Vlucht WS7102

fast forward naar web-scale architectuur

InfoSupport
Solid Innovator

Vlucht

WS7102

fast forward naar web-scale architectuur

Colofon

Dit boek werd geschreven door leden van het Competence Center Architectuur van Info Support. Dit zijn: Raimond Brookman, Ronald Bosma, Wiljag Denekamp, Wim van Gool, Sander Molenkamp, Rogier Schrama en Edwin van Wijk.
Zij zijn allen softwarearchitect bij Info Support.
Freek Jansen van communicatiebureau LEWIS en freelance copywriter Martijn Vet hebben geholpen bij het schrijven van de teksten.

Illustraties zijn van de hand van Wiljag Denekamp en Edwin van Wijk. De opmaak is gedaan door Erika Watt-van de Bovekamp.

Alle personages en gebeurtenissen in dit verhaal zijn fictief. Elke overeenkomst met bestaande of overleden personen en gebeurtenissen berust op louter toeval.

1e druk oktober 2016

Voorwoord

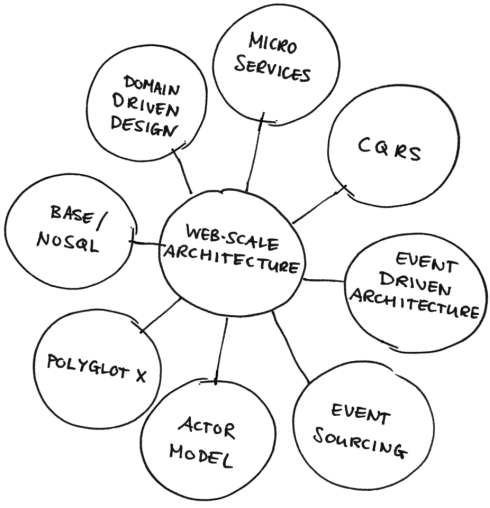
In deze roman volgen we de reis van twee CIO's die elkaar tegenkomen op vlucht WS7102.
Peter heeft net de zwaarste week uit zijn carrière achter de rug en is blij dat hij even zijn zorgen achter zich kan laten. Moet hij niet eens op zoek naar een andere baan? Daar wil hij tijdens de vlucht over nadenken, maar door de ontmoeting met vakgenoot Harold loopt alles anders dan hij van tevoren had verwacht.

Tijdens de reis wisselen Harold en Peter ook ervaringen met elkaar uit op het gebied van IT-architectuur. Zo hoort Peter gedurende de reis van zijn collega Harold wat web-scale architectuur inhoudt, waarom hij ervoor heeft gekozen en hoe ook Peters organisatie ervan kan profiteren.

Web-scale IT werd enkele jaren geleden geïntroduceerd door analistenbureau Gartner. Het wordt nog vooral gezien als een visie op IT die geïnspireerd is door de praktijk bij Silicon Valley-reuzen als Google en Facebook; 'web-scale' betekent in dit geval extreem flexibel en schaalbaar. Een web-scale software architectuur maakt dit mogelijk.

Hoe je die architectuur precies invult, daarover was Gartner minder uitgesproken. Bij Info Support hebben we daar wel een visie op ontwikkeld; een heel concrete invulling van het begrip web-scale architectuur. Een architectuur die volgens ons is opgebouwd uit acht

onderwerpen die nauw met elkaar samenhangen, te weten: microservices, CQRS, event driven architecture, event sourcing, actor model, polyglot 'X', BASE / NoSQL en domain driven design. In dit boek komen de meeste van deze onderwerpen en hun onderlinge samenhang aan bod.



Het idee van deze fictieve IT-roman is losjes gebaseerd op The Phoenix Project, een boek uit 2013 waarin het concept DevOps uiteen wordt gezet. Niet geheel toevallig, want zowel DevOps (een agile manier van samenwerken in teams) als continuous delivery (het in korte cycli updaten van software) passen uitstekend bij web-scale architectuur.

Maar daarover later dus meer, in dit boek. Je kunt Vlucht WS7102 beschouwen als een combinatie van een reisverhaal en een fictieve casestudy. Het ziekenhuis van Peter en de luchtvaartmaatschappij van Harold zijn allebei verzonnen, maar ze zouden wel kunnen bestaan.
Dat bleek tijdens het schrijven van dit boek; vlak na de eerste brainstorm kwamen de schrijvers erachter dat de crisis in het ziekenhuis uit het begin van dit verhaal in de echte wereld bijna één op één had plaatsgevonden.

Of dat ziekenhuis nu ook zijn heil zoekt in web-scale architectuur, dat weten we niet. Wie weet is het voer voor een tweede deel...

Veel leesplezier!

1. Een "kleine" update

De CIO die wij zoeken, gaat uitdagingen niet uit de weg en weet zich in onverwachte situaties flexibel op te stellen.

"Dat ben ik!" Peter de Graaf hoort het zichzelf hardop zeggen. En dat terwijl er niemand is die hem kan horen. Eindelijk is hij even alleen. Eindelijk hoeft hij aan niemand verantwoording af te leggen. Dat was de afgelopen week wel anders.

Peters werkgever, het Albert Havik Ziekenhuis, had met de dag meer schade opgelopen. Op een gegeven moment kon zelfs niemand meer het operatieschema raadplegen, laat staan dat er nieuwe operaties konden worden ingepland. Dagenlang hadden de landelijke kranten het debacle op de voet gevolgd en de reputatieschade voor het ziekenhuis was met de dag groter geworden.
Nee, dat was geen aangename week geweest. Het laatste zetje om nu toch echt maar eens uit te kijken naar een nieuwe uitdaging.

Terwijl hij de vacature in zijn notitie-app opslaat, vraagt hij het zichzelf voor de zoveelste keer af. Had hij het allemaal kunnen zien aankomen? Het was eigenlijk maar een heel kleine update geweest. Een standaardrelease. Alles was tot in de puntjes gepland, de livegang in het weekend was soepel verlopen.

Jubelstemming alom – veel mensen hadden lang op deze release zitten wachten.

Ook op maandagochtend aanvankelijk alleen maar blije gezichten. Maar aan het eind van de ochtend kwamen de eerste meldingen bij de servicedesk binnen dat de planningsmodule eruit lag en binnen de kortste keren viel het systeem als een kaartenhuis in elkaar. Nog voor de lunch stond Peter met de softwarearchitect tekst en uitleg te geven bij de directeur.

Daarna kwam alles in een stroomversnelling: terwijl er achter de schermen hard werd gewerkt aan een patch, had Peter samen met zijn directeur en de PR-afdeling een persconferentie voorbereid. De media zouden er toch wel lucht van krijgen, dus dan kon je ze maar beter voor zijn. Wat volgde waren dagen en halve nachten van zwoegen om meer ellende te voorkomen.

Achteraf hadden de consultants die hij had ingehuurd om het systeem weer up en running te krijgen het natuurlijk allemaal aan zien komen. Een nieuwe zoekfunctie in het elektronisch patiëntendossier, dat ook nog eens van encryptie wordt voorzien, dat is natuurlijk een gegarandeerd recept voor oververhitte servers. Nee, de consultants hadden dat wel geweten.

Maar ja, met de kennis van nu is het makkelijk oordelen. Tijd om die kleine update grondig te testen was er niet geweest. Als ze die handige zoekfunctionaliteit nu niet hadden geïmplementeerd, hadden ze opnieuw

een half jaar moeten wachten tot de volgende release. Dan was het hele ziekenhuis over Peter heen gevallen.

Ook dan.

Enfin, het is nu achter de rug. Een geluk bij een ongeluk is dat het systeem weer in de lucht is vlak voordat Peter vertrekt naar de Gartner Summit, toch wel een jaarlijks hoogtepunt. Tijd voor een verzetje. Even alles proberen te vergeten en ondertussen maar eens nadenken over de toekomst.

2. Een grote monoliet

"Welkom aan boord, kan ik u misschien een krant aanbieden?" Peter schudt zijn hoofd. Na de ellende van de afgelopen dagen kan hij geen krant meer zien. In de Telegraaf van vandaag zal wel te lezen zijn dat de problemen bij het ziekenhuis zijn opgelost, maar dan ongetwijfeld in een kort berichtje op pagina 7. De kop *ZIEKENHUIS AAN INFUUS*, in chocoladeletters over de volle breedte van de voorpagina van vorige week vrijdag, doet hem nog huiveren. Nee, nu even geen krant.

Het grootste voordeel van businessclass vliegen, is misschien wel dat je snel bij je stoel bent. Eindelijk even neerploffen en nergens aan denken. Met een diepe zucht laat Peter zich zakken in de comfortabele stoel.

"Alles oké?", hoort hij vanuit de stoel naast hem.

"Ja hoor, super!" Klonk dat overdreven? Peter is echt zó blij dat hij zit.

"Mooi zo. Ik hoorde u zo diep zuchten."

"Zeg maar jij. Ach ja... Ik zal me even voorstellen. Peter de Graaf."

