SMART Requirements 2.0

ÖMER AYDINLI, EDWIN HENDRIKS
EN JASPER ZANDVLIET

ACADEMIC SERVICE

SMART Requirements 2.0

ÖMER AYDINLI, EDWIN HENDRIKS
EN JASPER ZANDVLIET

ACADEMIC SERVICE

Meer informatie over deze en andere uitgaven kunt u verkrijgen bij:
Sdu Klantenservice
Postbus 20014
2500 EA Den Haag
tel.: (070) 378 98 80
www.sdu.nl/service

Academic Service is een imprint van Sdu Uitgevers bv.

Grafische vormgeving: Solid-ontwerp.nl
Omslagontwerp: Sjef Nix, Amsterdam
Druk- en bindwerk: Wilco bv, Amersfoort

ISBN 978 90 12 58583 5
NUR 991

Inhoudsopgave

Voorwoord

In de wereld van softwareontwikkeling doen zich veel problemen voor. Een van de meest wezenlijke is het ontbreken van eenduidige requirements en specificaties. Dit zorgt niet alleen vaak voor niet-optimale eindresultaten, ook levert het nogal eens frictie op tussen enerzijds de (business-)klant en anderzijds de ICT-ers. De klant is bijvoorbeeld teleurgesteld in ICT, omdat deze niet levert wat hij wil, danwel veel later en duurder dan noodzakelijk. Anderzijds wijst ICT naar de klant, die maar niet duidelijk wil worden in zijn wensen. Ook de modernere ontwikkelmethoden bieden jammer genoeg niet altijd een gedegen oplossing.

De droom om dit bronprobleem binnen de softwareontwikkelcyclus op te lossen is het startsein geweest voor de ontwikkeling van SMART Requirements. Deze methode is voortgekomen uit zowel de wetenschap als de dagelijkse praktijk van softwareontwikkeling en heeft in diezelfde praktijk aangetoond het hiervoor geschetste probleem te kunnen oplossen, met alle voordelige gevolgen van dien.

Graag willen wij alle personen bedanken die een bijdrage hebben geleverd aan de methode en aan dit boek. Onder andere onze collega's die ons hebben geïnspireerd, gemotiveerd en uitgedaagd. De collega's die ons hebben geholpen met het reviewen van ons werk. En in het bijzonder Richard Bremer die de katalysator is geweest voor de totstandkoming van de methode. Zonder zijn passie en drive was dit boek er misschien wel nooit gekomen.

Daarnaast willen wij dit boek opdragen aan professor Kees Koster. Zonder dat hij het misschien geweten heeft, hebben wij veel aan hem te danken en is een groot aantal van zijn ideeën in deze methode terechtgekomen. Wij hopen hem eer aan te doen door met behulp van deze methode, en dus ook zijn creatieve geest, softwareontwikkeling op een hoger, effectiever en efficiënter niveau te brengen.

Als laatste willen wij jou, de lezer van dit boek, veel plezier wensen bij het lezen ervan. Hopelijk kan dit boek voor jou net zo veel betekenen als het voor alle SMART Requirements-projecten heeft gedaan.

1. Inleiding en overzicht

Je staat in een rij met mensen die elkaar een geheime boodschap doorgeven. Het zachte gemompel komt steeds dichterbij en dan is het zover. Je sluit je ogen en spitst je oren. Je doet je uiterste best om te luisteren naar wat jouw buurman in je oor fluistert. Een lastige opgave, vooral omdat je geacht wordt alles te onthouden. Je hoort een onsamenhangend verhaal met allerlei obscure details en vraagt jezelf af hoe je dit ooit gaat overbrengen. Nu is het jouw beurt. Je buigt je voorover en begint met jouw verhaal. Maar wat was de boodschap ook al weer? Je fronst je wenkbrauwen en probeert ijlings zoveel mogelijk details op te lepelen. Je ratelt maar door, terwijl je weet dat je een hoop dingen bent vergeten en van alles door elkaar haalt. De klok tikt door. In paniek verzin je er daarom maar wat details bij. Nadat er nog een paar minuten zijn verstreken is het aan de laatste in rij om de geheime boodschap wereldkundig te maken. Maar hè, dat verhaal lijkt van geen kant op wat je net aan degene naast je hebt verteld. Hilariteit alom! Wie kent er niet het spelletje van de Chinese Whispers?

Een ongebruikelijke manier om een boek over requirements mee te beginnen? Helemaal niet! Want zijn er niet een hoop overeenkomsten tussen dit verhaal over de Chinese Whispers en het vraagstuk van business & ICT alignment?

Dit boek gaat over 'SMART Requirements'. Een methode voor requirements engineering en -management, die bestaande inzichten en technieken combineert met nieuwe, unieke elementen, zoals:

- een raamwerk met een duidelijk onderscheid tussen het business-, proces- en systeemniveau, dat houvast geeft en overlap en dubbel werk voorkomt;
- een checklist van zeven criteria om tot SMART Requirements te komen;
- een gestructureerde manier om uit het gewenste resultaat een proces af te leiden dat geen onnodige handelingen bevat en waarin geen enkele stap mist;
- een vorm van specificeren die eenduidigheid afdwingt.

SMART Requirements is een methode voor de ontwikkeling van software die de business optimaal ondersteunt. Het is opgebouwd als een modulair sjabloon, zodanig dat het prima samengaat met reeds bestaande aanpakken,

zoals agile, iteratief of waterval. De methode richt zich op softwareontwikkeling en brengt deze in lijn met de wens van de business. Hierin zijn het de functionele requirements binnen informatieverwerkende processen die volledig SMART gemaakt worden.

De belangrijkste redenen om met SMART Requirements te werken zijn:

1. *Kostenbesparing:* We maken requirements en specificaties in een vroeg stadium van een project SMART. Hierdoor komen de totale kosten voor softwareontwikkeling aanzienlijk lager uit.

2. *Business-ICT alignment:* We starten met het bepalen van het gewenste bedrijfsresultaat. Alle vervolgacties worden hierop gebaseerd. De ontwikkelde software zal dus het bereiken van het bedrijfsresultaat optimaal ondersteunen.

3. *First Time Right:* De methode genereert een opvallend hoog niveau van SMARTness, in een vroegtijdig projectstadium. Dit maakt het mogelijk hoogwaardige producten op te leveren, die minder fouten bevatten. Hierdoor zijn minder reviews en dus ook minder versies van documenten nodig en bevat de software minder fouten.

