware ontwikkelde. Software die door vele grote wereldwijde bedrijven gebruikt is. Zeker in het begin was hij ook zelf nog hands-on betrokken bij zowel front-end als middle tier software ontwikkeling. En natuurlijk als product manager verantwoordelijk voor de specs. Deze software draaide met name op Microsoft's Windows en SQL Server. En was met C, C++ en PowerBuilder ontwikkeld.

Sinds 1995 is hij samen met partners in meerdere bedrijven betrokken geweest als mede-oprichter en technisch geweten. CTO, of Chief Technology Officer heet dat dan zo mooi. Microsoft heeft deze technologie, die sinds het begin jarenlang verder door ontwikkeld werd uiteindelijk in 2007 overgenomen en geïntegreerd in hun integratie middleware. De samenwerking met Microsoft in Redmond gedurende 2000 t/m 2007 was erg leerzaam. Elk jaar zeker 8 keer op en neer vliegen naar Seattle en met al die hyper-slimme Microsofties in Redmond samenwerken. Geweldig!

Van 2008 tot en met 2021 is Gijs met name als CTO verantwoordelijk geweest voor de technische visie & stategie, innovatie, architectuur en kennismanagement bij Motion10, een Microsoft consultancy partner die in 2015 tot Partner of the Year werd verkozen en de laatste jaren telkens Great Place To Work © was. Vanaf 2022 is hij zelfstandig en betrokken bij cloud strategie en architectuur bij diverse organisaties.