"Toch niet de Peter de Graaf van..." De wijsvinger van Peters buurman gaat naar de krant die hij in zijn

linkerhand houdt. Bijna tegelijk lezen ze hardop wat er staat, in een stukje van één kolom op pagina 9.

Volgens IT-manager Peter de Graaf zijn de problemen met het informatiesysteem van het Albert Havik Ziekenhuis nu eindelijk opgelost.

"Toevallig wel ja. Vanwaar de interesse? Ook IT'er?"

"Inderdaad. Harold ten Kate, CIO bij MaxAir. Aangenaam!"

"Ha, een vakgenoot! Bij een luchtvaartmaatschappij nog wel. Aangenaam. Nou, ik ben wel blij dat ik toch nog naar die Gartner Summit kan."

"Snap ik. Ik heb er ook zin in. Maar... Petje af hoor. Je hebt goede damage control gedaan."

"Je wilt niet weten hoeveel liter zweet ik de afgelopen week ben kwijtgeraakt."

"Bespaar me de details", zegt Harold met een brede glimlach. "Volgens mij heb jij gewoon enorme pech gehad met die update. Wil je er eigenlijk wel over praten? Of liever eerst een borrel?"

Nou, een Johnnie Walker zou er wel ingaan...

Peter steekt van wal. "Weet je, we zagen dit totaal niet aankomen. Iedereen was zo blij dat die update er ein-

delijk was. Blijkt er door nieuwe fuzzy zoekfunctionaliteit een bug geïntroduceerd te zijn in het raadplegen van het elektronisch patiëntendossier. Waardoor alle processen tot stilstand kwamen."

"Dat EPD, daar is zeker ieder ander systeem bij jullie van afhankelijk?"

"Ja, inderdaad. Elke afdeling moet de patiëntkaart kunnen raadplegen en daarop mutaties doorvoeren rondom de behandeling. Als die functionaliteit het niet doet, kunnen ze dus niks."

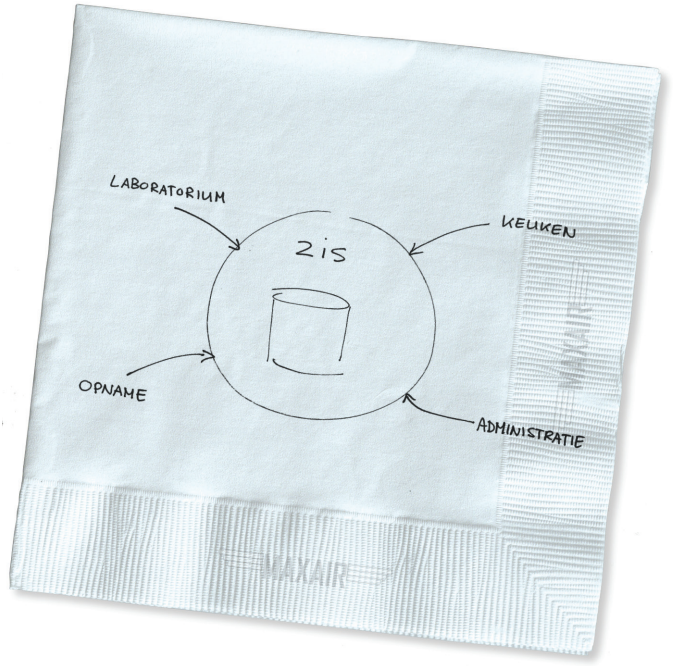
"Aha. Dus eigenlijk hebben jullie een centrale afhankelijkheid, een single point of failure, in jullie landschap geïntroduceerd met dat EPD."

"Een enkel systeem voor alle afdelingen binnen je bedrijf is een bedrijfsrisico"

"Misschien wel. Maar goed, we hebben niet zomaar voor dit EPD gekozen hè. Daar is een uitgebreid selectietraject aan voorafgegaan. Met dit systeem kunnen we voor veel afdelingen de functionaliteit in één keer afdekken. En daar zijn we op zich wel blij mee, want het is een hele vooruitgang ten opzichte van de oude situatie."

"Vroeger hadden jullie zeker veel maatwerk?"

Peter knikt. “Precies. En allerlei wettelijke vereisten en standaard-declaratieprocessen zijn afgedekt en daar willen we eigenlijk ook niet onze IT-capaciteit aan spenderen. We zijn in die zin niet uniek als ziekenhuis voor onze backoffice-processen. Maar ja, dat brengt ook nadelen met zich mee; we zitten nu wel vast aan de halfjaarlijkse releasecyclus van die leverancier.
En als de leverancier onze wensen te specifiek vindt, willen ze die lang niet altijd doorvoeren.
Kijk, zo ziet ons Ziekenhuis Informatie Systeem er grofweg uit.”



Hoewel alle problemen van de afgelopen week weer naar boven komen, voelt Peter zich ontspannener dan hij zich in weken heeft gevoeld. De whisky doet uiteraard zijn werk, maar dat hij een vakgenoot treft die eens echt naar zijn verhaal wil luisteren, stemt hem bijna vrolijk.
Hij kan zich in het ziekenhuis geen collega voor de geest halen die zo empathisch is als Harold.

"Ik begrijp het helemaal", antwoordt Harold. "Onze luchtvaartmaatschappij maakt ook gebruik van grote ERP-pakketten waar we erg blij mee zijn. Maar we moeten er wel voor zorgen dat bugs in die pakketten niet ons hele landschap onderuit trekken. Daar hebben we inmiddels gelukkig een hele mooie oplossing voor gevonden."

"Oh?"

Harold gaat er eens goed voor zitten. "Kijk, twee jaar geleden zaten we in exact dezelfde situatie als jullie nu. Toen het ERP-pakket er even uit lag, konden we niet eens meer behoorlijk mensen laten inchecken. Alle passagiersgegevens waren namelijk alleen maar vastgelegd in dat ene ERP-pakket, vanuit het bekende architectuurprincipe dat je de gegevens één keer vastlegt en meervoudig gebruikt. Alles moest toen met de hand, zelfs dingen die ogenschijnlijk niets met dat pakket te maken hadden. Toen bleek dat veel specifieke applicaties van afdelingen via services automatisch gekoppeld waren en daardoor stopten met

functioneren toen dat pakket even uit de lucht was. Onze architecten hebben toen gekeken of ze op een of andere manier de rest van het landschap runtime-onafhankelijk van dat ERP-pakket konden maken. Je weet wel, stabiel doordraaien zodra iets anders even niet beschikbaar is. We hebben een goede oplossing gevonden: een microservices-architectuur. Daarbij communiceren autonome, relatief kleine services door middel van berichten met elkaar."

3. Geduld is een schone zaak

"Cabin crew, boarding completed."

Dit gaat een lange vlucht worden, maar bang dat hij zich zal vervelen, is Peter niet. Ze zijn nog niet vertrokken of hij heeft al zoveel interessants gehoord dat de trip nu al de moeite waard is.
Nog even zijn berichten checken.
Beste Peter, het Universitair Medisch Centrum Groningen is enthousiast. Ze hebben gelezen hoe je de zaak hebt opgelost. Volgende week eens informeel bijpraten, schikt dat?

Groningen, dat betekent verhuizen. Maar een informeel gesprek kan nooit kwaad. *Graag! Ik ben maandag terug!* appt Peter voordat hij de vliegtuigmodus van zijn telefoon inschakelt.

Die Harold werkte bij MaxAir in het verleden dus precies op dezelfde manier als het Albert Havik Ziekenhuis nu.

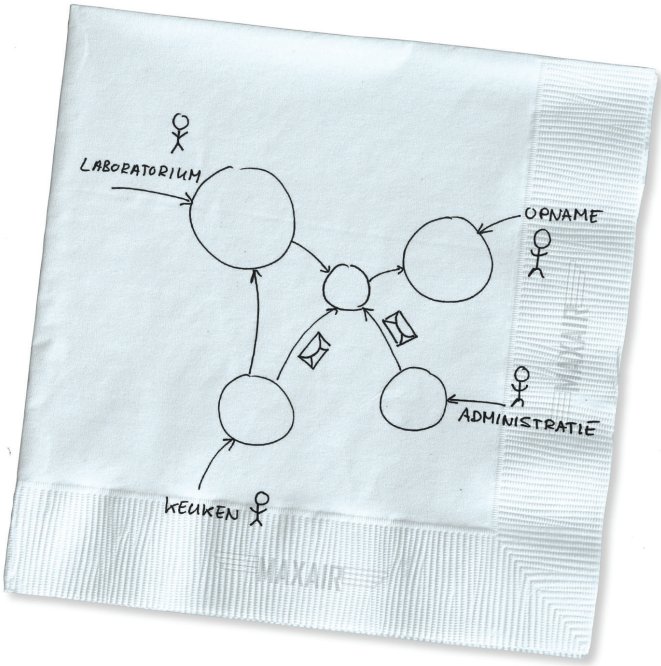
"Maar wacht even", zegt Peter. "Autonoom functionerende onderdelen, betekent dat niet dat je te maken krijgt met dataduplicatie? Als onze softwarearchitecten ergens allergisch voor zijn.... Laten we zeggen dat we wat minder prettige ervaringen hebben met een heleboel losstaande databases die in het client-server-tijdperk zijn ontstaan. De stap naar een ERP-sys-

teem hebben we juist gemaakt om al die ellende van 'verschillende waarheden' in verschillende systemen op te lossen."