4. *Uitbesteding van ICT:* Met de methode zorgen we voor eenduidigheid in wat er door de software bereikt moet worden en hiervoor is slechts een minimum aan informatie nodig. De opdrachtgever kan hierdoor eenvoudig en ondubbelzinnig zijn wensen kenbaar maken en voor de ontwikkelende partij is geheel duidelijk wat de bedoeling is. Hierdoor kunnen zij direct aan de slag. Dit biedt grote voordelen als softwareontwikkeling is uitbesteed aan een andere afdeling, een externe leverancier of bij offshoring situaties.

5. *Risicobeperking:* De belangrijkste reden dat projecten uitlopen en te veel geld kosten, is onduidelijkheid over het te bereiken resultaat. De methode fixeert mensen op dit resultaat. Dit resulteert in effectiviteit bij het uitvoeren van de handelingen om het resultaat te bereiken. Het voorkomt daarnaast het verzanden in oude ideeën of het herhalen van oude oplossingen.

6. *Kennisborging:* Projecten zijn vaak afhankelijk van de kennis van enkele kritieke medewerkers. Het komt daarom regelmatig voor dat deze resources een bottleneck vormen bij het tijdig kunnen uitvoeren van projecten. Met de methode wordt kennis op een transparante wijze vastgelegd en inzichtelijk gemaakt. Hiermee verdwijnt het knelpunt van de kritieke medewerker.

Dit boek legt de methode uit, biedt concrete handvatten voor het gestructureerd verzamelen en vastleggen van eenduidige requirements. Daarom is het als naslagwerk en handboek geschikt voor alle betrokkenen bij softwareontwikkeling. Denk hierbij aan informatie- en business-analisten, functioneel ontwerpers, bouwers, testers, projectmanagers, etc. Het uiteindelijke doel is dat zij allen in staat zijn om oplossingen te ontwikkelen, die één op één overeenkomen met de oorspronkelijke klantwens.

We gaan weer even terug naar de Chinese Whisper. Als we kijken naar het gehele traject waarbij de klant haar wensen uit, tot aan het moment dat deze in de vorm van software zijn uitgewerkt, dan kunnen we dit zien als een 'Whisper-rij'. Iedere schakel in dit proces levert iets op waar de volgende in lijn gebruik van moet maken. Maar wat gebeurt er als er gaandeweg details verloren gaan of als mensen elkaar niet goed begrijpen? Wat wordt opgeleverd zal dan niet voldoen aan de wens die oorspronkelijk is geuit. En waar dit bij de Chinese Whispers vermakelijk is, zijn de gevolgen bij softwareontwikkeling soms bijna niet te overzien.

Laten we ons nu eens het volgende voorstellen. We spelen het spel nog een keer, maar dit keer praten alle mensen in de rij een andere taal. Daarnaast vragen we aan de eerste deelnemer om in een paar woorden een idee te geven en gaandeweg het doorgeven moet dit idee tot ontwikkeling komen. De laatste in lijn moet vervolgens het hele verhaal vertellen, precies zoals de eerste in de rij dit in zijn hoofd had. We voelen haarfijn aan dat dit een onmogelijke opgave is. En, hoewel niet zo zwart-wit is als hier beschreven, zijn dit soort zaken binnen softwareontwikkeling aan de orde van de dag.

Om de theorie in dit boek wat luister bij te zetten en verder te verduidelijken zullen we regelmatig een praktijkgeval beschrijven. Dit zijn voorbeelden uit de praktijk die hebben bijgedragen aan het ontwikkelen van SMART Requirements of waar we de methode hebben toegepast. Als eerste een praktijkgeval dat de Chinese Whispers binnen softwareontwikkeling illustreert.

Chinese Whispers bij softwareontwikkeling

Een organisatie werkt met een bedrijfskritisch systeem, dat al decennia in gebruik is. Een typisch voorbeeld van een legacy-systeem. Gedurende de jaren is dit systeem steeds weer uitgebreid. Er zijn allerlei armen aangehangen, waarbij de ontwerpbeslissingen vaak van een twijfelachtig allooi waren. En, zoals wel vaker met dit soort systemen, is de documentatie sterk verouderd of ontbreekt in zijn geheel. Initieel was dit geen groot probleem, omdat de ontwikkelaars het systeem door en door kenden en

daarbij over uitvoerige business-kennis beschikten. Business en ontwikkelaars spraken zogezegd dezelfde taal. De interne klant kon in een paar steekwoorden uitleggen welke wijzigingen noodzakelijk waren en de ontwikkelaar verwerkte deze direct in het systeem.

Op een bepaald moment besluit de organisatie dat zij wil gaan outsourcen. Men kan zich dan beter richten op haar kernactiviteiten en een marktpartij kan doen waar zij goed in is, namelijk het onderhouden van dit legacysysteem. Er worden afspraken gemaakt, waarbij groots wordt ingezet op offshoring van de ontwikkeling in India. Op deze manier kan de prijs van de functionaliteit die ontwikkeld moet worden laag worden gehouden.

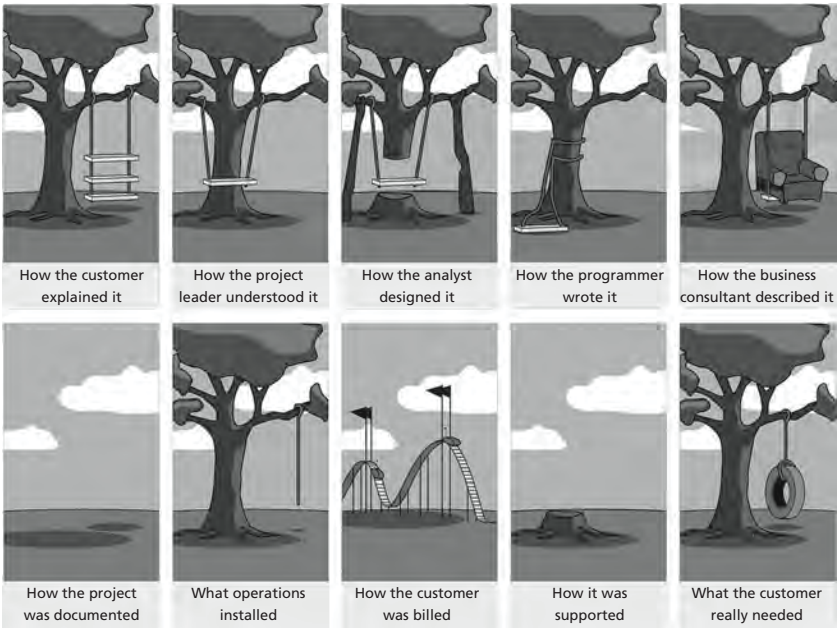
Het idee is dat de oude, in house ontwikkelaars gedurende een aantal weken de nieuwe, Indiase ontwikkelaars opleiden. Hun systeemkennis moet dus worden overgedragen. De interne klant kan dan via de analisten van de marktpartij hun wensen kenbaar maken, zodat deze worden doorgevoerd binnen het systeem. Iedereen tevreden... zou je denken.