"Ja, herkenbaar", antwoordt Harold. "Je wilt inderdaad niet dat allerlei verschillende applicaties dezelfde gegevens beheren, met als gevolg dat er allemaal verschillende waarheden over een passagier of patiënt rondzwerven in je systeem. Maar er zijn andere, misschien wel betere oplossingen dan alles maar terugbrengen tot één monoliet; het bepalen van eigenaarschap van data. Je spreekt dan af dat iedereen wel een kopie van de gegevens mag bewaren, maar dat maar één systeem verantwoordelijk is voor het bewerken van die gegevens. Als dat ene systeem er dan om een of andere reden uit ligt, zoals ons ERP-systeem of jullie EPD, kan de rest van de applicaties gewoon door blijven draaien op hun eigen data."

"Oké, dus op die manier kun je meerdere autonome systemen definiëren, met hun eigen logica en eigen dataverzameling en die ook nog eens optimaal inrichten voor de afdeling die het gebruikt."

"Precies! Eens kijken... Bij jullie zou het er ongeveer zo uit kunnen zien." Harold pakt een pen en begint enthousiast te schetsen.



Peter denkt hardop na. "Dat zou ons inderdaad wel helpen om die gespecialiseerde systemen echt te richten op specifieke wensen van die afdeling. Die afdelingen klagen nogal eens over de doorlooptijd van wijzigingen omdat we ziekenhuis-breed nieuwe functies in het systeem moeten prioriteren.
Maar als niet elke afdeling naar exact dezelfde gegevens kijkt, kan dat toch ook problemen opleveren? Stel nu dat mutaties in het centrale systeem niet direct worden doorgevoerd in het systeem van de afdeling. Dan zit die afdeling dus naar oude data te kijken."

Die vraag had Harold kennelijk verwacht. "Ja, je moet inderdaad rekening houden met een zekere vorm van tijdelijke inconsistentie. De vraag is alleen hoe erg dat is. Ik bedoel, maakt het echt heel veel uit dat mutaties van patiëntgegevens een minuut of misschien een paar seconden later in een ander systeem terechtkomen?"

"Beschikbaarheid van systemen is vaak belangrijker dan het overal kunnen beschikken over de allerlaatste gegevens"

"Hmm, lang niet altijd. Een adreswijziging van een patiënt is niet direct van belang bij de behandeling. Sterker nog, de afdelingssystemen hebben die adresgegevens eigenlijk helemaal niet nodig. Maar voor ons is er wel een heel belangrijk onderdeel: de patiëntenkaart. Daar staat op welke medicijnen iemand toegediend heeft gekregen, waar hij allergisch voor is en dergelijke. Als een patiënt bijvoorbeeld op de eerste hulp binnenkomt, behandeld wordt en daarna richting de OK gaat, moeten ze daar wel de meest actuele informatie hebben."

"Daar kan ik me wat bij voorstellen", zegt Harold lachend. "Dat is behoorlijk cruciale informatie. Maar hoe gaan jullie daar dan nu mee om? Wat doen jullie als die gegevens niet beschikbaar zijn door een falend EPD?"

"Voor dit soort cruciale gegevens hanteren we het principe dat er altijd een papieren kopie moet meereizen met de patiënt met de laatste gegevens. We hebben dan ook niet altijd tijd om überhaupt het systeem bij te werken. Het gebeurt regelmatig dat we dat pas na de operatie kunnen doen."

"Exact. Voor dat soort gevallen heb je procedures waar je op terug kunt vallen. Dat is niet anders in het geval van losgekoppelde systemen. Uiteindelijk kan alles een keer uitvallen. Het voordeel van losse systemen is wel dat je vaak nog steeds informatie uit het ene systeem kunt halen terwijl het andere er uit ligt, zoals het raadplegen van patiëntgegevens in de OK als het EPD even niet beschikbaar is. De meest recente updates staan op papier, maar de rest van de gegevens kun je er dan wel bij pakken. En vergeet niet, wanneer alle systemen weer beschikbaar zijn, worden de berichten alsnog verwerkt en zijn de gegevens weer consistent. *Eventual consistency* noemen ze dat. Je kiest er dan voor om beschikbaarheid van informatie belangrijker te maken dan de gegarandeerde consistentie op elk punt in de tijd."

Peter begint het te begrijpen. "Oké, als we goed naar de bedrijfsprocessen en de overdrachtsmomenten tussen afdelingen kijken, kunnen we volgens mij prima werken vanuit een model op basis van eventual consistency en zouden we dus best zo'n microservices-architectuur kunnen toepassen. Het EPD blijft dan dus gewoon bestaan, maar is dankzij de services en de

uitwisseling van berichten wel maximaal ontkoppeld van de afdelingssystemen.
Ik zie alleen nog wel een andere uitdaging: hoe krijgen we zoveel verschillende kleine services in samenhang getest en in productie? Het was al zo'n drama om het EPD werkend te krijgen en dat is maar één ding."

Harold veert op. "Ah, en hier wordt het dus pas echt interessant!"

4. Durf klein te denken

Peter krijgt het weer benauwd als hij eraan denkt hoeveel tijd en moeite het kostte om dat EPD in de lucht te krijgen. Grote releases, trage updates, het kost je jaren van je leven. Hoe doet Harold dat met al die microservices?

Harold drinkt met een ferme slok zijn glas leeg en begint te vertellen. "Bij een monoliet concentreert al het werk zich rond de grote releasemomenten. Daarbij bepaalt de leverancier vaak min of meer wanneer zo'n releasemoment is. Vooral de testinspanning is meestal enorm. En daarbij moeten systemen vaak een heel weekend in onderhoud, moeten mensen stand-by zijn... Je kent het wel.
Het idee van microservices is juist dat je iedere service niet alleen los kunt draaien maar ook los kunt ontwikkelen en naar productie kunt brengen. Dus steeds hele kleine stukjes functionaliteit, waarbij testwerk, risico en impact ook beperkt zijn tot dat kleine stukje van je systeem. Dat betekent ook dat die kleine brokjes functionaliteit veel vaker live kunnen worden gezet dan voorheen met grote releases, waardoor je veel sneller waarde voor je organisatie toe kunt voegen."

"Ah ja, continuous delivery, toch? Ik las daar iets over op de site van de Summit."

"Precies, die term zullen we de komende dagen nog

wel een paar keer horen, denk ik."

Peter herinnert zich wat er nog meer op de site stond en heeft direct een vraag. "Maar ik dacht dat dat alleen iets was voor de grote tech-bedrijven, de Googles en de Amazons. Want hoe ga je zoiets organiseren? Ik bedoel, het klinkt allemaal prachtig, maar hoe ga je die services apart naar productie brengen, terwijl ze qua functionaliteit uiteindelijk allemaal van elkaar afhankelijk zijn?"

"Ik zeg ook niet dat het makkelijk is. Services moeten bijvoorbeeld zo geprogrammeerd worden dat de koppelingen met andere services blijven werken als er nieuwe versies naar productie worden gebracht. Maar dat trucje is zeker niet uitsluitend weggelegd voor de grote tech-bedrijven. Je zult alleen wel moeten investeren in de kennis en vaardigheden van je IT-personeel om dat goed uit te voeren."

"Snap ik. Het klinkt eigenlijk ook allemaal zo logisch. Als ik een update op mijn mobiel of laptop binnenkrijg, verwacht ik ook dat alle apps die ik geïnstalleerd heb door blijven draaien. Maar zoals je weet, speelt geld ook altijd een rol en de IT-organisatie kost het ziekenhuis nu al een behoorlijke duit. De invoering van het EPD was al geen sinecure, ook financieel niet. Ik weet niet hoe ik dat moet verkopen aan het MT en de directie. Feitelijk zeg ik: de vorige investering heeft niet gebracht wat we hoopten. Ik heb nog meer geld en tijd nodig."

"Dat is altijd een terechte zorg", weet Harold. "Maar laat ik het eens omdraaien: hoe belangrijk is de IT-afdeling voor jouw ziekenhuis tegenwoordig?"

Daar kan Peter over meepraten. De IT-afdeling is natuurlijk essentieel. Hij heeft zelf ervaren wat er gebeurt als de systemen niet meer goed functioneren. Voor je het weet, komt je ziekenhuis bekend te staan als onbetrouwbaar en in een ziekenhuis gaat het om mensenlevens.

"Natuurlijk, IT is essentieel," antwoordt Peter. "Daarom investeren we er fors in. Maar het blijft een kostenpost op de begroting. Het budget is niet oneindig."