Hier beginnen de problemen echter. Indiase ontwikkelaars beschikken namelijk niet over de business-kennis die de in house ontwikkelaars wel hadden. Ook de interne klant moet ineens gaan nadenken over de manier waarop zij haar wensen (requirements!) kenbaar wil maken. De afstand tussen business en ICT is veel groter geworden en zij spreken niet meer in hetzelfde jargon.

Dit resulteert in het aanleveren van vage beschrijvingen, waarin allerlei noodzakelijke details ontbreken. Het gaat hier om een groot contract, dus de marktpartij besluit deze te accepteren, zodat zij hun klant tevreden houden. De analisten hebben echter grote moeite met het vertalen van deze beschrijvingen naar concrete ontwerpen, zodat de Indiase ontwikkelaars hiermee aan de slag kunnen. Dit betekent dat zij veelvuldig terug moeten naar de business om vragen te stellen en onduidelijkheden te laten ophelderen. Daarnaast zijn door het ontbreken van de juiste details grote inspanningen nodig om te kunnen testen.

Uiteindelijk worden grote groepen met Indiase ontwikkelaars naar Nederland gehaald. Het team met relatief dure Nederlandse analisten moet worden opgeschaald om de Indiase collega's van de juiste informatie te voorzien. De prijs van de te ontwikkelen functionaliteit stijgt navenant en dekt op een gegeven moment niet meer de kosten. De marktpartij voelt zich genoodzaakt het contract met verlies uit te voeren. De klant is daarbij ook nog eens ontevreden, omdat het ontwikkelen veel langer duurt dan begroot en de kwaliteit onder de maat is.

Deze situatie laat zich goed vangen in de inmiddels klassiek geworden 'project cartoon'. En dit voorbeeld staat uiteraard niet op zichzelf, niet voor niets herkennen we allemaal onderstaand plaatje binnen onze dagelijkse praktijk.



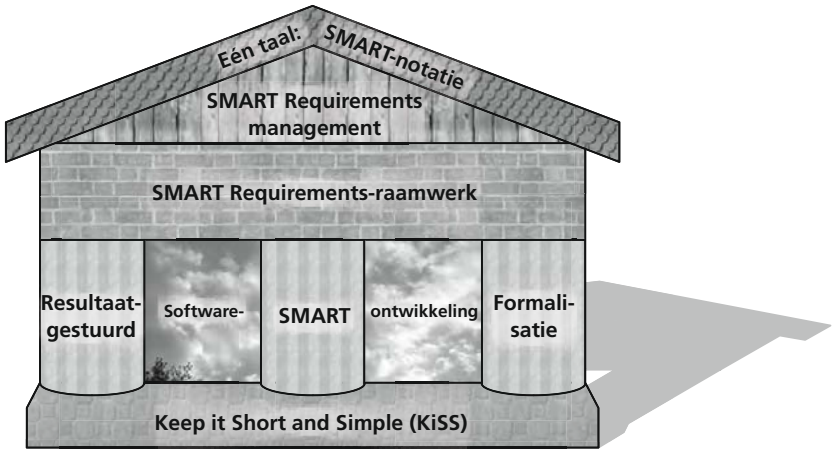
Afbeelding 1: Project Cartoon

Er zijn dus parallellen tussen het spel van de Chinese Whispers en de problemen waar we mee geconfronteerd worden binnen softwareontwikkeling. Ook uit onderzoek blijkt (Coughlan, Lycett & Macredie, 2005) dat effectieve communicatie een van de belangrijkste aspecten is van samenwerking tussen business en ICT. Het startpunt van dit boek is daarom:

> *Een gezamenlijk referentiekader en een gezamenlijke taal zijn essentieel voor een gedegen business & ICT alignment.*

We moeten in staat zijn om de regels van het spel naar onze hand te zetten. Wat bij de Chinese Whispers wordt verteld door de eerste in de rij, komt dan

overeen met het verhaal van de laatste in lijn. Daarom zijn wij van mening dat een gezamenlijk referentiekader (woordenboek) en een gezamenlijke taal (grammatica voor welgevormde zinnen) van cruciaal belang zijn. Zij zijn de fundering waarop we kunnen bouwen aan het verstevigen van de relatie tussen business en ICT. We hoeven dan niet langer een verhaaltje in elkaars oor te fluisteren, maar communiceren op gelijke voet en met open vizier. Dit doen we volgens vooraf vastgestelde kaders en met een taal die verschil in interpretatie en aannames overbodig maakt. Hierdoor worden we SMART, als in Specifiek, Meetbaar, Aanvaardbaar, Resultaatgeoriënteerd en Traceerbaar. Dat is de kern van SMART Requirements.



Afbeelding 2: Structuur SMART Requirements

In hoofdstuk 2 kijken we naar de pilaren waarop de methode gebouwd is. Met het SMART Requirements-raamwerk scheppen we de kaders in hoofdstuk 3. Om alles met elkaar te verbinden introduceren we het SMART Requirements-management in hoofdstuk 4. En ons bouwwerk wordt compleet gemaakt met één taal, in hoofdstuk 5. In hoofdstuk 6 duiken we nog wat dieper het proces van softwareontwikkeling in. En in hoofdstuk 7 kijken we naar hoe we SMART Requirements binnen een organisatie of project kunnen inpassen.

2. Pilaren van de methode

In dit hoofdstuk beschrijven we de basisprincipes die ten grondslag liggen aan SMART Requirements. Het zijn de uitgangspunten die we hebben toegepast bij het ontwikkelen van de methode. Hieronder geven we de theoretische achtergrond bij deze principes. In de volgende hoofdstukken werken we deze uit tot praktisch toepasbare technieken.