"Een kostenpost zeg je... Interessant. Vertel me eens, hoe onderscheiden jullie je van omringende ziekenhuizen uit de regio?"

"Tja, nu je het vraagt. Naast het feit dat we natuurlijk goede zorg verlenen, proberen we ook een stapje verder te gaan in onze service. Bijvoorbeeld door het beperken van wachttijden, het beperken van fouten in de administratie, en goede klantervaring en een persoonlijke benadering. Daar zijn we met deze catastrofe rondom het EPD helaas niet bepaald in geslaagd de laatste tijd..."

"Dus als ik het goed begrijp, is IT niet alleen belangrijk om de functie van ziekenhuis te ondersteunen, maar speelt het ook een essentiële rol bij het streepje voor

dat je bij je personeel, de zorgverzekeraars en de patiënten kunt halen. Is de IT-afdeling dan niet eerder een omzetgenerende afdeling dan een kostenpost?"

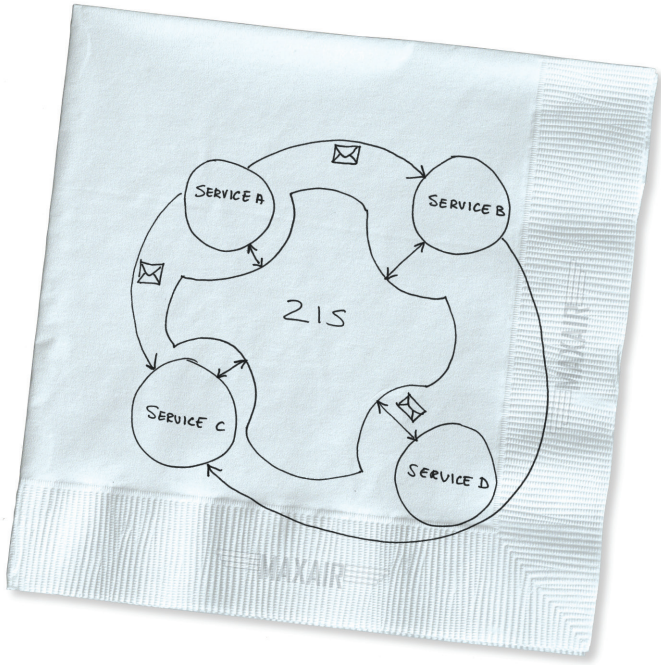
Peter knikt voorzichtig. "Hmm, zo had ik er als CIO zelf nog niet eens naar gekeken... Ik heb investeringen via de directie altijd moeten loskrijgen door met allerlei doemscenario's op de proppen te komen of acute problemen op te lossen, maar ze zijn natuurlijk veel eerder geneigd te investeren als ze zien wat voor kansen er voor ons op dat gebied liggen!"

"Het is inderdaad maar net hoe je er naar kijkt", zegt Harold lachend. "Het gaat niet alleen om problemen voorkomen, je organisatie wordt ook een stuk wendbaarder doordat je IT-systemen flexibeler worden. Als je dat eenmaal inziet, is investeren in de kennis en kunde van je personeel een no-brainer. En bedenk ook dat de nieuwe manier van werken binnen je IT-afdeling uiteindelijk net zo normaal wordt als dat je nu gewend bent. Je moet wel natuurlijk wel eerst een behoorlijk steile leercurve door, maar daarna ga je er vanzelf de vruchten van plukken."

Peter moet zeggen dat het idee van die microservices hem als een heel welkome oplossing klinkt. Toch zit iets hem niet lekker. Hij denkt nog steeds aan het enorme elektronische patiëntendossier dat nog maar vijf maanden geleden is opgeleverd. Hij had zelfs een etentje voor de hele IT-afdeling georganiseerd om de mijlpaal te vieren. Dat kan dus blijven bestaan, maar

hoe werkt dat alles dan samen? En worden de gebruikers er niet gek van? Hij ziet in gedachten de klagende verplegers, artsen en collega's van de informatiebalie alweer voor zijn bureau staan.

De CIO van de luchtvaartmaatschappij weet niet alleen het nodige, hij lijkt ook Peters gedachten te kunnen lezen. "Wij hadden ook één groot systeem. We zijn anderhalf jaar geleden begonnen een schil om dat systeem heen te leggen, met services die het pakket ontsluiten. Ons ERP-systeem is nog gewoon in gebruik, maar daarnaast hebben we steeds meer services erbij geplaatst. Je kunt ook gewoon met de gebruikersinterface van het EPD blijven werken."



"Het inzetten van microservices levert flexibiliteit en wendbaarheid voor de organisatie"

"Oké, dus met microservices heb je minder kans dat andere functies en andere afdelingen in de problemen komen, maar het probleem van de zoekfunctionaliteit is dan nog niet opgelost."

"Dat gaan we straks doen."

5. Lezen en schrijven

Hoe vaak hij inmiddels ook een vliegtuig van binnen heeft gezien, het moment dat het toestel van de grond loskomt, bezorgt Peter nog altijd een wat onbestemd gevoel. Het is die fractie van een seconde waarin de wielen het asfalt verlaten, waarin het vliegtuig ook altijd even een schommeling lijkt te maken.

"Gaat het, buurman?"

"Oh ja, sorry." Peter weet eigenlijk niet waarom hij zich verontschuldigt. " Ik heb geen moeite met vliegen, maar dat opstijgen..."

"Weet je wat helpt?", vraagt Harold. Hij wuift naar een stewardess. Of ze nog iets kunnen bestellen? Dat kan pas weer als de riemen los mogen.

"Eigenlijk geldt voor jullie planningssysteem toch ook dat het vooral belangrijk is om de afspraken in te zien, klopt dat? Dan kunnen die afspraken doorgaan en hoeven er geen mensen weggestuurd te worden?"

Peter knikt. "Dat is wel belangrijker dan het toevoegen. Alhoewel we natuurlijk ook wel afspraken moeten kunnen inplannen."

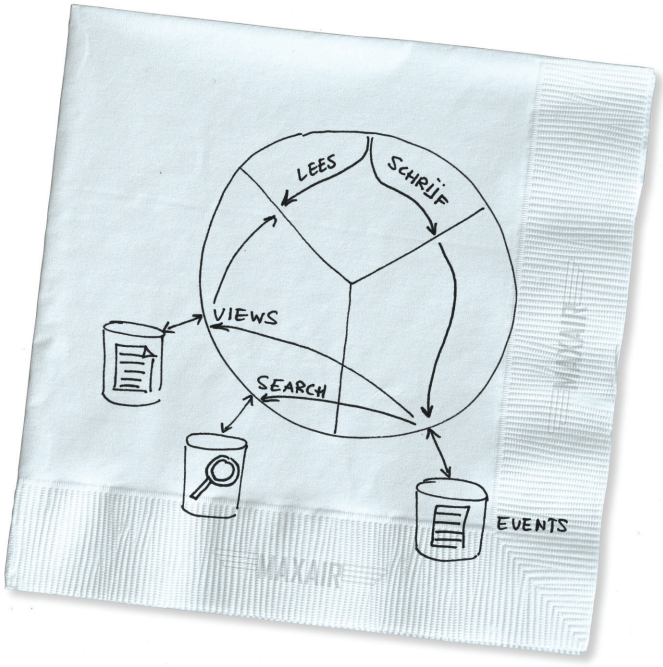
Harold is weer op dreef. "Wat nou als je de functionaliteit rondom planning ontwerpt als twee losse

microservices? Dan kun je een microservice maken die verantwoordelijk is voor het registreren van afspraken. En een tweede voor het inzien van gemaakte afspraken. Die twee services zijn beide onderdeel van het planningsdomein, maar hebben een verschillend doel. Dat patroon wordt ook wel CQRS genoemd; dat staat voor Command Query Responsibility Segregation. Vrij vertaald betekent dat het splitsen van de schrijfkant (command) en de leeskant (query's). In jouw voorbeeld is dus de service die verantwoordelijk is voor het registreren van afspraken de schrijfkant en de service voor het inzien van afspraken de leeskant.

Het mooie hiervan is dat het systeem niet meer zo kwetsbaar is. Als de schrijfkant een tijdje uit de lucht is, bijvoorbeeld omdat er een fout optreedt tijdens het doorvoeren van een systeemupdate, dan is de leeskant nog online en kunnen de medewerkers nog steeds de afspraken inzien. En hoeven er dus geen mensen weggestuurd te worden."

Peter moet toegeven dat dat bijzonder aanlokkelijk klinkt. "Maar, als die twee microservices helemaal los van elkaar staan, hoe blijven die dan in-sync? Er moet toch wel iets van een koppeling zijn?"