2.1 Softwareontwikkeling

Als we kijken naar een organisatie kunnen we deze zien als een verzameling van (bedrijfs)processen. Deze processen zijn niet statisch. Sterker nog, binnen de huidige dynamische en veranderlijke omgeving worden organisaties gedwongen continu te veranderen. Het (door)ontwikkelen van bedrijfsprocessen is hier een belangrijk onderdeel van. Binnen informatieverwerkende processen speelt ICT vrijwel altijd een prominente rol. We moeten dus constateren dat softwareontwikkeling een integraal en cruciaal onderdeel is van organisatieontwikkeling.

Softwareontwikkeling is een complexe aangelegenheid en zal dat in de toekomst ook steeds meer worden. Dit gezien ontwikkelingen in de markt zoals cloud computing, big data, integratie van systemen over ketens heen, bring your own device of werken op afstand. Het is daarom van belang dat de doelen van een organisatie, de bedrijfsprocessen die deze doelen realiseren en de systemen die dit ondersteunen volledig op elkaar aansluiten. En dit is nu precies waar het zo vaak mis gaat. Er is vaak sprake van onvoldoende afstemming tussen de verschillende stakeholders die betrokken zijn bij softwareontwikkeling. De lijst met eisen en wensen blijft ongecontroleerd groeien, waardoor doelstellingen vertroebelen en de kans op falen sterk toeneemt.

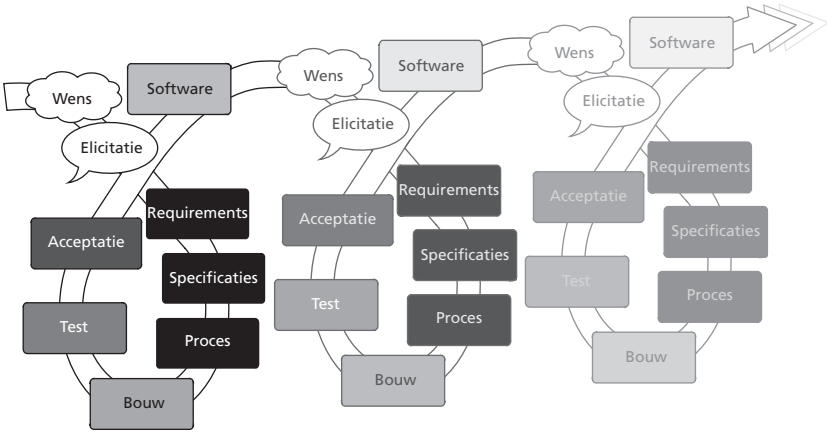
SMART Requirements gaat over het gecontroleerd SMART maken van eisen en wensen. Een essentieel onderdeel van softwareontwikkeling en daarmee dus ook van organisatie ontwikkeling van de moderne organisatie. We definiëren het speelveld van de methode daarom als **softwareontwikkeling**.

Let wel dat we dit proces breder zien dan alleen het ontwikkelen van een ICT-systeem. Zoals gezegd focust SMART Requirements zich in de eerste plaats op

de doelen, wensen en eisen van de business. Pas als we deze goed in het vizier hebben gaan we over tot de zaken die van belang zijn voor het ontwikkelen van de benodigde software.

> *Softwareontwikkeling omvat het gehele voortbrengingsproces van software dat is gebaseerd op de realisatie van de doelen, wensen en eisen van de business.*

Voordat we de details van de methode induiken, schetsen we eerst de manier waarop we met SMART Requirements naar softwareontwikkeling kijken. Afbeelding 3 laat daarom een generiek model voor softwareontwikkeling zien. Dit zijn de basale uitgangspunten die we hanteren als we het in dit boek hebben over softwareontwikkeling.



Afbeelding 3: SMART Requirements softwareontwikkeling

Om te beginnen zien we een aantal lussen. Deze representeren de verschillende ontwikkelcycli die we tijdens softwareontwikkeling kunnen doormaken. We kunnen deze lussen plotten op verschillende typen softwareontwikkeling:

- Als we strikt volgens waterval zouden werken, doorlopen we slechts één lus.
- Hanteren we een iteratieve aanpak, dan doorlopen we verschillende lussen na elkaar.
- Bij agile zijn het vele kleine lusjes die we aansluitend zullen doormaken.

De individuele stappen binnen de lussen blijven echter hetzelfde, onafhankelijk van welke aanpak we hanteren. Ook is de diepgang van de verschillende stapjes variabel. We hanteren daarom dit generieke model als uitgangspunt voor softwareontwikkeling. Een model dat te plotten is op de reeds bestaande technieken en aanpakken.

Wens en elicitatie
We beginnen nu linksbovenaan en kijken naar de individuele stapjes binnen een lus. We zien hier de wens van de business. Deze is altijd het startpunt van softwareontwikkeling met SMART Requirements. We willen immers software ontwikkelen die één op één overeenkomt met de wens van de business. Waar de wens nog weinig concreet is maken we deze tijdens de elicitatie tastbaar. Dit doen we door het verzamelen van bronnen, zoals bestaande documentatie, wetgeving, interviews en workshops, die de basis vormen voor de rest van het traject.

Requirements
De bronnen gebruiken we als basis om hier de requirements uit te destilleren. In SMART Requirements beginnen we altijd met het duidelijk in kaart brengen van het resultaat. Met de requirements geven we daarom puntsgewijs aan aan welke voorwaarden het te ontwikkelen resultaat moet voldoen. Een voorbeeld om dit te verduidelijken.

Op reis naar de Champs-Élysées
Stel jezelf het volgende voor. Je wilt met de auto naar de Champs-Élysées in Parijs. Als je de weg niet weet dan gebruik je tegenwoordig een navigatiesysteem. Je wilt niet zelf na hoeven denken met een kaart in je handen, een navigatiesysteem doet dat veel beter en sneller dan jij. Je voert enkel het adres in (wat) en het systeem berekent de route (hoe) en loodst je vervolgens moeiteloos naar de plek van bestemming.

De Champs-Élysées in Parijs als gewenste bestemming is in dit voorbeeld het requirement voor het resultaat. Over hoe we hier moeten komen, hoeven we ons in eerste instantie nog niet druk te maken. Dit regelt het navigatiesysteem voor ons. In deze kunnen we SMART Requirements zien als het navigatiesysteem. De methode helpt ons zowel in het eenduidig maken van het te bereiken resultaat, als ook bij het vormgeven van het proces oftewel de oplossing om dit resultaat te bereiken.

Je maakt het je navigatiesysteem nog wat lastiger op je reis naar de Champs-Élysées. Je voegt extra eisen toe aan de routes die wegen mogen worden. Je stelt het navigatiesysteem zo in dat je wilt reizen volgens de kortste route en ook wil je tolwegen vermijden.

Deze eisen zijn de voorwaarden waaraan de route moet voldoen. Binnen SMART Requirements zien we dit als requirements voor het uitvoeren van het proces, ofwel de beperkingen waar we rekening mee moeten houden bij het inrichten van het proces om het resultaat te bereiken.

In een wat geavanceerder navigatiesysteem ben je ook in staat extra eisen te stellen aan de middelen die je gebruikt. Zo kun je bijvoorbeeld besluiten om met de fiets te gaan in plaats van de auto. Dit heeft uiteraard consequenties voor onder andere de route en de reistijd.

Deze eisen zijn de voorwaarden waar de gebruikte middelen aan moeten voldoen. Binnen SMART Requirements zien we dit als requirements op de middelen, hetgeen in het ICT-vakgebied ook wel de requirements op de (ICT-) systemen worden genoemd.

Al dit soort wensen, eisen en voorwaarden noemen we binnen SMART Requirements de **requirements**. En uit bovenstaande voorbeelden blijkt dat we requirements opdelen in:

- requirements voor het resultaat;
- requirements voor de manier waarop het proces moet worden uitgevoerd; en
- requirements voor de (ICT-)systemen die gebruikt moeten worden.

Requirements zijn de eisen, wensen en voorwaarden waaraan een product of dienst (resultaat) of een proces of het te gebruiken systeem moeten voldoen.

Specificaties

SMART Requirements helpt bij het SMART maken van de requirements die we vastleggen. Een verdere detaillering van deze requirements werken we uit in specificaties. Deze beschrijven de manier waarop de requirements moeten worden vormgegeven en hoe zij worden doorgevoerd binnen de resultaten, processen en systemen.

Specificaties zijn de eenduidige uitwerkingen van de requirements.

SMART zijn betekent het volgende. Degene die informatie ontvangt is in staat om zijn werkzaamheden uit te voeren, zonder verdere achtergrondinformatie of het stellen van aanvullende vragen. Inhoudelijk experts (proces- en systeemontwerpers) zijn bijvoorbeeld afhankelijk van SMART geformuleerde requirements. Bouwers en testers zijn op hun beurt weer afhankelijk van SMART opgestelde specificaties. Het doel is dat zij allemaal in staat zijn om oplossingen te bedenken die het gewenste resultaat zullen gaan realiseren. Daarnaast moeten zij dit snel en effectief kunnen doen. SMART Requirements geeft hier duidelijke richtlijnen en technieken voor.

Proces
Voor het bereiken van het gewenste resultaat moet een proces ontworpen worden. Dit proces baseren we binnen SMART Requirements op twee dingen. De specificaties van het resultaat en de aanvullenden requirements die gaan over de manier waarop het proces moet worden uitgevoerd. Hierdoor verzekeren we onszelf ervan dat het een proces is dat alleen handelingen bevat die iets toevoegen aan het resultaat, die rekening houden met de geldende beperkingen en waarin niets mist. Vanuit de Lean-gedacht zeggen we dus dat we een proces ontwerpen dat een absoluut minimum aan 'waste' bevat.

Terugkomend op onze reis naar de Champs-Élysées…

Je hebt besloten dat je liever geen autoreis van zes uur wil maken naar de binnenstad van Parijs. Je wilt liever sneller op de plaats van bestemming zijn en ondertussen je handen vrij hebben, om tijdens de reis nog wat werk te kunnen doen. Je boekt daarom een vlucht. Als het zover is stap je in de auto en voert in je navigatiesysteem het adres van de luchthaven in, waardoor je wordt begeleid op je reis naar het vliegveld.

Je eindbestemming is uiteraard niet de luchthaven, maar de Champs-Élysées in Parijs. De luchthaven is dus slechts een onderdeel van de totale oplossing. Het hele proces is namelijk:
· met de auto naar de luchthaven;
· met het vliegtuig naar Parijs;
· met de taxi naar de Champs-Élysées.

Dit voorbeeld geeft aan dat we een gewenst resultaat hebben, de Champs-Élysées in Parijs. Daarnaast zijn er beperkingen voor het proces hoe we hier moeten komen, namelijk sneller reizen dan zes uur en ondertussen nog werk kunnen doen. Hierdoor moeten we enkele tussenresultaten toevoegen aan het proces om hier te komen. Omdat we onze focus op het resultaat scherp houden en daarnaast rekening houden met de eisen die aan het proces worden gesteld, kunnen we een proces ontwerpen met alleen handelingen die iets toevoegen en waarin niets mist. Dit geeft de ontwerpers van het proces houvast en richting bij het kiezen van de best passende oplossingen.

We hebben met requirements, specificaties en processen zogezegd drie hulpmiddelen in ons arsenaal, om het gewenste SMART-niveau te bereiken. Afhankelijk van de situatie en de gekozen ontwikkelaanpak (agile, iteratief of waterval) zijn we hierbij in staat om van deze hulpmiddelen gebruik te maken op bepaalde momenten in het ontwikkeltraject. We moeten ons namelijk realiseren dat het inzetten van deze hulpmiddelen en het SMART-niveau dat we willen bereiken, geld kost. Hoe meer zaken we uitwerken en hoe meer details we opleveren, hoe meer kosten dit met zich meebrengt. Het is daarom dat we steeds een afweging maken tussen de verdere uitwerking van requirements, specificaties en processen versus de risico's die we lopen als we onvoldoende details beschikbaar maken. In paragraaf 2.4 staan we stil bij de manier waarop we hier een afweging in maken.

We onderkennen daarnaast requirements, specificaties en processen op drie verschillende niveaus: business, proces en systeem. Binnen het businessniveau kijken we naar de requirements en specificaties van het resultaat van het gehele (bedrijfs)proces binnen scope. Op het procesniveau concentreren we ons op de menselijke actoren binnen het proces en de requirements, specificaties en gebruikersprocessen die hier van toepassing zijn. Op het systeemniveau kijken we naar de systeemactoren en de requirements, specificaties en systeemprocessen die hier van toepassing zijn. We maken dus een strikt onderscheid in de verschillende typen requirements, specificaties en processen. Dit houdt onze focus scherp en maakt het mogelijk een scheiding aan te brengen binnen de verantwoordelijkheden van de verschillende specialisten binnen softwareontwikkeling. In het volgende hoofdstuk 3 *SMART Requirements-raamwerk* komen we uitgebreid terug op de manier waarop we deze principes praktisch kunnen toepassen.