Harold kijkt opnieuw alsof hij deze vraag had verwacht. "De microservice die gaat over het inplannen bevat altijd de waarheid. Dit noemen we wel het 'system of record' of 'single point of truth'. Vanuit de schrijfkant kunnen we dan berichtjes publiceren op



basis waarvan de leeskant zijn data kan bijwerken. Iedere keer als er een nieuwe afspraak wordt ingepland, kan er bijvoorbeeld een bericht worden verstuurd dat de relevante informatie over de afspraak bevat. De leeskant kan de informatie in de berichten gebruiken om zijn eigen database te vullen. De leeskant die je raadpleegt, staat dan dus helemaal los van de schrijfkant."

En toch wringt het bij Peter, en dat zit hem wederom in de dataduplicatie. "Dus je bedoelt dat er tussen het inplannen en synchroniseren met de leeskant wat tijd

kan zitten? Als ik een afspraak inplan, moet die toch direct zichtbaar zijn voor de medewerkers?"

"Ligt eraan wat je direct noemt", antwoordt Harold. "Stel dat het nu vijf of tien seconden duurt voordat de leeskant is bijgewerkt. Is dat dan een probleem?

"Nee, eigenlijk niet. Nu je dat zo zegt valt dat eigenlijk wel mee. Afspraken worden altijd minimaal een dag van tevoren ingepland."

"Dat is vaak zo", beaamt Harold. "Dit is dus zo'n situatie waarin je te maken krijgt met eventual consistency. Het voordeel is dat het ten goede komt aan de beschikbaarheid van de individuele services."

"Het gebruik van CQRS verhoogt de beschikbaarheid van systemen"

"Oké, je hebt me overtuigd dat dit bij ons planningsysteem mogelijk is. Maar het lijkt me niet dat dat altijd kan."

"Niet altijd", zegt Harold. "Er zijn situaties waarin je nu eenmaal transacties nodig hebt om twee of meer systemen gesynchroniseerd te houden. Het is altijd een afweging. Uiteindelijk gaat het erom wat belangrijker is in een bepaalde situatie: beschikbaarheid of consistentie. En uiteindelijk wordt deze keuze bepaald

door wat de meeste waarde oplevert voor de business. In jouw situatie is beschikbaarheid duidelijk het belangrijkste. Het registreren van afspraken zou je bij wijze van spreken nog tijdelijk op papier kunnen doen als de schrijfkant down is. Niet ideaal, maar het is een uitzonderingssituatie."

Net op het moment dat het vliegtuig recht lijkt te gaan vliegen en het wachten is totdat het fasten your seatbelt-lampje uitgaat, klinkt het door de intercom: "Wij kunnen helaas nog niet bij u langskomen. Op dit moment ervaren we meer turbulentie dan normaal. Wij verzoeken u te blijven zitten met uw stoelriem vast."

Vliegangst is irrationeel, denkt Peter. En turbulentie is een uitdaging.

6. Waar blijft de tijd?

Achteraf had Peter die Johnnie Walker ook helemaal niet nodig gehad om het grootste deel van de vliegreis te slapen als een blok. Het zal de vermoeidheid wel zijn, denkt hij. In de verte ziet hij al de lichten van Las Vegas uit het raam van het vliegtuig.

Vlak voordat het toestel lijkt te gaan landen, zwelt het geluid van de motoren onverwacht aan. De neus van het toestel wordt weer opgetrokken en het vliegtuig draait naar rechts, weg van de landingsbaan.

"Hé, een doorstart", zegt Harold, met een stem die een zekere mate van opwinding niet kan verbergen. "Dat zal met de sterke zijwind te maken hebben."

"Good evening, ladies and gentlemen. This is your captain speaking. As you may have noticed..."

Peter rolt met zijn ogen. "Nog een rondje woestijn dan maar." Hij weet niet wat hem ongeduldiger maakt: het ongemak van het vliegtuig, het vooruitzicht van een hotelkamer met bed of het vooruitzicht dat hij straks wellicht weer een berichtje heeft over een mogelijke nieuwe baan.

Na een extra rondje om het vliegveld lukt de tweede landingspoging gelukkig wel. Blijkbaar is Peter niet de enige die daar zichtbaar door is opgelucht, want er

klinkt een applaus vanuit de economy class. Normaal gesproken associeert Peter dit eerder met goedkope chartervluchten, maar vandaag ziet hij zichzelf opgelucht meeklappen. Ergere turbulentie dan op deze vlucht heeft hij nog nooit meegemaakt. Zijn buurman had nog zo geprobeerd hem gerust te stellen: een vliegtuig is geen monoliet en de kans dat je neerstort is kleiner dan de kans dat je de Staatsloterij wint, maar zelfs Harold had af en toe toch wat zorgelijk gekeken.

Op dat moment pakt Harold zijn telefoon. "Kijk, dit is onze nieuwste app. Vorig jaar gelanceerd. In het onderdeel vluchtinformatie kun je nu zien dat we geland zijn. Daarvoor hebben we een aantal services gebouwd die ook gebruikmaken van CQRS. Kijk maar: hier zie je dat er een doorstart heeft plaatsgevonden."

"Wacht even, dat soort informatie wil je toch helemaal niet aan je klanten tonen?", vraagt Peter verbaasd.

"Nee, dat is inderdaad niet voor iedereen relevant. Ik kan dat zelf zien omdat ik een interne versie van de app gebruik die alleen voor werknemers is. Deze versie raadpleegt andere read models dan de consumentenversie."

"Geinig. Mocht de vliegtuigmodus trouwens al uit?"

"We hebben wifi aan boord, wist je dat niet?"

Peter kan zichzelf wel voor zijn hoofd slaan. Hij selec-

teert het wifikanaal en terwijl zijn berichtenservice een aantal nieuwe notificaties laat zien, downloadt Peter de app van MaxAir. "Verrek, je hebt gelijk. Op mijn telefoon zie ik inderdaad alleen dat we geland zijn, met twintig minuten vertraging."

"Precies!", zegt Harold enthousiast. "En realiseer je dat er als gevolg van die gebeurtenissen, of die 'events', nog veel meer gebeurt. Het feit dat wij een doorstart hebben gemaakt, heeft er op dit drukke moment waarschijnlijk toe geleid dat andere toestellen een extra rondje hebben moeten vliegen en dat de approach planning in de software van de luchtverkeersleiding automatisch is bijgewerkt. Wanneer dat heeft geleid tot vertraging van bepaalde vluchten is dat weer te zien op de borden van de luchthaven. Allemaal het gevolg van een paar events."

"Waarmee alles dus uiteindelijk weer 'eventually consistent' is?"

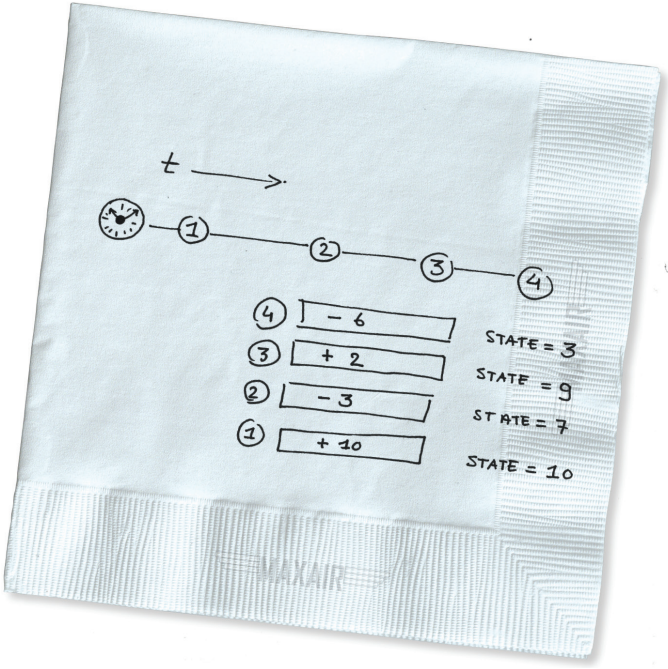
"Ja, inderdaad. De oorspronkelijke gebeurtenissen, zoals 'Vliegtuig geland' en 'Vliegtuig doorgestart' worden opgeslagen in een microservice die de schrijfkant vertegenwoordigt. Andere services vormen de leeskant en werken de weergaven bij die je nu in de app ziet."

"Hoe sla je zo'n gebeurtenis dan op aan die schrijfkant?", wil Peter weten. "Is dat dan gewoon een relationele database?"

"Dat is zeker mogelijk. Wij hebben ervoor gekozen om gebruik te maken van event sourcing. Dat is een alternatieve manier om de verzameling gebeurtenissen die hebben geleid tot een bepaalde status op te slaan; niet in een relationele database, maar als een lijst van events die zijn opgetreden in de tijd. Omdat we alleen events toevoegen aan de lijst, worden updates sneller doorgevoerd dan bij de traditionele aanpak waarbij we vaak gegevens moeten vergrendelen als we ze bijwerken om concurrencyproblemen te voorkomen."

Peter is nu toch wat in verwarring. "Wacht even, dat klinkt complex. Komt het er in feite niet op neer dat je de status van het vliegtuig vastlegt? Is het niet eenvoudiger om een statusveld te maken in de database waar je een update op kunt doen als er nieuwe events binnenkomen?"