Bouw
SMART Requirements geeft ons de mogelijkheid om middels de requirements en specificaties volledig eenduidig te zijn over de gewenste resultaten en de hiervan afgeleide processen op business-, proces- en systeemniveau. Bouw

profiteert hiervan door het beschikbaar zijn van alle benodigde informatie. Bouwers kunnen, zonder extra vragen te stellen, direct aan de slag met het ontwikkelen en testen van de van de benodigde ICT-middelen.

Test
Wat voor bouw geldt, geldt ook voor test. Bovendien is gestructureerd testen ook resultaatgeoriënteerd. Het toepassen van SMART Requirements faciliteert het testen hierdoor nog verder.

Acceptatie
De business-wensen liggen vast in de vorm van bronnen. Uit deze bronnen worden requirements gedestilleerd. Deze requirements worden vervolgens geëvalueerd en geaccordeerd door de business. Zij worden hiermee het 'contract' op basis waarvan de oplossingen voor de business worden ontwikkeld. Acceptatie vindt plaats op basis van de door de business geaccordeerde requirements. Door het SMART maken van deze requirements hebben we dus ook SMART-acceptatiecriteria gecreëerd.

Software
Hiermee bedoelen we software die gebruikt kan worden in de dagelijkse praktijk, door de organisatie en de beoogde gebruikers.

2.2 Keep it Short & Simple

We hebben al enkele praktijkvoorbeelden voorbij zien komen met betrekking tot de problemen die zich kunnen voordoen binnen softwareontwikkeling. En zo zijn er natuurlijk nog veel meer voorbeelden te noemen. Bij bestudering van vele verschillende cases hebben wij daarom de volgende conclusie getrokken.

Om een organisatie, haar processen en software te veranderen of te verbeteren wordt vaak een berg met documenten geproduceerd. Echter, het komt regelmatig voor dat deze berg niet alle informatie bevat die werkelijk nodig is om de gewenste verandering te bewerkstelligen.

Uiteindelijk is deze informatie toch echt nodig, met twee mogelijke gevolgen:
1. het pak papier wordt nog groter en/of
2. de benodigde informatie is alleen aanwezig in de hoofden van een paar medewerkers.

Deze situaties zijn we zo vaak tegengekomen dat dit eerder een regel dan een uitzondering lijkt. Onderstaande case is hier een treffend voorbeeld van.

Kritieke medewerkers en wildgroei aan documentatie
Een organisatie heeft altijd een eigen ICT-afdeling gehad. Om strikter te kunnen werken volgens het klant/leverancierprincipe gaat de afdeling echter verzelfstandigen. Bij deze transitie wordt veel aandacht gegeven aan de personele consequenties en aan de eisen die gesteld worden aan de dienstverlening van deze ICT-afdeling. Er wordt echter onvoldoende nagedacht over de manier waarop de communicatie op het gebied van ICT-wijzigingen en -nieuwbouw tussen de twee organisaties moet worden ingeregeld.

Dit heeft voor ons twee interessante gevolgen. De ene is dat er continue uitwisseling nodig is van medewerkers tussen de verschillende organisaties. Men is immers niet goed in staat de kennis op een andere manier uit te wisselen dan vanuit de 'hoofden' van een aantal sleutelfiguren binnen de organisatie. Er is weinig voor nodig om te bedenken dat zij voor vrijwel ieder project kritieke resources blijken te zijn. Hierdoor raken deze medewerkers overbezet en overbelast en lopen projecten uit.

Een tweede punt is dat er op het raakvlak tussen beide organisaties een wildgroei aan documentatie plaatsvindt. Er bestaan 15 formulieren en sjablonen die over en weer moeten worden ingevuld en goedgekeurd. Hierop komt ook nog eens veel dubbele of tegenstrijdige informatie voor en er moeten steeds weer een hoop handtekeningen worden verzameld ter goedkeuring. Je kunt je voorstellen dat dit vreselijk veel tijd kost. Onnodige ballast ten gunste van schijnzekerheid.

Tenzij we ons van dit soort mechanismes erg goed bewust zijn, ontstaan dergelijke situaties als mensen geconfronteerd worden met grote en ingewikkelde uitdagingen. Management en medewerkers weten niet goed waar ze moeten beginnen en zijn onzeker of ze wel tot een succesvol einde kunnen komen.

Dit zet de probleemeigenaren aan tot het strooien met mechanismen van houvast en controle. In sommige gevallen worden deze effectief ingezet, maar vaak levert dit slechts een berg met documentatie op. Dit geeft de schijn van zekerheid en wekt de suggestie van het behouden van de regie. Op deze manier sleept men onbewust en onbedoeld vaak veel onnodige ballast een organisatie of project in. Zo ook bij onderstaande voorbeeld.

Onnodige ballast

Het besturen van een organisatie gaat gepaard met grote hoeveelheden gegevens. Dit is informatie die vaak zo omvangrijk en complex is dat we deze maar moeilijk kunnen overzien. Daarom worden de grote lijnen van wat er zich binnen een organisatie afspeelt, gevangen binnen rapportages. Deze zijn gebaseerd op geaggregeerde gegevens en bevatten overzichten die managers moeten helpen bij het nemen van de juiste beslissingen.

Dit is een goede zaak. Echter, het genereren van gegevens en het verzamelen ervan binnen rapportages heeft alleen dan zin wanneer hier daadwerkelijk wat mee gedaan wordt. Rapportages verdwijnen maar al te vaak ongelezen in een la en als deze al worden gelezen, wordt hier regelmatig helemaal niet op gestuurd.

Op een gegeven moment worden de werknemers geconfronteerd met het registreren van uren binnen een in eigen huis ontwikkelde tijdschrijfsysteem. Dit blijkt een behoorlijke opgave te zijn, die veel tijd opslokt. Het systeem is bijzonder gebruikersonvriendelijk, maar belangrijker is de grote hoeveelheid gegevens die de werknemers moeten invoeren. Sommige velden bevatten zelfs honderden (!) codes die de werknemers moeten doorbladeren, voordat ze bij de juiste komen.

Een paar nieuwsgierige medewerkers gingen op onderzoek. Wat was immers de noodzaak voor het genereren van zulke grote hoeveelheden informatie? Uiteindelijk kwamen zij tot de schrikbarende conclusie dat er helemaal niets met deze gegevens wordt gedaan. Ooit had het hoger management besloten dat er tijd moest worden geregistreerd zodat zij 'in control' konden komen. Dit had geleid tot een verregaande opdeling in codes en subcodes. Een duidelijk voorbeeld van 'micromanagement' met als gevolg een systeem zonder toegevoegde waarde.

Achteraf gezien zullen velen het erover eens zijn dat dit verspilde moeite is. Het gevoel van zekerheid dat het management overhoudt aan het idee van het genereren en verzamelen van gegevens wordt op deze manier verheven tot een doel op zich. En wellicht zullen de hiervoor genoemde voorbeelden je bekend in de oren klinken. Maar het wordt lastig als we hier ook echt wat aan willen doen. Onze conclusies gaven ons echter een duw in de goede richting:

Richt je primair op wat je wilt bereiken: kijk naar het gewenste resultaat in plaats van naar alles tegelijk.

We moeten ons dus alleen bezighouden met de zaken die van belang zijn voor het gewenste resultaat. Kijk bijvoorbeeld kritisch naar processen en de tussenresultaten die worden opgeleverd. Bepaal of deze nog waarde toevoegen en of zij daarom meegenomen moeten worden bij softwareontwikkeling.

Als we resultaten vervolgens vastleggen, doen we dat zo eenvoudig mogelijk. Er is een principe, dat zijn oorsprong vindt in de techniek en softwareontwikkeling, dat naadloos op deze gedachtegang aansluit:

Keep it Short & Simple (KiSS) / Hou het kort en bondig.

Steeds voordat we een stap wilden zetten binnen de ontwikkeling van SMART Requirements vroegen wij ons af of deze kort en bondig zou zijn. We zijn er namelijk van overtuigd dat het al moeilijk genoeg is om ons te richten op wat we willen bereiken; zeker als we te maken krijgen met complexe situaties die van zichzelf al om complexe oplossingen kunnen vragen. Daarom vinden wij eenvoud in dit boek zo belangrijk. Of, zoals Albert Einstein het zei: “Maak de dingen zo eenvoudig mogelijk, maar niet eenvoudiger dan dat.”

2.3 Resultaatgeoriënteerd

Denken in termen van een te bereiken resultaat is meestal een stuk eenvoudiger dan denken in termen van hoe dit resultaat moet worden bereikt. Het is bijvoorbeeld veel gemakkelijker om aan een taxichauffeur duidelijk te maken waar hij je naartoe moet brengen, in plaats van hem de weg ernaartoe uit te leggen. Zeker als je ook nog eens in een onbekende stad bent waar je de weg niet kent.

Ieder proces moet iets opleveren. We noemen dit het **resultaat**.

Het resultaat is dat wat een proces moet opleveren.

Voordat we starten met het bedenken van de oplossingen moeten we onszelf dus verzekeren van *wat* het resultaat moet zijn van bijvoorbeeld het systeem, product of dienst dat we gaan ontwikkelen. Dit is de meest fundamentele

vraag die we onszelf moeten stellen. We zien namelijk regelmatig gebeuren dat veel te snel wordt overgegaan naar *hoe* het resultaat moet worden bereikt. Op deze manier gaat in een project alle aandacht naar de manier waarop het resultaat wordt bereikt en niet naar het te bereiken resultaat. De focus van projecten wordt hierdoor vertroebeld, waardoor de kans dat zij falen aanzienlijk toeneemt. Veel organisaties zijn zich hier onvoldoende van bewust.

Begin altijd met wat het beoogde resultaat is. De oplossing, hoe dit resultaat moet worden bereikt, leidt je hiervan af.

We worden regelmatig geconfronteerd met een vraag van de business, die betrekking heeft op een specifieke oplossing. Denk hierbij aan: "Wij willen gebruik gaan maken van tool X" of "Onze problemen willen we het hoofd bieden door een systeem te ontwikkelen dat dit of dat kan." Dat is denken in termen van een bepaalde oplossing. Echter, hierbij wordt vaak onvoldoende stil gestaan bij het gewenste resultaat dat deze oplossing moet gaan bereiken. En ook al is er bijvoorbeeld een andere afdeling of een concullega die werkt met dezelfde tool, jouw gewenste resultaat is toch vaak anders dan dat van iemand anders. We moeten daarom altijd doorvragen naar het achterliggende probleem of, uitgedrukt in termen van het resultaat, naar welk resultaat we nu eigenlijk willen gaan bewerkstelligen.

Middelen als doel

Een afdeling binnen een grote organisatie ziet zich geconfronteerd met de volgende uitdaging: er is te weinig grip op de manier waarop projecten worden uitgevoerd. De gerealiseerde oplossingen zijn te duur, worden te laat opgeleverd en bieden vaak niet de gewenste functionaliteit.

Om dit probleem het hoofd te bieden wordt gekeken naar een softwarepakket dat het mogelijk maakt project- en portfoliomanagement uit te voeren. Een andere afdeling heeft hier immers ook goede resultaten mee behaald. Er wordt gekozen voor hetzelfde pakket, dat zeer veel mogelijkheden biedt. Het maakt namelijk mogelijk projecten in detail te plannen en over projecten heen gegevens te verzamelen, zodat het projectportfolio optimaal kan worden beheerd.

Er wordt hard gewerkt aan het organisatiebreed beschikbaar maken van dit pakket. Zo wordt er geïnvesteerd in het optimaal inrichten van het systeem en er moet maatwerk worden toegevoegd om het specifiek bruikbaar te maken voor de verschillende afdelingen.

Bij het in gebruik nemen van het pakket loopt men binnen de afdeling al snel tegen nieuwe uitdagingen aan. De projectleiders zijn namelijk veel tijd kwijt met het steeds up to date houden van gegevens en planningen. Projectleden worden geacht hun tijd te registreren en forecasts te geven van het werk dat zij nog moeten doen. Dit stuit op veel weerstand, waarbij het argument wordt gegeven dat zij nu meer tijd kwijt zijn dan eerst met zaken die hun problemen niet oplossen.

Bij een evaluatieronde blijkt dat de andere afdeling, die de tool wel succesvol weet in te zetten, een duidelijke regierol heeft. Zij heeft er baat bij om gegevens tot in detail vast te leggen, omdat hun resultaat hiervan afhankelijk is. In deze afdeling speelden echter heel andere problemen. Het ging hier bijvoorbeeld over het niet goed maken van afspraken en nakomen hiervan. Ook is er onvoldoende zicht op wanneer welke resources aan welk project zijn toebedeeld. Een simpel planningsvraagstuk.