"Nee, juist niet. Het probleem met updates van databases is dat je informatie vernietigt. Als we de informatie in de database wijzigen op basis van binnenkomende events houden we zogenaamde 'current state' bij. We weten dan alleen nog wat de laatste status is van een vliegtuig. Dat was niet genoeg voor ons systeem. Wij willen de volledige status van het vliegtuig kunnen opvragen, oftewel alle events die opgetreden zijn. Op die manier kunnen we bijvoorbeeld ook zien hoeveel tijd er tussen de doorstart en de uiteindelijke landing zat."



Peter laat de woorden even op zich inwerken. "Dat klinkt een beetje als een audit-log."

"Zeker! Dat is een automatisch bijeffect van event sourcing. Hierbij is het niet nodig de huidige situatie vast te leggen met daarnaast een aparte audit log die beschrijft hoe die huidige situatie ontstaan is. Event sourcing dekt beide behoeften."

"Maar dat lijkt me wel lastig query'en dan. Hoe kom ik er nu bijvoorbeeld eenvoudig achter welke vliegtuigen de status 'geland' hebben?", vraagt Peter zich af.

Uiteraard heeft Harold ook hier weer een antwoord op. "Dat is nu net waar we de read models voor gebruiken. Omdat we de verantwoordelijkheden voor het schrijven en lezen van elkaar losgemaakt hebben, is het niet nodig om uitgebreide query-functionaliteit te ondersteunen in de event store, waar al die gebeurtenissen zijn opgeslagen. Dat maakt een event store ook eenvoudiger van opzet dan een volledige database.
Daar komt nog bij dat we de 'current state' van een vliegtuig kunnen achterhalen door alle events vanaf het begin af aan te lezen. Sterker nog, we kunnen nu ook de status van een vliegtuig op een willekeurig tijdstip bepalen door alleen naar de events te kijken die zich voor dat tijdstip hebben afgespeeld."

"Met event sourcing wordt de dimensie tijd een expliciet concept in de software"

Peter vraagt zich af waarom MaxAir deze gegevens allemaal zou willen bewaren. "Doen jullie dit puur voor auditing?"

"Soms wil je nu eenmaal heel specifieke informatie terug kunnen zien", zegt Harold. "Voor ons is het bijvoorbeeld heel handig om te weten hoe mensen die een vlucht boeken hun stoelen kiezen. Klikken ze direct op de plaats die ze reserveren? Of klikken ze op verschillende stoelen en komen ze terug op hun eerste

keuze als de extra beenruimte toch wel erg veel duurder blijkt te zijn? Als je dat soort analyses wilt kunnen doen heb je de historische events nodig."

"Maar als nu achteraf blijkt dat er een verkeerd event heeft plaatsgevonden?" Peter is blij dat hij ondanks de lange vlucht nog tamelijk scherp is. "Stel nou dat een klant zijn of haar geselecteerde stoel bij het boeken op een later moment wijzigt. Dan moet je dus je hele verzameling events door om het betreffende event terug te vinden en te wijzigen? Dat is een nogal bewerkelijke actie."

"Dat is nu juist het mooie van event sourcing; je kunt events niet wijzigen. Nooit. Dat staat gelijk aan geschiedvervalsing, ze hebben namelijk ooit plaatsgevonden, dat is een feit. Maar nieuwe events die een correctie bewerkstelligen, zijn wel toegestaan. Vergelijk het met de werkwijze van accountants. Je weet toch dat accountants nooit een potlood en gum gebruiken? Je moet altijd aan kunnen tonen waar een transactie vandaan komt. In plaats van een event weg te gooien, voeg je een compenserend event toe."

Geen geschiedvervalsing. Daar wordt Peter wel enthousiast van: alle wijzigingen in het planningsysteem zouden op die manier terug te vinden moeten zijn.

Peter checkt zijn mail: wéér een bericht van een potentiële werkgever. Nu met concreet aanbod voor een baan. Nog mooier dan die in Groningen. Toch is er die

plotselinge twijfel; opeens weet hij niet meer zo zeker of hij wel klaar is bij zijn huidige werkgever. Misschien is er nog te veel te doen...

7. Ieder zijn eigen ding

"Passengers of flight WS7102, proceed to luggage belt 42."

Geprikkeld door zijn discussie met Harold kan Peter het niet laten om door te vragen. "Dat systeem weet dus dat we geland zijn en waar we onze koffers kunnen oppikken. Komt dat nou uit hetzelfde systeem als die app?"

"Nee. Sterker nog: het systeem dat je hier ziet is helemaal niet van MaxAir, maar van McCarran International Airport. Deze luchthaven maakt trouwens nog gebruik van een mainframe; het systeem wordt bijgewerkt op basis van de events die wij als vliegmaatschappij publiceren en blijft zo up-to-date."

"En zit daar dan ook een event store onder?"

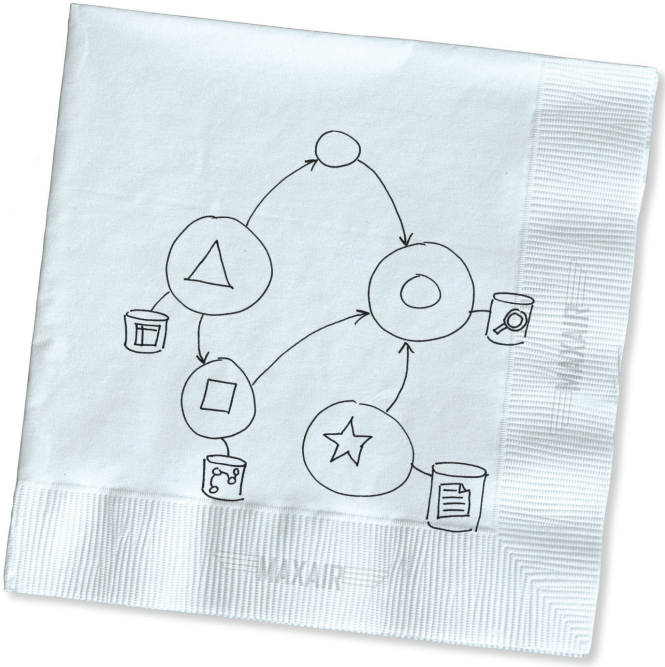
"Wederom nee", knipoogt Harold. "En dat is ook helemaal niet nodig. De luchthaven is alleen geïnteresseerd in de laatste stand van zaken op basis van de events die wij publiceren. Als ik me niet vergis, zit hier een relationele database onder. Maar dat hoeft niet altijd de beste keuze te zijn. In onze app bieden we bijvoorbeeld de mogelijkheid aan om relaties te leggen met andere gebruikers. Om dit vast te leggen, gebruiken we geen relationele database, maar een graph-database. Daarin kun je namelijk eenvoudig verschillen-

de relaties vastleggen zonder dat we daarvoor allerlei koppeltabellen moeten introduceren, zoals we over het algemeen in een relationele database doen."

"Daar kan ik me wel iets bij voorstellen. Je kunt dus verschillende databasetypes gebruiken, afhankelijk van de functionaliteit die je aan de gebruikers wilt bieden?"

"Precies. Een ander voorbeeld is zoekfunctionaliteit. Hiervoor wordt vaak een search engine gebruikt die goed is in het indexeren van gegevens en complexe zoekopdrachten mogelijk maakt. En dat geldt niet alleen voor opslag, maar ook voor je architectuur en de technologiestack die je gebruikt. Ook die moet passen bij de oplossing die je wilt bieden. Voor onze microservices hebben wij bijvoorbeeld gekozen voor Java, maar onze front-end is gebouwd met ASP.NET omdat ons front-end team hiermee het meest productief is. Dus in plaats van één standaard af te dwingen voor alle teams binnen de organisatie, kiest elk team datgene wat het beste past bij hen en de gewenste oplossing. Dit noemen we ook wel Polyglot 'X'."

"Een team dat gebruik kan maken van fit for purpose tools en frameworks kan optimaal presteren"



Dat klinkt Peter als muziek in de oren. "Dus je hoeft niet meer alle teams om te laten scholen als je een nieuwe technologie wilt introduceren? Briljant! Maar hoe voorkom je dan dat je een wildgroei aan verschillende technologiestacks en frameworks krijgt?"

"Je wilt je teams wel de ruimte geven om de beste tools voor de oplossing te kiezen, maar uiteraard wel binnen bepaalde kaders en richtlijnen. Hierin moeten ook de applicatiearchitectuur en beheeraspecten meegenomen worden."

Vrijheid om te kiezen. Het klinkt Peter als een kleine revolutie; nooit meer gesteggel tussen Java- en .NET-developers. Ieder team kiest zijn eigen taal, als de microservices maar met elkaar kunnen communiceren.

Hij betrapt zich erop dat hij sterk de neiging moet onderdrukken om zich volledig te laten meeslepen door het verhaal van Harold. Zal deze aanpak een volgende crisis met het planningsysteem kunnen voorkomen? In zijn hoofd klopt het plaatje en het lijkt eigenlijk allemaal heel logisch, maar werkt het ook in het ziekenhuis? Eerst maar eens bespreken met zijn team, stelt hij zichzelf gerust.
Eén ding staat wel vast: Peter ziet niet meer op tegen die eerste teammeeting na Las Vegas. Hij kijkt er zowaar naar uit!

8. Een positieve uitslag

"Harold, kerel! Is de pijn alweer wat gezakt?"

"Jawel. Dat is maar goed ook, want het was niet om te harden."

"Ik ken het, heb ook eens een gebroken been gehad. Maar dan wel van de skivakantie en niet van de rolband op Schiphol..."

"En die aardige dame me maar waarschuwen: mind your step... Mij zul je er nooit meer op zien, op die ondingen."

De onfortuinlijke afloop van de reis naar Las Vegas is Harold duidelijk niet in de koude kleren gaan zitten.

"Hoe lang moet je nog in het gips, denk je?"

"De dokter kwam net langs: de uitslag van het onderzoek is positief, dus ik denk dat ik weer snel de oude ben."

"Ligt eraan wat voor test. In een ziekenhuis betekent een positieve testuitslag niet altijd iets goeds..."

"Ha! Nee, ik bedoel gewoon 'positief' als in 'gunstig'."

"Kwestie van beroepsdeformatie", lacht Peter. "Wij

zijn ook nog de enige afdeling binnen het ziekenhuis die nog niet is genezen van SOA's. Je zou er trouwens versteld van staan hoe groot het risico op spraakverwarring in de medische wereld is. Dat geldt zelfs voor zoiets simpels als 'links' en 'rechts'; het is maar net of je vanuit de patiënt denkt, of vanuit jezelf redeneert. Als je daar geen hele strenge regels voor opstelt, krijg je te maken met verkeerd afgezette ledematen."

"Dat is in ons vakgebied natuurlijk niet anders." Harold komt langzaam een stukje overeind. "Wij hebben daar in het verleden behoorlijk mee geworsteld. In het begin, toen we nog maar een paar microservices hadden, leek het allemaal nog wel overzichtelijk. Maar naarmate er meer afdelingen betrokken raakten kwamen er ook meer spraakverwarringen tussen onze ontwikkelaars en de gebruikers van de systemen. We realiseerden ons dat we betere afspraken moesten maken over de gebruikte terminologie en zijn toen DDD, domain driven design, gaan toepassen in onze projecten."

Het is ook wat, denkt Peter. Binnen vijf minuten gaat het alweer over ons werk. Over beroepsdeformatie gesproken...

"Een belangrijk onderdeel van DDD is het opstellen van een universele taal. We noemen dat de 'ubiquitous language'.", gaat Harold verder. "Die zorgt ervoor dat ontwikkelaars, gebruikers en domeindeskundigen elkaar allemaal goed kunnen begrijpen. Deze taal wordt zelfs gebruikt in de programmacode, wat fouten in de

software moet voorkomen."

> "Het inzetten van DDD zorgt ervoor dat domeinexperts, ontwikkelaars en eindgebruikers elkaar beter kunnen begrijpen"

Peter is sceptisch, want Harolds verhaal roept weer wat vragen op. "Een aantal jaar geleden hebben wij zo'n universele taal voor het ziekenhuis geprobeerd op te stellen. Toen werd dat nog een enterprise wide canonical datamodel genoemd. Dat project is uiteindelijk op niets uitgelopen. Het model werd simpelweg veel te complex. De financiële software heeft namelijk heel andere gegevens van een patiënt nodig dan de software die de operaties inplant, die weer afwijkt van de software die de ziekenhuisapotheek gebruikt. Om dat allemaal op elkaar af te stemmen, dat is bijna niet te doen."

"Zo'n canonical datamodel zou wel erg mooi zijn, maar in onze ervaring is het binnen een grotere organisatie inderdaad gewoon niet haalbaar", vertelt Harold.

"Grotere organisaties bestaan nu eenmaal uit eilandjes met eigen verantwoordelijkheden, met hun eigen definities en jargon. In DDD noemen we zo'n eilandje een bounded context en iedere bounded context heeft zijn eigen ubiquitous language. Zo kan bijvoorbeeld

de financiële afdeling met hun eigen definite van een patiënt blijven werken."

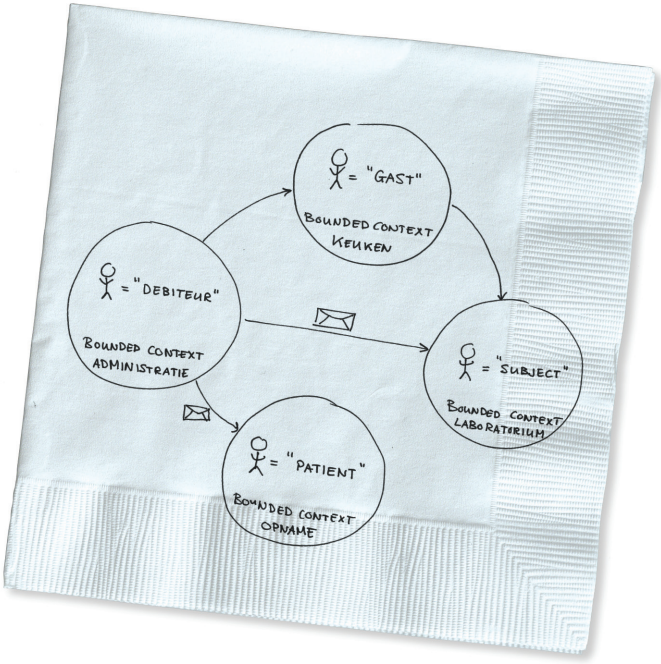
"Dan snap ik het toch nog niet helemaal. Het is toch juist het idee dat afdelingen informatie kunnen uitwisselen?"

"Klopt, maar daarvoor hoeven we alleen maar afspraken te maken over de gegevens die uitgewisseld worden tussen verschillende bounded contexten. Dat zijn vaak veel minder gegevens dan binnen de bounded context gebruikt worden. Daarvoor zou je wel een aantal generieke termen kunnen gebruiken die bedrijfsbreed worden afgesproken. In jouw ziekenhuis kan ik me voorstellen dat jullie gebruik maken van een patiëntnummer en dat iedereen daar hetzelfde onder verstaat en waarmee iedereen een patiënt uniek kan identificeren."

Daar kan Peter zich wel iets bij voorstellen. "Maar hoe weet ik nou welke bounded contexts er allemaal zijn? Moet ik hierover gaan brainstormen met domeindeskundigen en ze dan verder gaan invullen?"

"Ja, inderdaad", bevestigt Harold. "Om die contexten te ontdekken en te bepalen wat echt bij elkaar hoort, passen we event storming toe. Daarmee brengen we heel snel, met brown paper en stickies, samen met de mensen van de afdeling in kaart welke processen we moeten ondersteunen. Ook bepalen we dan welke relevante gebeurtenissen of business events er optreden

binnen deze processen. Wanneer je die afhankelijkheden scherp hebt is het relatief eenvoudig om onderdelen te ontdekken die een sterke samenhang hebben. Die stoppen we samen in zo'n context."



Hoe interessant het ook klinkt, Peter laat merken dat hij toch nog wat sceptisch is. "Dus na zo'n event storming sessie heb je je complete domein met alle afhankelijkheden en interacties in kaart? Dat klinkt eerlijk gezegd een beetje te mooi om waar te zijn. Hoe krijg je ooit overeenstemming over dat hele model met alle betrokkenen? Net vertel je me dat een canonical

datamodel een slecht idee is, maar dit lijkt toch wel heel erg sterk op een variant daarvan, maar dan zelfs met de processen erbij."

"Dan heb ik het misschien niet helemaal goed uitgelegd", lacht Harold. "Het idee van al die onafhankelijke domeinen is juist dat ze ook onafhankelijk van elkaar zullen veranderen. Daarom hoeft een model ook niet bedrijfsbreed te worden afgestemd. De inhoud van die domeinen kan door de teams zelf worden ingevuld, samen met de domeinexperts. Tussen de bounded contexts moet je dan vertalingen maken van begrippen uit de ene context naar begrippen uit de andere context, maar dan gaat het vaak maar om een klein gedeelte van alle begrippen. Onderdelen die intensief samenwerken zullen immers al binnen dezelfde bounded context geplaatst zijn."

"Dat klinkt mooi", zegt Peter. "Het lijkt erop dat DDD je een hoop handvatten biedt om met de verschillende domeinen binnen je systeemlandschap om te gaan. Maar aan de andere kant voelt het voor sommige situaties ook wel een beetje als overkill. Binnen ons ziekenhuis kunnen complete bounded contexts volgens mij worden ingevuld door standaardpakketten."