Door het te snel kiezen voor een bepaalde oplossing is voor iets gekozen dat het werk alleen maar lastiger maakt. Terwijl het onderliggende probleem niet wordt aangepakt. Als men eerst had nagedacht over het gewenste resultaat – beter afspraken maken en nakomen en zicht op wie, wanneer bij welk project is ingedeeld – had dat waarschijnlijk geresulteerd in een beter passende oplossing.

De vraag die centraal zou moeten staan bij de start van ieder project is: *Wat moet het proces als resultaat opleveren?* Hoe dit resultaat bereikt moet worden is hier vervolgens een afgeleide van. Dit noemen we een **resultaatgedreven aanpak.**

Met SMART Requirements kijken we eerst naar het gehele (bedrijfs)proces dat in scope is en stellen onszelf de vraag welk resultaat dit proces moet opleveren. We kijken dus nog niet naar hoe het bedrijfsproces eruit moet zien of welke handelingen hierin worden uitgevoerd, maar concentreren ons volledig op het resultaat. Dit maken we vervolgens geheel duidelijk en transparant. Hierna kijken we pas naar de menselijke actoren en de gebruikersprocessen die zij uitvoeren om dit resultaat te realiseren. Vervolgens stellen we onszelf opnieuw de vraag wat ieder gebruikersproces als resultaat moet opleveren. Uiteindelijk kijken we naar de verschillende systeemactoren en systeemprocessen die worden uitgevoerd en welk resultaat zij moeten opleveren. Al deze beschrijvingen worden gevangen in het SMART Requirements-raamwerk (zie hoofdstuk 3) en tezamen bepalen zij hoe de uiteindelijke oplossing eruit komt te zien.

We willen benadrukken dat het redeneren in termen van het resultaat een omslag vergt in de manier van denken. Hoewel resultaatgeoriënteerde methoden de laatste jaren in opmars zijn, leert de praktijk dat veel organisaties nog maar net gewend zijn geraakt aan het denken in processen, van hoe iets tot stand moet komen. Door het resultaat centraal te stellen en andere zaken in eerste instantie uit te sluiten kunnen we echter voorkomen dat een project onnodig complex wordt. Iets dat natuurlijk perfect aansluit op onze KiSS-filosofie.

2.4 SMART

Voor het bereiken van een bepaald resultaat moeten we zekerheid hebben over hoe dit eruit moet zien. Doen we dit niet, dan wordt het schieten op een bewegend doel en het gewenste resultaat zal zeer waarschijnlijk uitblijven. Een taxichauffeur een straat en huisnummer geven werkt immers beter dan hem vertellen dat je eindbestemming ergens in de buurt van een groot plein is. Dit noemen we: SMART.

Het SMART-principe is ontstaan binnen het projectmanagement en wordt inmiddels veel breder toegepast. Het staat voor het eenvoudig en eenduidig opstellen en controleren van resultaten. Met SMART geformuleerde resultaten kunnen we sturen op vooraf duidelijk geformuleerde afspraken. Binnen SMART Requirements hanteren we de volgende uitleg van het SMART-acroniem:

SMART	*Staat voor*	*Uitleg*
S	*Specifiek*	*Het resultaat is volledig eenduidig gedefinieerd.*
M	*Meetbaar*	*Er kan worden geverifieerd dat het resultaat overeenkomt met datgene wat is afgesproken.*
A	*Aanvaardbaar*	*Alle relevante belanghebbenden kunnen instemmen en hebben ingestemd met het gewenste resultaat.*
R	*Resultaat-georiënteerd*	*De focus ligt primair op het resultaat, waarbij het proces en de procesondersteunende middelen hiervan zijn afgeleid.*
T	*Traceerbaar*	*Requirements vormen de schakel tussen bron en realisatie, waardoor altijd te herleiden is waarom voor bepaalde oplossingen gekozen is.*

Register

Veel organisaties kampen met dezelfde vraag: "Hoe kunnen we het beste onze wensen omschrijven zodat de ICT-afdeling begrijpt wat we nodig hebben?" Dit boek beschrijft de methode SMART Requirements, waarmee concreet invulling wordt gegeven aan deze vraag en waarmee bestaande Agile-, iteratief- of waterval georiënteerde softwareontwikkeling wordt verbeterd.

Requirements spelen binnen iedere softwareontwikkelmethode of -aanpak een cruciale rol. Wanneer requirements eenduidig, ofwel SMART worden opgesteld, leidt dit tot een substantiële afname van 'herstelwerk' tijdens de bouw- en testfase. Ontwikkelaars kunnen direct en efficiënt aan de slag en leveren kwalitatief beter werk. Testers kunnen testen tegen betrouwbare specificaties. Hierdoor worden de risico's en kosten van softwareontwikkeling aanzienlijk gereduceerd.

Dit boek is bestemd voor iedereen die te maken heeft met:

- deadlines die niet worden gehaald binnen softwareontwikkeling;
- te hoge ICT-ontwikkelkosten;
- software die niet voldoet aan de eisen en wensen van de organisatie;
- software die onvoldoende getest kan worden.

Over de auteurs

Ömer Aydinli heeft Informatiekunde gestudeerd aan de Universiteit Utrecht. Hij heeft verschillende wetenschappelijke artikelen gepubliceerd op gebied van business process improvement. Ömer werkt als teamleider binnen de Business Information Analysis (BIA) en Business Process Management (BPM) afdeling van CGI. Daarnaast adviseert hij klanten op het gebied van Business IT alignment, business process improvement en requirements engineering.

Edwin Hendriks heeft informatica gestudeerd aan de Radboud Universiteit. Hij heeft na zijn studie veel rollen vervuld binnen zowel business als ICT en in het bijzonder binnen softwareontwikkeling. Uit zijn passie om softwareontwikkeling zo efficiënt en eenvoudig mogelijk te maken, is hij gestart met het vergaren, combineren en ontwikkelen van verschillende werkwijzen en tools die de hele cyclus van wens tot een werkend systeem optimaliseren.

Jasper Zandvliet is na het voltooien van zijn studie technische bedrijfskunde werkzaam als organisatie adviseur bij verschillende organisaties. Hij richt zich hierbij op het raakvlak tussen mens, organisatie en techniek. Het bij elkaar brengen van verschillende ideeën en deze laten leiden tot concrete resultaten is zijn grote passie. Hierbij ligt zijn expertise bij het organiseren van mensen en werkzaamheden, met stevige kennis en ervaring binnen de ICT.

CGI

ISBN 978 90 12 585 83 5
NUR 980

9 789012 585835

ACADEMIC SERVICE

www.academicservice.nl