Harold vervolgt: "Je hoeft het zeker niet overal tot in detail toe te passen. Eigenlijk kun je per bounded context de keuze maken of je daarbinnen ook de principes van DDD gaat toepassen. En vaak is dat alleen nodig voor een aantal kernactiviteiten, waarin je als

organisatie ook echt onderscheidend wilt zijn. Onze luchtvaartmaatschappij heeft er bijvoorbeeld voor gekozen om te concurreren op kostprijs. Daarom is het erg belangrijk dat onze vliegtuigen zoveel mogelijk passagiers kunnen vervoeren en zo kort mogelijk uit de lucht zijn vanwege onderhoud. Voor de logistiek die daar achter zit hebben we zelf een slimme planningsservice gebouwd omdat dat voor ons een cruciale functionaliteit is die we nog niet in de markt konden vinden."

"Dus als ik het goed begrijp, creëer je onafhankelijkheid door je landschap op te knippen in bounded contexts en daarbinnen microservices toe te passen. En dat levert dan flexibiliteit. Hierdoor kun je plaatselijk bijvoorbeeld je eerder gemaakte keuzes bijstellen als dat nodig is. Of dat nou is doordat je een standaardpakket tegen bent gekomen dat je huidige maatwerk kan vervangen, of omdat één van je bedrijfsprocessen door een interne reorganisatie is veranderd."

"Klopt, je hebt dus de mogelijkheid om bounded contexts te veranderen qua invulling", gaat Harold verder. "Vanuit de best practices van DDD wordt dit ook aangeraden. Zo gauw een model begint af te wijken van de realiteit van het bedrijfsproces moet je de benodigde veranderingen gewoon doorvoeren. Ook als ze redelijk rigoureus zijn. Omdat de microservices binnen de bounded contexts ook onafhankelijk van elkaar zijn, wordt het eenvoudiger om dergelijke aanpassingen door te voeren. In DDD-termen wordt

dat ook wel supple design genoemd."

Peter krabt even achter z'n oor terwijl hij de laatste woorden van Harold op zich in laat werken. Harold blijft maar met goede ideeën komen. "Oh jee, hoe laat is het eigenlijk!?" Peter realiseert zich ineens dat ze alweer even hebben bijgepraat. Hij moet zo weer een belangrijke vergadering met de directie bijwonen.

"Harold, ondanks dat jij hier met een fractuur ligt, heb je mij weer een grotere dienst bewezen in deze korte tijd dan wij jou. Ik moet er nu helaas wel snel van door. Als je nog iets van me nodig hebt, laat het me weten, dan regel ik het. En zorg dat je snel weer op de been bent!"

"Ach, dat valt denk ik wel mee. En maak je niet druk Peter, ik word hier prima verzorgd. Ik hoop je snel weer eens te spreken. Ik stuur je wel een appje."

Na een stevige handdruk en een welgemeende glimlach loopt Peter de zaal af richting de bestuurskamer. Terwijl hij over de gangen loopt, gaan alle nieuwe ideeën en concepten die hij via Harold en bij Gartner langs heeft horen komen door z'n hoofd. Direct na de conferentie heeft Peter de directie weten te overtuigen van het belang en zelfs de noodzaak om web-scale architectuur te gaan inzetten om de problemen die binnen het IT-landschap spelen het hoofd te bieden. Hij heeft toen ook zijn architectuurteam de opdracht gegeven om eens goed in de nieuwe concepten te

duiken en een paar veranderingen voorzichtig toe te gaan passen. Na wat vallen en opstaan, waar Harold al voor gewaarschuwd had, begonnen de teams het door te krijgen en tot steeds dieper inzicht te komen. Het duurde niet lang voordat de planningsafdeling de meest stabiele software afleverde en ook nog eens het meest productief bleek. Daarna begonnen andere teams natuurlijk nieuwsgierig te worden, en inmiddels wordt het hele IT-landschap binnen het ziekenhuis opgebouwd uit kleine, autonome services en hebben ze geen last meer gehad van complete black-outs.

Terwijl Peter zich dat realiseert legt hij zijn hand op de deur van de bestuurskamer. Normaal gesproken waren dit vergaderingen waar hij vreselijk tegenop zag. Vandaag niet. Vanaf nu niet meer. Nu zijn het de vergaderingen waarbij hij goed nieuws kan brengen en investeringen loskrijgt. Hij haalt nog even diep adem, voelt een glimlach opkomen, en stapt vol zelfvertrouwen de bestuurskamer in.

9. Een paar tips

Wat een jaar! Terwijl hij de laatste haring van de tent in de grond slaat, bedenkt Peter dat die crisis in het ziekenhuis alweer negen maanden achter hem ligt. En vlak daarna die trip naar Las Vegas…

Een welverdiende vakantie mag je het gerust noemen. Eerst die zoekfunctie gepatcht, daarna het hele Ziekenhuis Informatie Systeem stukje bij beetje wendbaarder gemaakt. En en passant nog even de ene na de andere aanbieding voor prachtige banen afgeslagen. Nee, hij heeft goed werk kunnen verrichten bij zijn huidige werkgever en daar zelfs de nodige waardering voor gekregen. Het komende jaar is er bovendien nog meer dan genoeg te doen.

Maar nu is het tijd voor twee weken ontspanning, op een eenvoudige camping vlakbij een Frans gehucht waar hij beslist geen Nederlandse CIO's tegen zal komen.

"Je hoeft echt je mail niet te checken hoor," had de CEO van het ziekenhuis hem nog op het hart gedrukt. Hij had eigenlijk wel gelijk. Ze hadden een prima team en mócht een van de functies van het systeem down gaan, dan zou in ieder geval niet de hele boel in elkaar storten.

Niet je mail checken, hoe moeilijk is dat! Tenslotte is

wifi zelfs doorgedrongen tot de meest bosrijke gebieden van het Franse platteland. Ach, de tent staat, de kinderen vermaken zich al in het speeltuintje, dus waarom ook niet.

Veel interessants heeft de mailbox niet te bieden. 'Storing inschrijfsysteem patiënten opgelost', mooi. 'Vacature hoofd ICT Ministerie van Veiligheid', verwijderen. Hee, een bericht van Harold.

"Peter, alles oké? Met mij alles weer prima. Moet je dit lezen!"
Harold heeft in het mailtje een krantenartikel geplakt: 'Geen superdag voor supermarktketen.'
In het stuk vertelt de CIO van TopMarkt Nederland dat de problemen met het informatiesysteem van het concern nu eindelijk zijn opgelost. Een kleine update had ervoor gezorgd dat de voorraden niet meer werden aangevuld. Het gevolg: boze klanten, gefrustreerde supermarktmedewerkers en radeloze managers.

"Arme man, ik weet wat je hebt doorgemaakt", denkt hij terwijl hij zichzelf een glas bier inschenkt.

"Goedemiddag, sorry dat ik stoor!" Toch een andere Nederlander op deze camping? Dat verbaast Peter.

"Hallo," begroet Peter zijn landgenoot, die zijn auto blijkbaar een paar plekken verderop heeft neergezet.

"Misschien een gekke vraag, maar ik wil mijn tent op-

zetten, blijkt dat ik mijn hamer ben vergeten. Beetje haastig van huis vertrokken. Zou ik die van u misschien even mogen lenen?"

"Natuurlijk, zeg maar jij. Ik zal me trouwens even voorstellen: Peter de Graaf."

"Lieuwe van der Woude. O, dat is heel fijn, dank je."

Dit kan niet waar zijn. Die naam heeft Peter net nog ergens gelezen. Hij ontgrendelt zijn tablet waarop het bericht over de problemen bij TopMarkt nog openstaat.
"Toch niet de Lieuwe van der Woude van..."

"Toevallig wel ja. Vanwaar die interesse? Ook IT'er?"

"Ja, ook CIO zelfs, van het Albert Havik Ziekenhuis. Zeg, die tent kan vast wel even wachten.
Ook zin in een biertje? Ik denk dat ik wel een paar tips voor je heb."

Vlucht WS7102

fast forward naar web-scale architectuur

"Ik zal me even voorstellen. Peter de Graaf."

"Toch niet de Peter de Graaf van..." De wijsvinger van Peters buurman gaat naar de krant die hij in zijn linkerhand houdt. Bijna tegelijk lezen ze hardop wat er staat, in een stukje van één kolom op pagina 9.

Volgens IT-manager Peter de Graaf zijn de problemen met het informatiesysteem van het Albert Havik Ziekenhuis nu eindelijk opgelost.

Het zal je gebeuren: heb je net het dieptepunt in je carrière achter de rug, zit je een hele vlucht naar een congres in Las Vegas naast een collega-CIO. Tijdens hun reis verkennen de CIOs verschillende concepten van web-scale architectuur: van microservices, CQRS en event sourcing, tot domain driven design. Wat houden deze begrippen in, waarom zou je ervoor kiezen om ze in te zetten en welke afwegingen moet je zoal maken? Vlucht WS7102 blijkt voor Peter een ware ontdekkingsreis.

infoSupport
Solid Innovator